Final Project

on

A Data driven story of Airbnb in MA & NY

ALY6015 21282 Intermediate Analytics SEC 11

Winter 2019

submitted to

Prof. Valeriy Shevchenko

Submission date: February 17, 2019

Submitted by: Shikhar Chhabra

Table of Contents

Introduction to dataset

For our case study on Airbnb listings in the states of New York and Massachusetts, we have collected the data from Kaggle and Inside Airbnb.

1. Massachusetts Airbnb listings source: https://www.kaggle.com/airbnb/boston/version/1#listings.csv

2. New York Airbnb listings source: http://insideairbnb.com/get-the-data.html

The Massachusetts and New York Airbnb listings have been cleaned and compiled in a single csv file named 'NYMA_listings'. The different attributes of the dataset have been listed and described below:

1. **Id** - Unique identifier corresponding to each listing
2. **host_id** - Unique identifier corresponding to each host
3. **street** - Street name of a listing
4. **neighbourhood_cleansed** - Neighborhood name of each listing
5. **city** - City name of each listing
6. **state** – Name of the state where a listing is located
7. **zipcode** - zipcode of each listing
8. **country** - Country name of each listing
9. **latitude** - Latitude corresponding to each listing
10. **longitude**- Longitude corresponding to each listing
11. **property_type** - Type of property corresponding to each listing
12. **room_type** - Type of room corresponding to each listing i.e. private room, shared room or entire apartment
13. **accommodates** - Maximum number of persons permitted in a listing
14. **bathrooms** - Number of bathrooms in a listing
15. **bedrooms** - Number of bedrooms in a listing
16. **beds** - Number of beds in a listing
17. **bed_type** - Type of bed in a listing
18. **price** - This is the price corresponding to each listing
19. **currency** - Currency in which price is listed for each accommodation
20. **review_scores_rating** – Rating score of each listing out of 100

Objectives

The objectives of the project on ‘Airbnb listings in NY and MA’ are achieved by finding answers to the following research questions:

1. Is there a statistical significance between mean price of a 3-bedroom Airbnb listing in NY and MA?
2. Is there a statistical significance between the variance in price of a 3-bedroom Airbnb listing in NY and MA?
3. Is the claim made by Airbnb on its website that the average price of a 3-bedroom Airbnb listing is $227* in NY, true?
4. Is the claim made by Airbnb on its website that the average price of a 3-bedroom Airbnb listing is $338* in MA, true?
5. Is there a linear relationship between price (response variable) and its predictors such as id, host_id, street, neighbourhood_cleansed, city, state, zipcode, latitude, longitude, property_type, room_type, accommodates, bathrooms, beds, bed_type and review_scores_ratings.
6. What is the predicted price of an Airbnb listing in New York and Massachusetts, respectively, based on our training data?
7. How accurate is our predictive linear regression model and what are the interpretations of our findings?
8. How many relevant variables exist that determine our response variable i.e. price of an Airbnb listing in NY and MA, both?
9. What is the best classification method out of linear discriminant analysis and K nearest neighbor on Massachusetts data and New York's data?
10. What are the optimal number of clusters based on number of beds and price, in MA and NY, using K-means clustering?

Research Methodology

We have addressed our research questions using the following statistical techniques:

1. Inferential Statistics using Point estimation – Hypotheses Testing
2. Multiple Linear Regression
3. Regularization using LASSO model
4. Data Mining - LDA classification, K Nearest Neighbors Classification and K means Clustering

1. Point Estimation – Hypotheses Testing

Point estimation is an essential part of inferential statistics that involves the process of finding an approximate value of a parameter by drawing random samples from a population. A population parameter that we are trying to estimate can be a population mean or population standard deviation. Whereas, the point estimator, drawn from a random sample, that we are using to estimate a given population parameter is called a statistic, e.g. sample mean, sample standard deviation, etc. The prime goal of inferential statistics, hence, is to infer about a population parameter using a statistic (estimated by drawing a random sample from a given population). Hence, using inferential statistics, we estimate a statistic that represents the true value of a given population parameter.

Hypothesis Testing is a point estimation technique and an essential part of inferential statistics that involves the process of decision making for evaluating a given claim about a population from a given sample. In other words, the concerned statistics are calculated from a given sample to draw inferences about a population parameter. It is an essential part of inferential statistics that helps researchers make informed decisions about a population.

To test claims associated with our Airbnb dataset, we have used a t-test and f-test in the following section.

Loading data

To begin with, we have first loaded our csv file 'NYMAlistings.csv' in R using the read.csv() function as shown below:

```r
#1) Loading data
nymalistings <- read.csv("NYMAlistings.csv", header = TRUE)
nymalistings
```

Filtering the data

We have performed an analysis on the overnight prices of 3-bedroom Airbnb listings in New York and Massachusetts. After loading the data, we have defined our variables (price) and criteria (3-bedroom) for NY and MA, using subset() function in R, as shown below:

```r
#2) Filtering data by state  and no. of bedrooms for NY
nylistings <- subset(nymalistings, state=='NY'& bedrooms=='3')
nylistings
```

```r
#3) Filtering data by state  and no. of bedrooms for MA
malistings <- subset(nymalistings, state=='MA' & bedrooms=='3')
malistings
```

The subset() function filters the dataset according to the criteria mentioned along with it. In our case, we have created two subsets, nylistings and malistings, from the original data nymalistings. We have set bedrooms equal to 3 and state equal to NY and MA, respectively. As a result, we get the following results for NY and MA, each.

Output: NY

```
> #2) Filtering data by state  and no. of bedrooms for NY
> nylistings <- subset(nymalistings, state=='NY'& bedrooms=='3')
> nylistings
         id host_id                          street neighbourhood_cleansed       city state zipcode        country
3603  23686   93790     New York, NY, United States           West Village   New York    NY   10014 United States
3617  26012  109589     Brooklyn, NY, United States                Gowanus   Brooklyn    NY   11217 United States
3637  26969  115307     Brooklyn, NY, United States           Williamsburg   Brooklyn    NY   11249 United States
3643   8343   24222     New York, NY, United States           East Village   New York    NY   10009 United States
3682  31902  137292     Brooklyn, NY, United States              Flatlands   Brooklyn    NY   11234 United States
3686  32100  138579     Brooklyn, NY, United States             Greenpoint   Brooklyn    NY   11222 United States
3795  59121  204539       Queens, NY, United States              Ridgewood     Queens    NY   11385 United States
3802  60164  289653     New York, NY, United States                   SoHo   New York    NY   10013 United States
3813  60794  293394     New York, NY, United States        Upper West Side   New York    NY   10025 United States
3833  80924  438133     Brooklyn, NY, United States             Park Slope   Brooklyn    NY   11215 United States
3857  84059  459054     Brooklyn, NY, United States          Crown Heights   Brooklyn    NY   11216 United States
3886  68765  282655     Brooklyn, NY, United States        Carroll Gardens   Brooklyn    NY   11231 United States
3890  68974  281229     New York, NY, United States           Little Italy   New York    NY   10002 United States
3892 101053  530032     Brooklyn, NY, United States           Williamsburg   Brooklyn    NY   11211 United States
3901  70381  356484     New York, NY, United States                   SoHo   New York    NY   10013 United States
3903  70609   72062     New York, NY, United States           East Village   New York    NY   10009 United States
```

```
4816 508154 1559494      Brooklyn, NY, United States            Williamsburg      Brooklyn     NY   11211 United States
     latitude longitude property_type       room_type accommodates bathrooms bedrooms beds bed_type square_feet price
3603 40.73096 -74.00319         House Entire home/apt            5       2.0        3    3 Real Bed          NA   500
3617 40.68157 -73.98989     Townhouse Entire home/apt            6       2.0        3    3 Real Bed          NA   200
3637 40.71942 -73.95748         House Entire home/apt            6       1.5        3    4 Real Bed          NA   295
3643 40.72481 -73.98057   Condominium Entire home/apt            7       1.5        3    3 Real Bed          NA   272
3682 40.63188 -73.93248         House    Private room            2       1.0        3    1 Real Bed          NA    77
3686 40.73409 -73.95348     Apartment Entire home/apt            5       1.0        3    3 Real Bed          NA   275
3795 40.70411 -73.89934     Apartment Entire home/apt            9       1.0        3    1 Real Bed        1100   140
3802 40.72003 -74.00262          Loft Entire home/apt            6       1.0        3    3 Real Bed        2000   500
3813 40.80021 -73.96071     Apartment Entire home/apt            6       1.0        3    3 Real Bed          NA   195
3833 40.67542 -73.98142     Townhouse Entire home/apt            5       1.5        3    5 Real Bed          NA   163
3857 40.67591 -73.94715     Apartment Entire home/apt            6       1.5        3    3 Real Bed          NA   150
3886 40.67817 -73.99495     Apartment Entire home/apt            5       1.0        3    5 Real Bed          NA   250
3890 40.71943 -73.99627          Loft Entire home/apt            8       1.0        3    3 Real Bed        1500   575
3892 40.71125 -73.95613     Apartment    Private room            6       2.0        3    3 Real Bed          NA    80
3901 40.72195 -74.00356     Apartment Entire home/apt            5       2.0        3    3 Real Bed          NA   450
3903 40.72542 -73.97986     Apartment Entire home/apt            7       2.0        3    6 Real Bed          NA   500
3918 40.67539 -73.96093     Apartment Entire home/apt            6       1.0        3    4 Real Bed         900   165
3952 40.68480 -73.96219     Apartment Entire home/apt            5       1.5        3    5 Real Bed        2000   350
```

MA

```
> malistings
          id  host_id                                             street neighbourhood_cleansed                city state zipcode       country latitude
24   6400432 23127285            Metropolitan Ave, Boston, MA 02131, United States              Roslindale              Boston    MA    2131 United States 42.27838
38   8548176  6570877            Kittredge Street, Boston, MA 02131, United States              Roslindale              Boston    MA    2131 United States 42.27918
39   4922204  6570877 Kittredge St, Roslindale, Boston, MA 02131, United States              Roslindale Roslindale, Boston    MA    2131 United States 42.28214
56  12927298 68001856                Knoll Street, Boston, MA 02131, United States              Roslindale              Boston    MA    2131 United States 42.29062
97   4767023 24593919            Edge Hill Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.32328
112   950046  5153431      Gordon Street, Jamaica Plain, MA 02130, United States           Jamaica Plain       Jamaica Plain    MA    2130 United States 42.31185
114  5652147   352441              Ballard Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.30705
125  3881993 11487565              Rossmore Road, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.30434
130  6053700 31247560       Eliot Street, Jamaica Plain, MA 02130, United States           Jamaica Plain       Jamaica Plain    MA    2130 United States 42.30983
135 12368012 66824316            Carolina Avenue, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.30872
155  9057435 47251069           Burroughs Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.31434
183  5025015 25932315                 Day Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.32505
208  9906565  6807512           Brookside Avenue, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.31098
235  6298216 32750893             Boylston Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.31923
246 10004575  3130698              Pershing Road, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.31969
279   957224  4210157    Moraine Street, Jamaica Plain, MA 02130, United States           Jamaica Plain       Jamaica Plain    MA    2130 United States 42.31886
280  4119345  1414385               Green Street, Boston, MA 02130, United States           Jamaica Plain              Boston    MA    2130 United States 42.31162
281  5495547  2753598              Malcolm Road. Boston. MA 02130. United States           Jamaica Plain              Boston    MA    2130 United States 42.30189
```

```
    longitude property_type       room_type accommodates bathrooms bedrooms beds bed_type square_feet price currency review_scores_rating  X
24  -71.12878         House Entire home/apt            8       1.0        3    5 Real Bed          NA   150  Dollars                   88 NA
38  -71.13277         House Entire home/apt           16       1.0        3    5 Real Bed          NA   125  Dollars                   91 NA
39  -71.12905         House Entire home/apt           16       2.5        3    7 Real Bed          NA   200  Dollars                   95 NA
56  -71.13273         House Entire home/apt            7       2.0        3    3 Real Bed          NA   285  Dollars                   80 NA
97  -71.10706         House Entire home/apt            5       2.0        3    4 Real Bed          NA   205  Dollars                   87 NA
112 -71.10926     Apartment Entire home/apt            6       2.0        3    3 Real Bed          NA   250  Dollars                   90 NA
114 -71.11662         House Entire home/apt            7       2.0        3    4 Real Bed          NA   300  Dollars                   93 NA
125 -71.10872     Apartment Entire home/apt            6       1.0        3    3 Real Bed          NA   500  Dollars                   89 NA
130 -71.11458         House Entire home/apt            6       3.0        3    6 Real Bed          NA   450  Dollars                  100 NA
135 -71.11052         House Entire home/apt            6       2.0        3    4 Real Bed          NA   295  Dollars                   NA NA
155 -71.11641     Apartment Entire home/apt            6       1.0        3    3 Real Bed          NA   319  Dollars                  100 NA
183 -71.10630     Apartment Entire home/apt            5       1.0        3    4 Real Bed          NA   175  Dollars                   85 NA
208 -71.10416     Apartment Entire home/apt            7       1.5        3    4 Real Bed          NA   225  Dollars                   NA NA
235 -71.10887     Apartment Entire home/apt            6       1.0        3    3 Real Bed          NA   299  Dollars                   NA NA
246 -71.11297     Apartment Entire home/apt            6       1.0        3    5 Real Bed          NA   300  Dollars                   97 NA
279 -71.11452         House Entire home/apt            8       1.0        3    4 Real Bed          NA   275  Dollars                   93 NA
280 -71.10910         House Entire home/apt            8       2.5        3    3 Real Bed          NA   300  Dollars                  100 NA
281 -71.12908         House Entire home/apt            5       1.5        3    3 Real Bed          NA   185  Dollars                  100 NA
292 -71.12399     Apartment Entire home/apt            6       2.0        3    3 Real Bed          NA   168  Dollars                   88 NA
```

Removing outliers

After filtering the dataset with the required variables and criteria, we have removed outliers such that the observations in our data set our not skewed as normal distribution is one of the major assumptions of a t-test. We have used the **'outliers'** package and library to install the rm.outliers() function in R.

```
#4) Remove outliers NY
install.packages("outliers")
library(outliers)
cleannylistings <- rm.outlier(nylistings$price, fill = FALSE, median = FALSE, opposite = FALSE)
cleannylistings

#5) Remove outliers MA
cleanmalistings <- rm.outlier(malistings$price, fill = FALSE, median = FALSE, opposite = FALSE)
cleanmalistings
```

Checking normality

A t-test is one of the most widely used tests by statisticians in which a test statistic follows a Student's t-distribution. It is popularly used to test if there exists a significant difference between two sample means. Two basic underlying assumptions of a t-test are:

1. Population from which a sample is drawn follows a standard normal distribution
2. Population deviation is unknown

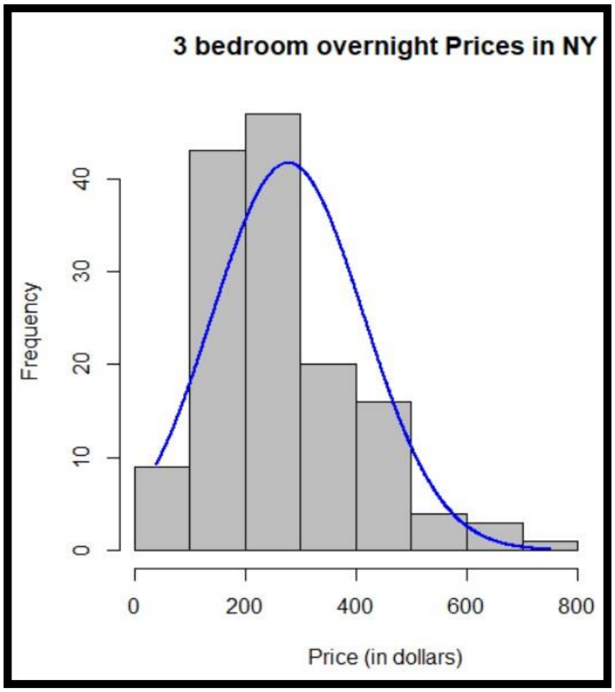
The second assumption is true for our data, however, we are yet to discover whether our populations of 3-bedroom Airbnb listings in NY and MA approximate towards the standard normal distribution or not.

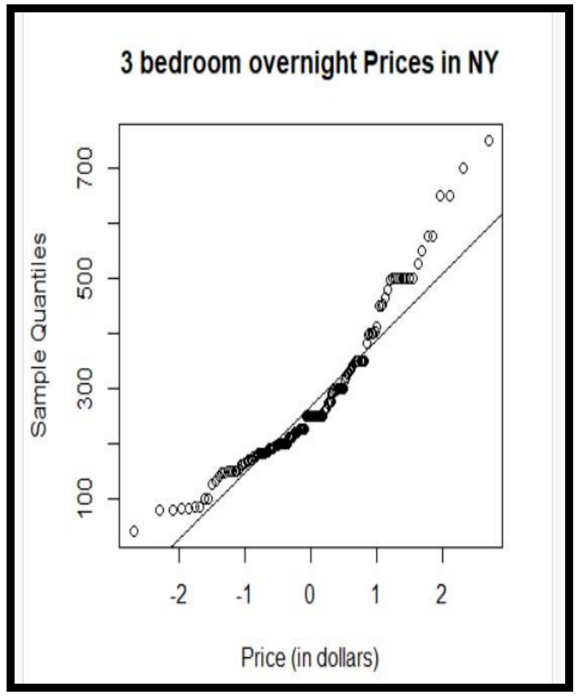
Therefore, we perform a procedure for checking the normality of our populations. We have used the qqnorm, qqline and histogram plots for observing an approximately normal trend in our populations of NY and MA.

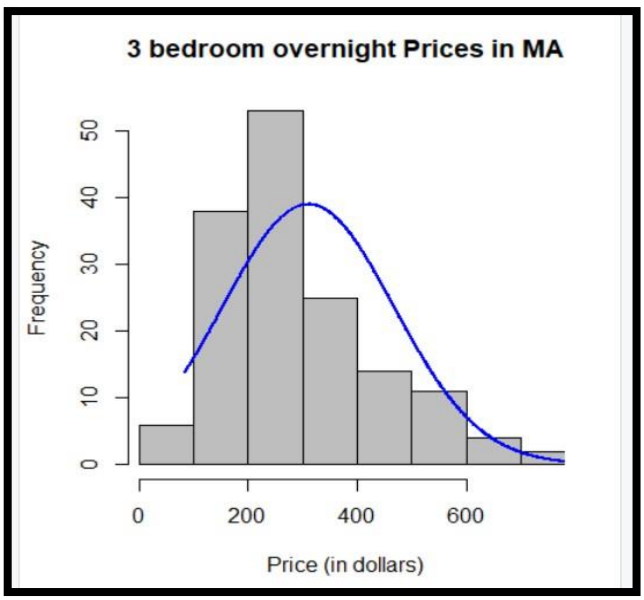
```
#6) Check normality NY
qqnorm(cleannylistings,main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)')
qqline(cleannylistings,main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)')
install.packages("rcompanion")
library(rcompanion)
plotNormalHistogram(cleannylistings, main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)', xlim = c(10,900))

#7) Check normality MA
qqnorm(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)')
qqline(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)')
plotNormalHistogram(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)', xlim = c(10,750))
```

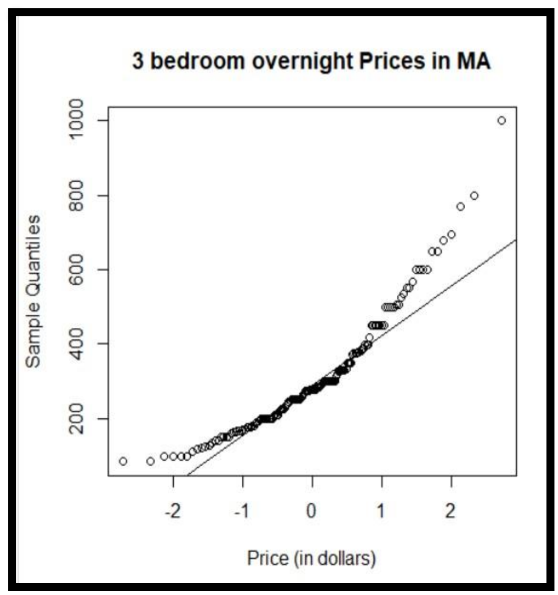
The qqnorm() function plots the mentioned values against a normal distribution. The qqline adds a line to the qqnorm plot that passes through the first and third quartiles. The plotNormalHistogram() function plots a histogram against the mentioned values and produces a curve that follows the shape of that histogram. In case of a normal distribution, the curve has a bell shape that represents a normal distribution.

To run the plotNormalHistogram() function, we have installed the package and library named **'rcompanion'**.









As we can observe from the histogram and qqnorm plots, the population data based on the overnight prices of Airbnb listings of 3 bedroom in MA and NY are slightly skewed to the right. We used square, cube-root and log transformations to completely normalize our populations, however, the results of the p-value were very different using each transformation. Hence, we decided to use the t-test as the histogram plots for each population, MA and NY, have a bell-shaped curve that have their means located not exactly in the middle, but somewhere around the center of the two datasets. Therefore, most real-world datasets are not exactly normally distributed and only approximate towards normal distribution, we apply t-test to our case study.

Next, we have drawn 30-random samples from 3-bedroom Airbnb listings of NY and MA, both, using the sample() function. The sample() function in R allows us to pick the mentioned number of random samples from given populations as shown below.

```
#4) Taking a random sample of 30 from NY, 3 bedroom listings
nysample <- sample(cleannylistings,30)
nysample

#5) Taking a random sample of 30 from MA, 3 bedroom listings
masample <- sample(cleanmalistings,30)
masample
```

Formulation of Hypothesis

The first step in hypothesis testing is to formulate Null and Alternative Hypothesis for our given claim. Formulation of Hypothesis is one of the most essential parts of Hypothesis testing, defined as:

H_0: Null Hypothesis – It is a Hypothesis/claim that there is no (Null) difference/no relationship between two variables

H_a: Alternative Hypothesis – It is a claim that states that there is some relationship/distinct difference between two variables

We test the above formulated hypothesis at a given level of significance or using some level of confidence.

A **confidence interval** is defined as a range within which an unknown population parameter lies, given some level of confidence, say, 90%. 90%, 95% and 98% are some of the most commonly used confidence intervals while testing for a hypothesis.

Level of significance defines our area of rejection of a given Null Hypothesis. It is most commonly denoted by α (alpha) which is the probability of rejecting a null hypothesis, given that it is true.

We use the criteria of a **P-value** for our decision rule associated with our result. P-value is defined as a conditional probability based on the assumption that our Null Hypothesis (H_0) is true. A P-value, therefore, signifies the measure of strength of evidence against our Null Hypothesis. A p-value always lies between 0 and 1 **($0< p <1$)** as it is a probability.

Decision rule:

- If P-Value < 0.05, we reject H_0 or Null Hypothesis
- If P-Value > 0.05, we accept H_0 or Null Hypothesis

Testing Claims

1.1 Comparing Mean prices of 3-bedroom Airbnb listings in NY and MA

First, we set up the hypothesis stated as follows:

H_0: $\mu_1 - \mu_2 = 0$ or $\mu_1 = \mu_2$

H_a: $\mu_1 - \mu_2 \neq 0$ or $\mu_1 \neq \mu_2$

where, μ_1 = Mean price of a 3-bedroom Airbnb listing in NY

μ_2 = Mean price of a 3-bedroom Airbnb listing in MA

H_0: There is no significant difference between the mean prices of 3-bedroom listings in NY and MA

H_a: There is a significant difference between the mean prices of 3-bedroom listings in NY and MA

Testing at 5% (0.05) significance level (95% Confidence Interval)

We use t.test() function in R to run a t-test for our two samples as shown below:

```
#6) Testing the difference between Mean prices of 3 bedrooms in NY and MA
t.test(nysample, masample, alternative = 'two.sided')
```

nysample and msample are our sample means drawn from 3-bedroom Airbnb listings in NY and MA, respectively. We have set the alternative option equal to 'two.sided' as we are performing a two-sample test. We are testing our hypothesis at confidence level which is set to 95% by default in R. After executing the above code, we get the following output:

```
        Welch Two Sample t-test

data:  nysample and masample
t = -0.20361, df = 53.48, p-value = 0.8394
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -96.55353  78.75353
sample estimates:
mean of x mean of y
    302.5     311.4
```

Analysis/Interpretation:

As our p-value is 0.8394 > 0.05, we accept our Null Hypothesis H_0 that there is no significant difference between the mean prices of 3 bedrooms in NY and MA.

Therefore, we can be 95% confident that there exists no difference between the mean overnight prices of 3-bedrooms in NY and MA.

1.2 Comparing variance in prices of 3 bedrooms Airbnb listings of NY and MA

For comparing two sample variances, we use an f-test as it assumes that a population from which samples are drawn is normally distributed and samples are independent from one another. We set up our hypothesis as:

Hypothesis formulation for a two-sample f-test:

H_0: $\sigma_1^2 = \sigma_2^2$ 0 or $\sigma_1^2 / \sigma_2^2 = 1$

H_a: $\sigma_1^2 \neq \sigma_2^2$ 0 or $\sigma_1^2 / \sigma_2^2 \neq 1$

where, σ_1^2 = variance in price of a 3-bedroom Airbnb listing in NY

σ_2^2 = variance of a 3-bedroom Airbnb listing in MA

H_0: There is no significant difference between the variation in prices of 3-bedroom listings in NY and MA or the ratio of the variances in prices of NY and MA is 1

H_a: There is a significant difference between the variation in prices of 3-bedroom listings in NY and MA or the ratio of the variances in prices of NY and MA is not equal to 1

We have used the var.test() function in R to run an f-test for the two sample variances as shown below:

```
#7) Testing the difference in variances in 3 bedroom prices in NY and MA
var.test(nysample, masample, alternative = "two.sided")
```

Output:

```
        F test to compare two variances

data:  nysample and masample
F = 0.77065, num df = 29, denom df = 29, p-value = 0.4874
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3668018 1.6191303
sample estimates:
ratio of variances
          0.770649
```

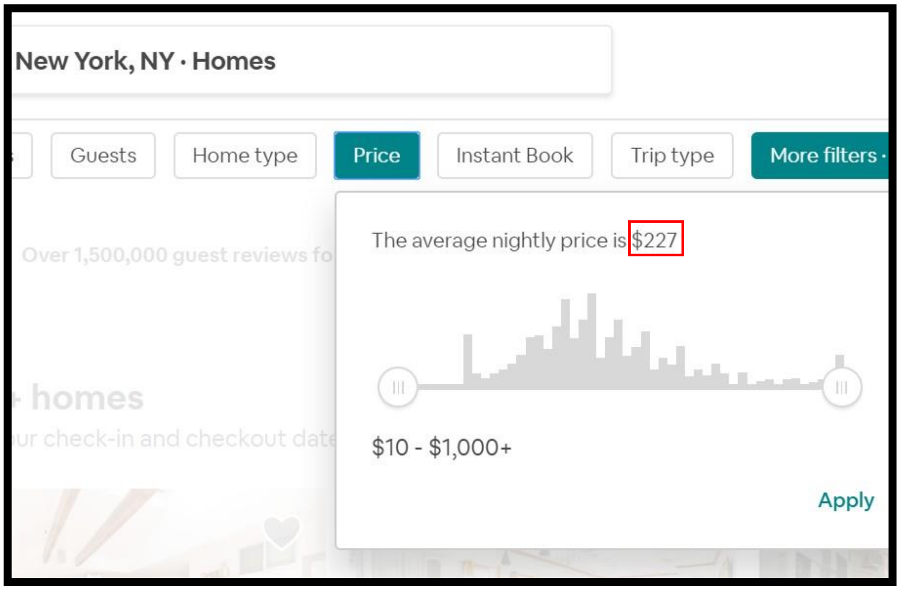
Analysis/Interpretation:

As the p-value 0.4874 > 0.05, we accept our Null Hypothesis that there is no significant difference between the variances of prices of 3 bedrooms in NY and MA.

Therefore, we can be 95% confident that there exists no difference between the sample variance in price of a 3 -bedroom listing in NY and MA.

1.3 Average price of 3-bedroom Airbnb listing is $227* in NY

According to Airbnb's own website, the average overnight price of a 3-bedroom listing in NY is $227. The claim is shown in the screenshot below:



***Note: The price varies, accordingly, from time to time based on the demand and supply of Airbnb accommodations**

Using hypothesis testing, we can check this claim made by Airbnb on its official website. In this case, we re required to test our population mean against the given average price claim of $227 in NY for a 3-bedroom listing. We use the t.test() function in R as shown below:

H_0: $\mu_1 = 227$

H_a: $\mu_1 \neq 227$

Where, μ_1 = mean price of a 3-bedroom Airbnb listing in NY

H_0: The mean price of a 3-bedroom Airbnb listing in NY is not significantly different from population mean $227

H_a: The mean price of a 3-bedroom Airbnb listing in NY is significantly different from population mean $227

```
#8) Testing claim mean price in NY for 3 bedrooms = 227
t.test(nysample, alternative = 'two.sided', mu=227)
```

In the t.test() function above, we have set our mu = 227 as we are testing the claim as made on the Airbnb website.

Output:

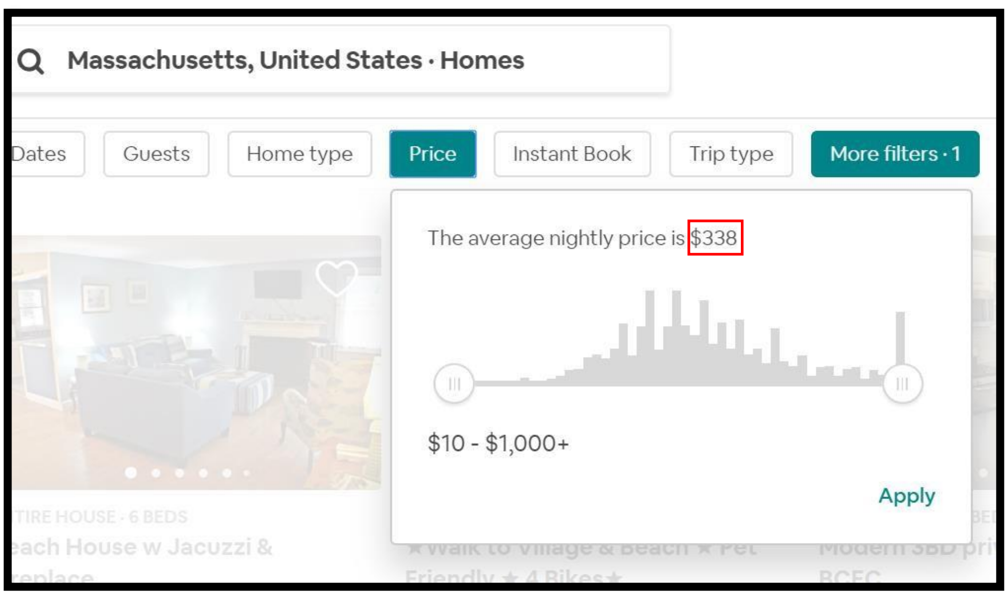
```
        One Sample t-test

data:  nysample
t = 1.777, df = 29, p-value = 0.08606
alternative hypothesis: true mean is not equal to 227
95 percent confidence interval:
 220.1221 325.0113
sample estimates:
mean of x
 272.5667
```

As the p-value 0.08606 > 0.05, we accept our Null Hypothesis that the mean price of an Airbnb listing in NY is not significantly different from $227.

Therefore, we can state with 95% confidence that the mean price of a 3-bedroom listing in NY is not significantly different from $227 i.e. the claim stated by Airbnb on their website is true.

1.4 Average price of a 3-bedroom Airbnb listing is $338* in MA

Similarly, the Airbnb website states that the average overnight price of a 3-bedroom listing in Massachusetts is $338.



***Note: The price varies, accordingly, from time to time based on the demand and supply of Airbnb accommodations**

We perform a two-sample t-test to check this claim, using t.test() function in R, as shown below:

```
#9) Testing claim mean price in MA for 3 bedrooms = 338
t.test(masample, alternative = 'two.sided', mu=338)
```

We have set the value of mu=338 as we are testing the claim given by Airbnb on its website.

Output:

```
        One Sample t-test

data:  masample
t = -1.9445, df = 29, p-value = 0.06159
alternative hypothesis: true mean is not equal to 338
95 percent confidence interval:
 221.459 340.941
sample estimates:
mean of x
    281.2
```

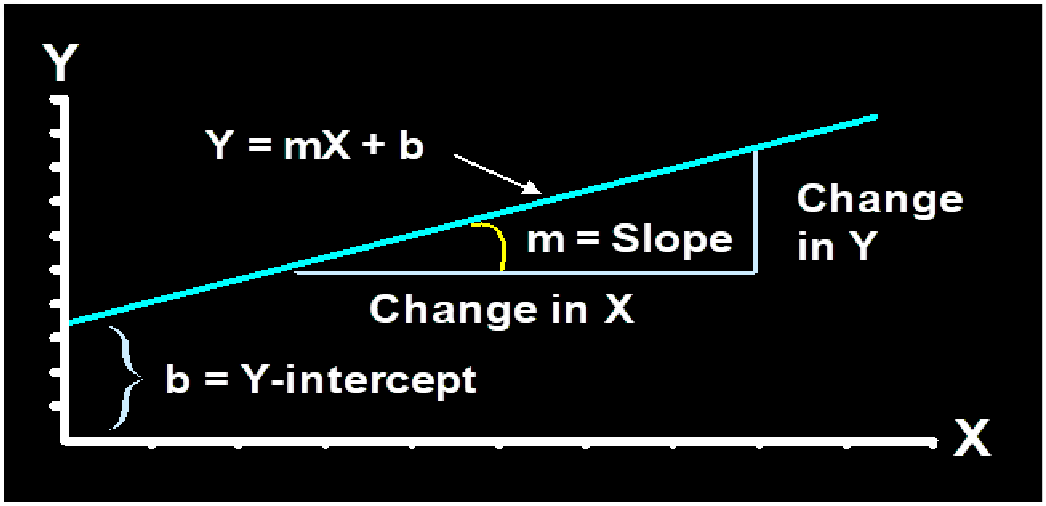
As the p-value 0.06159 > 0.05, we accept the Null Hypothesis that the mean price of a 3-bedroom Airbnb listing in MA is not significantly different from population mean $338.

Therefore, we can say with 95% confidence that the mean price of an Airbnb listing in MA is not significantly different from $338 or the claim made by Airbnb website regarding prices of 3-bedroom listings in Massachusetts is true.

2. Linear Regression

Multiple linear regression is an algorithm which is used to create a model and relationship between the dependent response variable and two or more independent explanatory variable.

It is used to predict the outcome of a dependent response variable taking into consideration the independent variable.



Linear equation – **Y = mX + b**

Where,

m = Slope = change in Y / change in X

b = Y intercept

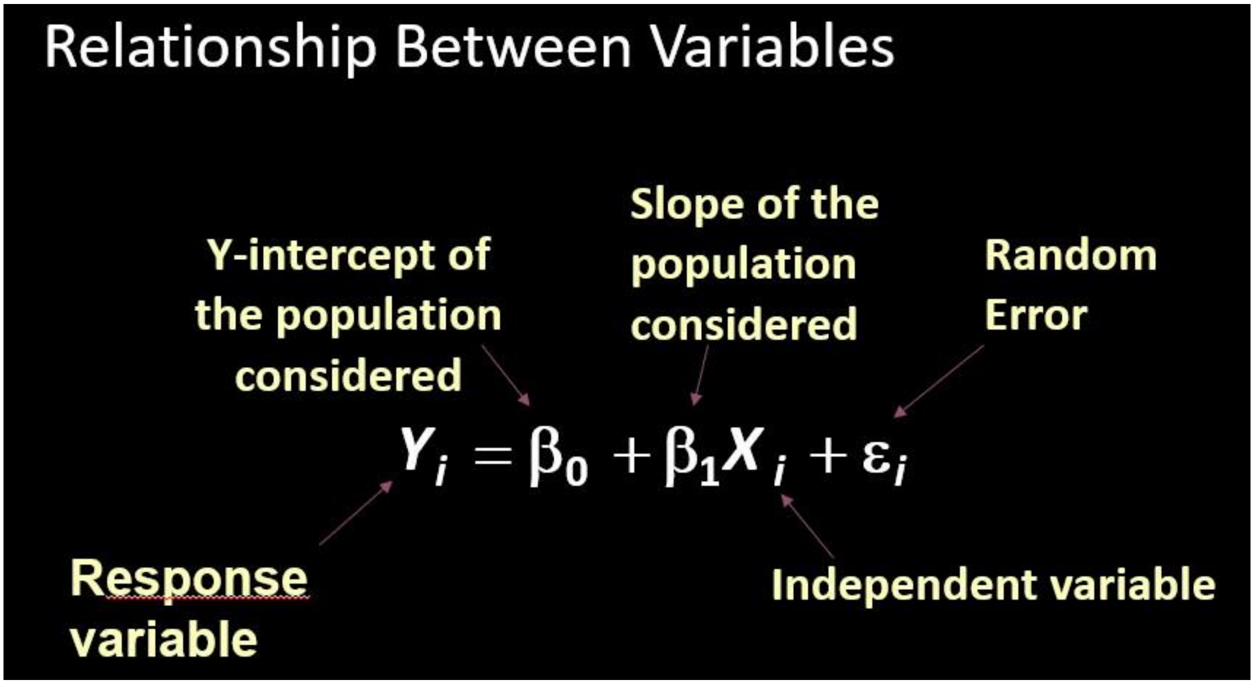
Types of variables

1. **Dependent**

- These variables are used to measure the effect of the independent variable.
- They are completely dependent on the independent variables.
- They are also called as Predicted variables.
- Variable taken – Price

2. **Independent**

- These are variables which are frequently changed, and their effects are then measured and compared.
- They are called as predictors since they are used to predict the values of the dependent variable.
- Variables taken – id, host_id, street, neighbourhood_cleansed, city, state, zipcode, latitude, longitude, property_type, room_type, accommodates, bathrooms, beds, bed_type and review_scores_ratings.



Since we will be dealing with more than one independent variable, we will be using Multiple linear regression.

nymalistings = read.csv(file.choose(), header = TRUE, na.strings = "")

nymalistings

The above command is used to read the data from the local directory. Since this is a real-world data, there are ought to be a lot of missing values, discrepancies and useless columns. Since R is not designed to handle empty values at large, so we will use " **na.strings=""** " to turn empty values into NA.

install.packages("lars")

install.packages("glmnet")

install.packages("ggplot2")

library(lars)

library(glmnet)

library(ggplot2)

These above commands are used to install and use the packages which are necessary for the implementation of linear model functions.

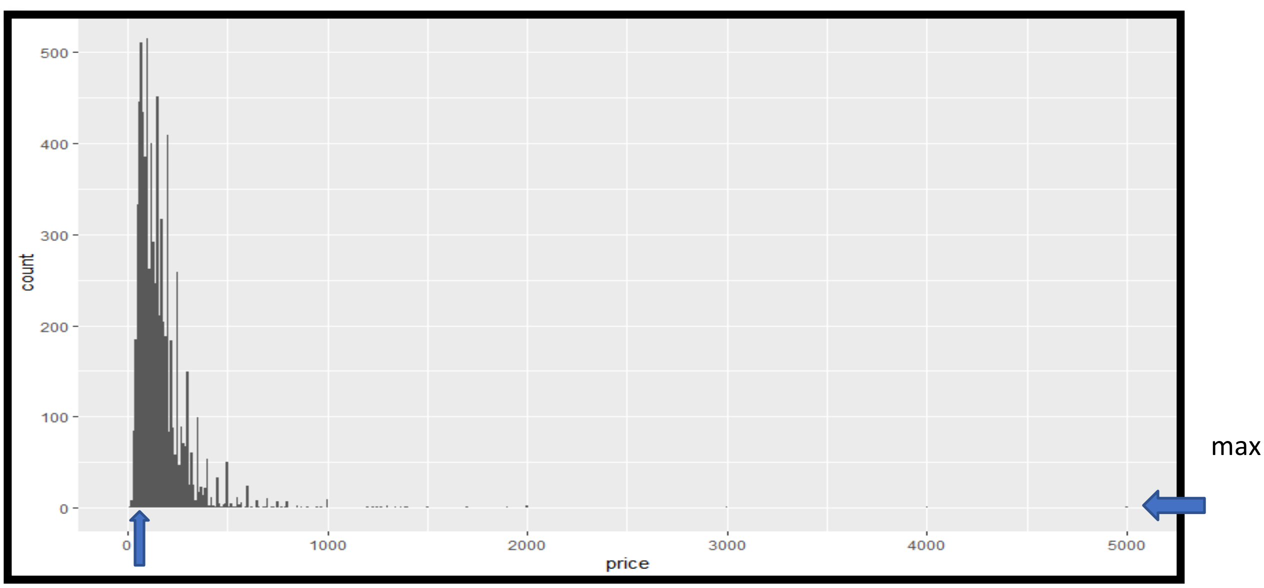
summary(nymalistings)

```
       id              host_id                                                          street
Min.   :    2515   Min.   :     2571   New York, NY, United States                         :1890
1st Qu.:  688008   1st Qu.: 1491738   Brooklyn, NY, United States                         :1784
Median : 1747814   Median : 5087142   Queens, NY, United States                           : 165
Mean   : 4456628   Mean   :13548119   Boylston Street, Boston, MA 02215, United States:  64
3rd Qu.: 8238381   3rd Qu.:17891752   Beacon Street, Boston, MA 02116, United States  :  50
Max.   :14933461   Max.   :93854106   Bronx, NY, United States                            :  45
                                      (Other)                                             :3584
      neighbourhood_cleansed                  city           state        zipcode             country          latitude
Williamsburg      : 427        Boston           :3382   MA:3583   2116    : 388   United States:7582   Min.   :40.51
Jamaica Plain     : 343        New York         :1890   NY:3999   2130    : 331                        1st Qu.:40.72
South End         : 326        Brooklyn         :1784             11211   : 302                        Median :40.82
Back Bay          : 302        Queens           : 165             2118    : 247                        Mean   :41.49
Bedford-Stuyvesant: 290        Bronx            :  45             2215    : 237                        3rd Qu.:42.34
Fenway            : 290        Long Island City:  38             (Other):6010                         Max.   :42.39
(Other)           :5604        (Other)          : 278             NA's    :  67
   longitude          property_type             room_type         accommodates       bathrooms         bedrooms
Min.   :-74.24   Apartment  :5797   Entire home/apt:4566   Min.   : 1.000   Min.   :0.000   Min.   :0.000
1st Qu.:-73.96   House      : 831   Private room   :2894   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:1.000
Median :-73.91   Condominium: 328   Shared room    : 122   Median : 2.000   Median :1.000   Median :1.000
Mean   :-72.60   Loft       : 266                          Mean   : 2.975   Mean   :1.167   Mean   :1.236
3rd Qu.:-71.08   Townhouse  : 181                          3rd Qu.: 4.000   3rd Qu.:1.000   3rd Qu.:1.000
Max.   :-71.00   (Other)    : 176                          Max.   :16.000   Max.   :6.000   Max.   :9.000
                 NA's       :   3                                           NA's   :81      NA's   :48
      beds              bed_type          price            currency       review_scores_rating
Min.   : 0.000   Airbed       :  59   Min.   :  10.0   Dollars:7582   Min.   : 20.00
1st Qu.: 1.000   Couch        :  19   1st Qu.:  85.0                  1st Qu.: 90.00
Median : 1.000   Futon        : 123   Median : 135.0                  Median : 95.00
Mean   : 1.595   Pull-out Sofa:  82   Mean   : 165.6                  Mean   : 92.93
3rd Qu.: 2.000   Real Bed     :7299   3rd Qu.: 200.0                  3rd Qu.: 98.00
Max.   :16.000                        Max.   :5000.0                  Max.   :100.00
NA's   :19                                                            NA's   :1092
```

Summary function is used to get result summaries of various model fitting functions. It tells us about each variable. If it is a numeric variable, it tells about the minimum and maximum value along with the quartile values and if it is a categorical variable, it tells about the number of levels in the variable and the count of the levels.

From the results, we can see that the minimum value of the price variable is 10 whereas maximum being 5000. First quartile value is 85, median is 135, mean is 165 and 3rd being 200. (prices are calculated in dollars)

ggplot(nymalistings, aes(x=price)) + geom_histogram(binwidth=10) + labs(x="price")



min

(l <- sapply(nymalistings, function(x) is.factor(x)))

```
               id           host_id            street neighbourhood_cleansed                city
            FALSE             FALSE              TRUE                   TRUE                TRUE
            state           zipcode           country               latitude           longitude
             TRUE              TRUE              TRUE                  FALSE               FALSE
    property_type         room_type      accommodates              bathrooms            bedrooms
             TRUE              TRUE             FALSE                  FALSE               FALSE
             beds          bed_type             price               currency review_scores_rating
            FALSE              TRUE             FALSE                   TRUE               FALSE
```

Sapply function is used to calculate the type of the variables. Here, we have analysed the variables which are factor using **is.factor()** function. Output as TRUE tells us that they are functions and FALSE says otherwise.

m <- data.frame(nymalistings [, l])

The above command is used to create or convert the dataset into a data frame which makes it easy to work on. This variable m has only the factors and not int variables.

ifelse(n <- sapply(m, function(x) length(levels(x))) == 1, "DROP", "NODROP")

```
> ifelse(n <- sapply(m, function(x) length(levels(x))) == 1, "DROP", "NODROP")
                street neighbourhood_cleansed                   city                  state                zipcode
              "NODROP"               "NODROP"               "NODROP"               "NODROP"               "NODROP"
               country          property_type              room_type               bed_type               currency
                "DROP"               "NODROP"               "NODROP"               "NODROP"                 "DROP"
```

Since we cannot consider the predictors/factors which takes only a single value throughout, which is, the factor which has only one level because it won't affect the data or a dependent variable.

So, we run this command to check the number of levels the factor has and if it does, it will ask us to drop it.

which(sapply(m, function(x) length(unique(x))<2))

```
> which(sapply(m, function(x) length(unique(x))<2))
 country currency 
       6       10 
```

The above command is used to check for the variables/factors which has length/level less than 2 and return their index/column number.

As we can see from the results that the factors currency and country turned out to be the ones which has length less than 2, ie. Which has the same values throughout the data.

The index of

Country – 6

Currency – 10

This means we will not be considering these independent variables in our linear model.

linearMod <- lm(price ~ id + host_id + street + neighbourhood_cleansed + city + state + zipcode + latitude + longitude + property_type + room_type + accommodates + bathrooms + beds +bed_type + review_scores_rating, data=mydata)

linearMod

summary(linearMod)

Linear model function is used to present a continuous response variable as a function of one or more than one predictor variables. They are used to analyze and predict the behavior of the system depending on the data.

To implement linear model, we use **lm(response variable ~ predictor variables + …)** function.

```
> summary(linearMod)

Call:
lm(formula = price ~ id + host_id + street + neighbourhood_cleansed +
    city + state + zipcode + latitude + longitude + property_type +
    room_type + accommodates + bathrooms + beds + bed_type +
    review_scores_rating, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max
-298.1  -30.9    0.0   20.0 4623.7

Coefficients: (156 not defined because of singularities)
                                                                      Estimate Std. Error t value
(Intercept)                                                         -3.225e+03  3.726e+04  -0.087
id                                                                  -2.091e-06  8.959e-07  -2.333
host_id                                                             -2.063e-08  1.724e-07  -0.120
street13th Street, Boston, MA 02129, United States                  -1.069e+02  1.271e+02  -0.841
street1st Avenue, Boston, MA 02129, United States                   -5.352e+01  1.334e+02  -0.401
```

Here, from the results we can see that the residuals of the price are calculated which is by comparing the calculated data with the original dataset. The minimum value is -298.1, max being the 4623.7 with median being 0. This means that some value in the middle of the list is predicted correctly.

From the values of the coefficients, we can fit them into the equation as

Mean price value = (-3.225e+03) +(-2.091e-06)(id)+(-2.063e-08)(host_id)+…so on

```
                                                    Pr(>|t|)
(Intercept)                                         0.931022
id                                                  0.019665 *
host_id                                             0.904773
street13th Street, Boston, MA 02129, United States  0.400395
street1st Avenue, Boston, MA 02129, United States   0.688362
```

We can observe that there is an asterisk besides id which means that it is significant, and rest are not since their p-values are too high. (p-value > alpha)

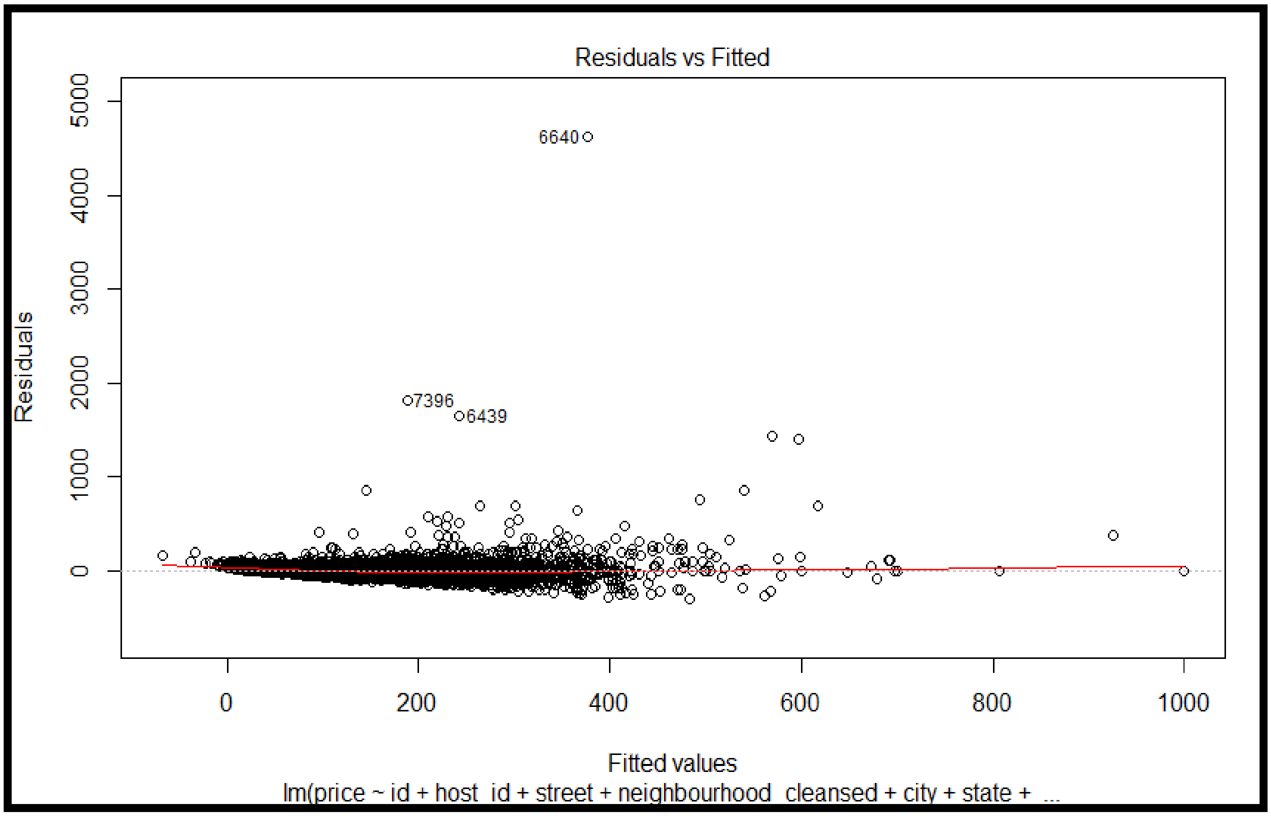
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 108.9 on 5000 degrees of freedom
  (1213 observations deleted due to missingness)
Multiple R-squared:  0.4844,	Adjusted R-squared:  0.3433
F-statistic: 3.434 on 1368 and 5000 DF,  p-value: < 2.2e-16
```

R-sq = 0.4844, which means that approx. 48.44% of variations can be explained using the independent variables

We have F-statistics and p value for the test of significance.

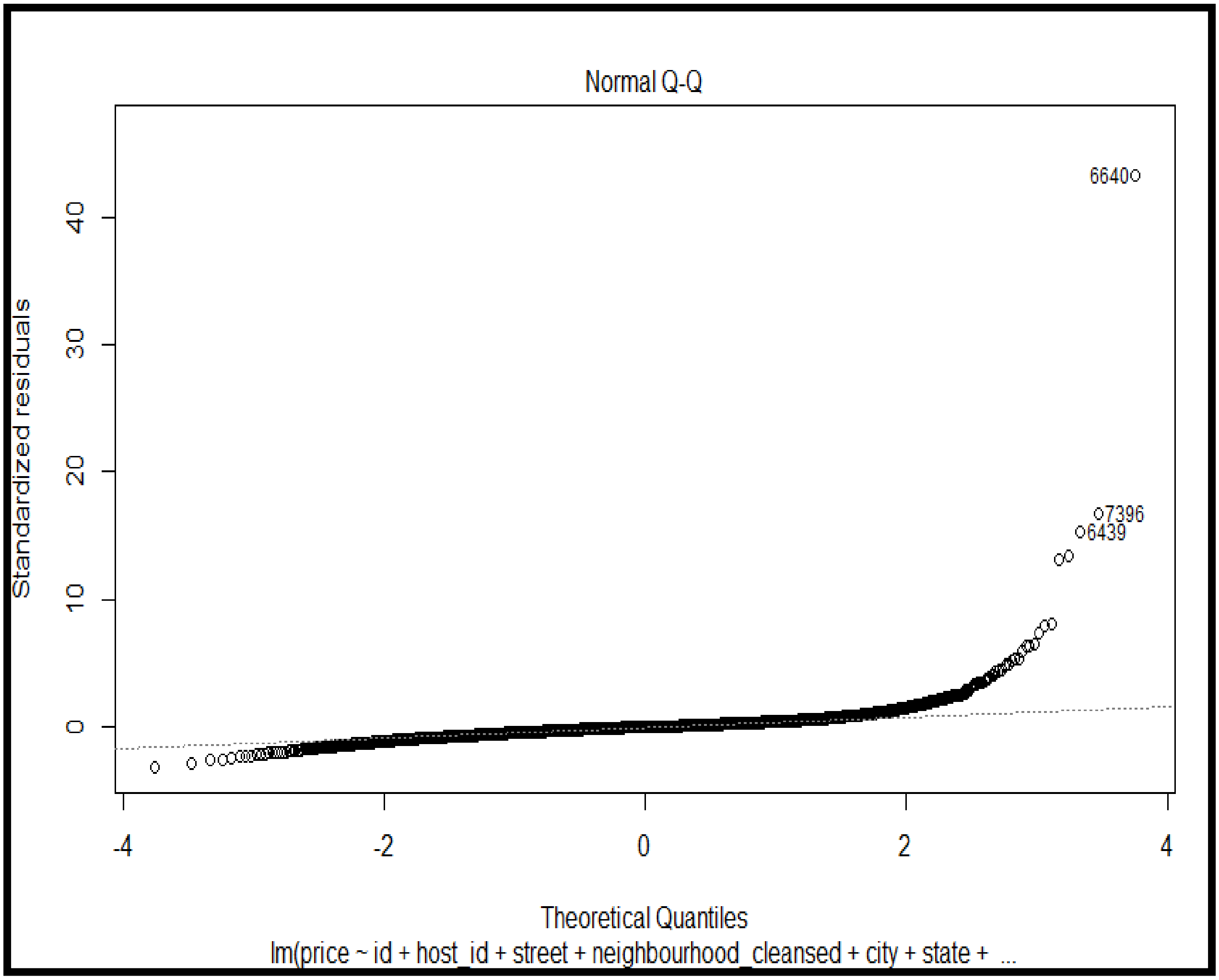
plot(linearMod)

There are 4 important graphs plotted using the function. Please use 'enter' to generate the graphs one by one in R-studio.



Residuals vs Fitted is a scatter plot of residuals on the y axis and fitted or estimated values being on the x axis. This plot is highly considered while accounting for any errors, outliers, non-linearity and unequal error variance. From the graph we can see that there are a few outliers which way outside the normal deviation. Also, we can see the red line is horizontal to the points plotted and the origin mark, this means that there is no non-linearity in the model. Thus, this model is good.

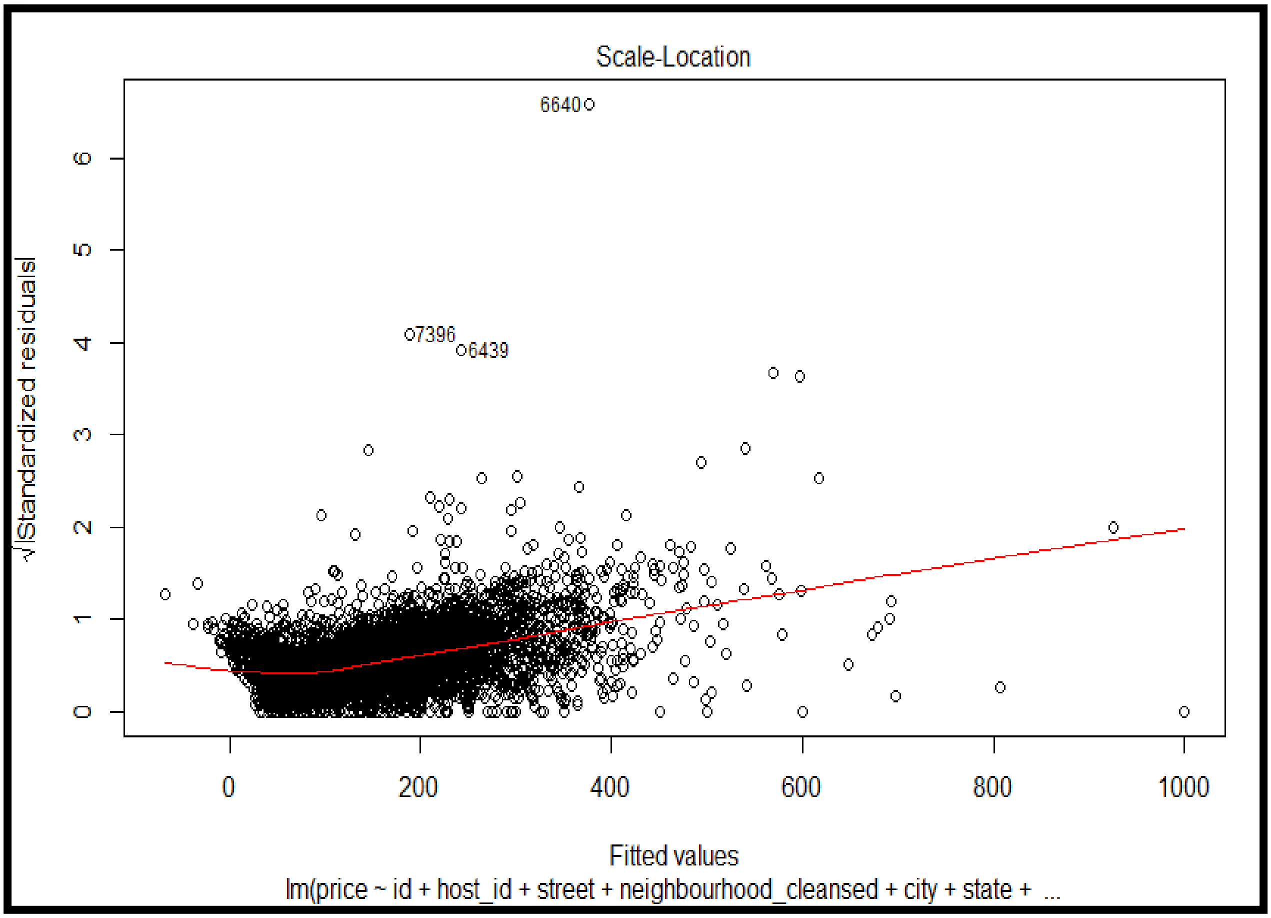
The above results show that our assumptions and consideration of the independent variables as the predictor variables might have some affect on the response variable. The relationship showing linearity is a good sign for the model to be in good position to test the outcomes of other variables by changing the dataset. This way we can see the difference between the results when more data is added to the database. These outliers can further be removed or neglected while working on the model for better results.



The normal Q-Q plot is used to show whether the residuals are normally distributed or not. While determining the reliability of the graph, we check whether the residuals are following the linear line or not. If it does, the model is good and not concerning. The normal Q-Q plot has standardized residuals on the y-axis and Theoretical quantiles on the x-axis.

In this case, we can notice that the residuals start off well but, in the end, it started to deviate, and the deviation is way off to be considered normal. This means that there might be issues with the residual model. This indicates that there are some independent variables which are influencing the response variable in producing such large deviations of residuals.

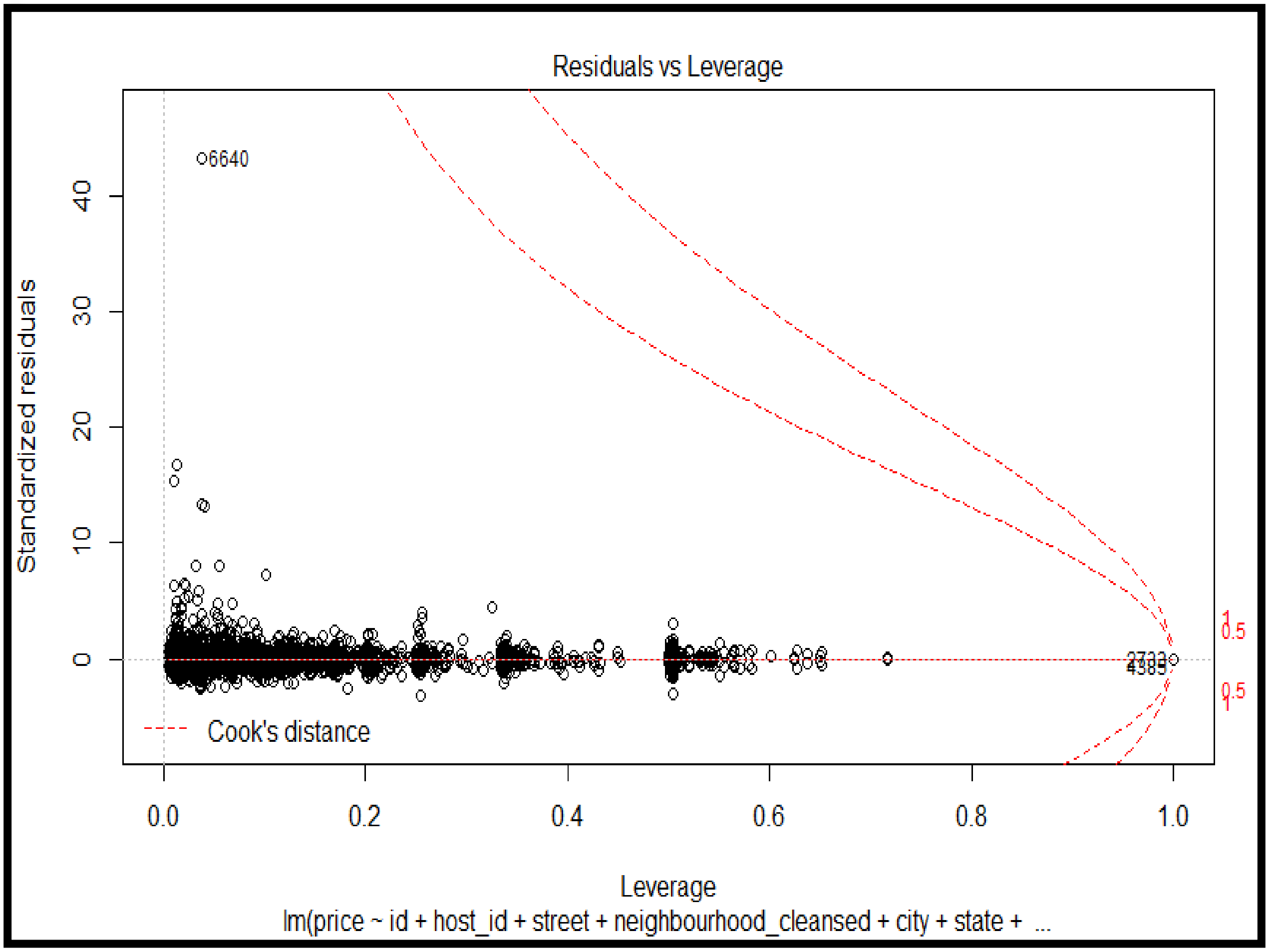
The values 6439, 7396 and 6640 might be concerning and not good for the model.



This graph is also known as Spread- Location plot. This graph is highly used to note whether the residuals are equally spread along the ranges of predictors or not. The graph has square root of mod of standardized residuals on the y-axis and the fitted values on the x-axis.

As we can see that the red line is the decider in this case. And we can notice that the spreading of the residuals is quite good on both the sides, though a few outliers being an exceptional case. This means that our assumption of equal variance is correct, and the model supports the assumption which might be helpful in further analysis.

Though, the outliers are very less and stay an exceptional case, we should not neglect them completely. Here, 6439, 7396 and 6640 stands out to be outliers and will be taken into consideration while making the further decisions.



This graph is best used to find out if there are any influential cases present in the model or not.

The graph has standardized residuals on the y-axis and the leverage on the x-axis.

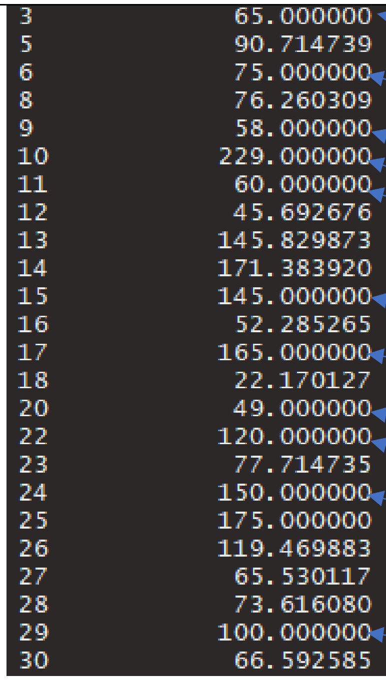
In this graph, we will talk about the importance or residuals and the cases when to consider and when we can neglect them. Not all outliers are influential in regression analysis. Even though the values are too high, we might not have problems because of them. This is determined by the cook's distance or cook's score. If the outliers are well within the cook's distance, which is the red line, then they are not influential as such and can be considered or not depending on the situation but if they are outside the line, then we cannot neglect them as they can have massive impact on the results.

In our case, all the outliers and the points are well within the cook's distance and a few are marginally touching the contact point. That means that they might have an effect on the results but not much.

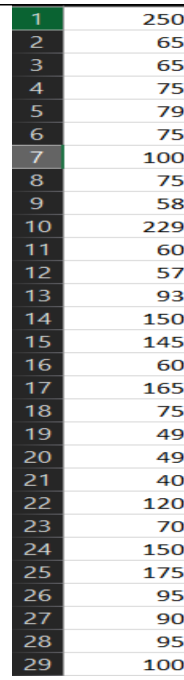
predict_price <- predict(linearMod)

predict_price

PREDICTED VALUES	ORIGINAL VALUES



3	65.000000
5	90.714739
6	75.000000
8	76.260309
9	58.000000
10	229.000000
11	60.000000
12	45.692676
13	145.829873
14	171.383920
15	145.000000
16	52.285265
17	165.000000
18	22.170127
20	49.000000
22	120.000000
23	77.714735
24	150.000000
25	175.000000
26	119.469883
27	65.530117
28	73.616080
29	100.000000
30	66.592585



1	250
2	65
3	65
4	75
5	79
6	75
7	100
8	75
9	58
10	229
11	60
12	57
13	93
14	150
15	145
16	60
17	165
18	75
19	49
20	49
21	40
22	120
23	70
24	150
25	175
26	95
27	90
28	95
29	100

Predict() function is used to predict the values of the response variable from the linear model depending on the independent or the predictor variables. We can see that there are a lot of values which have been predicted correctly but some are a bit off. This might be because of there might be some useless independent variables or predictor variables which we have considered in the model might not be useful for the prediction and might be hindering the values.

To make our model more accurate, we use the new and updated values as the training dataset and use the same procedure to predict the new values of the price which will be more accurate than before.

residual_price <- data.frame(resid(linearMod))

format(residual_price, scientific=FALSE)

Resid() function is used to calculate the difference between the predicted value and the original value from the dataset.

This difference will help us understand where and which values might be having a greater effect due to the predictor variables.

3	-0.0000000000000704991621
5	-11.7147390332091028142258 4
6	0.0000000000001859623566
8	-1.2603094617377907926680 8
9	0.0000000000002353672812
10	0.0000000000040284442449
11	-0.0000000000026917357232
12	11.3073241599613041330485 4
13	-52.8298731387572999551594 0
14	-21.3839202211001655484778 9
15	-0.0000000000031069591344
16	7.7147346701188066475651 8
17	0.0000000000011146639167
18	52.8298731387562483519104 7
20	-0.0000000000001981748099
22	0.0000000000010674794382
23	-7.7147346701188341810961 9
24	-0.0000000000002231548279
25	0.0000000000000771605002
26	-24.4698831029562064998117 4
27	24.4698831029565475603249 1
28	21.3839202211003538423028 7
29	0.0000000000015415446697
30	0.4074148732481734369592 6
31	-5.8659417790428420858006 6
32	5.8659417790436112483121 2

As we can see that there are a lot of values which are 0, this indicates that the difference between the predicted value and the original value are the same. There are a few cases where the difference is very less which is a good thing but there are cases where difference is quite big which might indicate that there are some independent variables which are useless and hindering the prediction without having an actual dependency.

The problem with linear model is that it considers every variable to create a model irrespective of the actual dependency. This is the reason we do LASSO regression after linear regression. LASSO removes the useless variables and take into consideration only the ones which are having an impact on the dependent variable.

3. Regularization Techniques

The purpose of machine learning algorithm is to train an algorithm with training data which makes the prediction of test data accurate. A machine learning algorithm can fail either because of under fitting or overfitting. Underfitting rises if the hypothesis space if restricted to capture important variables in model and overfitting occurs when hypothesis space is so big that it captures the noise in the dataset. Overfitting in a model is quite common and one of rising concerns in machine learning across models and algorithms. Reasons for overfitting: Less variables in dataset, noise in the dataset & hypothesis space is large. Avoiding the problem of overfitting can improve model's performance.

Regularization is a technique in which more information is added to an algorithm in such a way that overfitting is resolved. It helps to improve the prediction accuracy by penalizing the variables which make the model complex and by minimizing the error between predicted and actual observations. Lambda (λ) is regularization parameter. It penalizes all parameters except intercept to prevent overfitting.

Some of the techniques used to create a parsimonious model when there are large number of variables in a dataset are L1 Regularization and L2 Regularization. They address the issues of overfitting and feature selection.

Regression model used for Lasso regression: L1

Regression model used for Ridge regression: L2

They both work on the same principle. They penalize the coefficient using the penalty terms so that important variables are observed in the model. All variables are retained in case of Ridge regression and few in case of Lasso regression.

The key difference is how the penalty term works. In Ridge regression, squared magnitude of coefficient acts as penalty term. Lasso adds absolute value of coefficient as penalty term to the function.

This can be explained mathematically as follows:

a. Ridge Regression:

$$\boxed{\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2}$$

When Lambda is zero, the equation is same as OLS (ordinary least squares).

When Lambda is large, it adds significant weight causing underfitting.

Choosing the value of lambda is important.

b. Lasso Regression:

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

When lambda is zero, OLS is obtained.

When lambda is large, coefficients will become zero causing underfitting of the model.

The question arises: What is the difference between the two techniques?

The answer is the key difference in working between the two techniques. Lasso shrinks the coefficients of not so important variables to zero. This removes those variables, thus features to zero. This helps in feature selection when the number of features is quite high.

Before Lasso, stepwise regression was used. It helped to improve the accuracy of prediction only. In some cases, it made it worse.

Ridge regression was a popular technique to improve prediction accuracy by shrinking the large coefficients in order to reduce overfitting. It doesn't help in feature selection and thus no help in making the model easily interpretable.

Reasons to consider LASSO REGRESSION:

a. Prediction Accuracy:

The OLS estimate often has low bias but large variance. Accuracy of prediction can be improved by shrinking the value of coefficients. This way bias is introduced but variance of predicted values is reduced. This way, overall prediction accuracy improves.

b. Purpose of Interpretation:

When number of variables is large, the intention is to find a small set of variables with strongest effects.

For the purpose of analysis of AIRBNB data of Massachusetts and New York, installed the following two important libraries in RStudio:

- Glmnet
  A package that fits a generalized linear model via penalized maximum likelihood. It can be downloaded from CRAN directly using the code: install.packages("glmnet")
  Load the glmnet package using: library(glmnet).

```
nymalistings <- read.csv("C:/Users/unnaty/Downloads/data.csv")
install.packages("glmnet")
library(glmnet)
```

- Lars

  A package used for efficient procedures for fitting entire lasso sequence with cost of single least squares fit. Least angle regression and infinitesimal forward stage wise regression are related to the lasso. It can be downloaded from CRAN directly using the code: install.packages("lars")

  Load the lars package using: library(lars).

```
nymalistings <- read.csv("C:/Users/unnaty/Downloads/data.csv")
install.packages("glmnet")
library(glmnet)
install.packages("lars")
library(lars)
```

Consider the data in data frame named as DF.

```
nymalistings <- read.csv("C:/Users/unnaty/Downloads/data.csv")
install.packages("glmnet")
library(glmnet)
install.packages("lars")
library(lars)
DF <-data.frame(nymalistings$id, nymalistings$host_id, nymalistings$street, nymalistings$neighbourhood_cleansed, nymalistings$city,

DF <- na.omit(DF)
```

Omit the null values from the data frame to make the analysis efficient.

```
DF <- na.omit(DF)
```

```
> DF <-data.frame(nymalistings$id, nymalistings$host_id, nymalistings$street, nymalistings$neighbourhood_cleansed, nymalistings$city, nym
alistings$state, nymalistings$zipcode, nymalistings$latitude, nymalistings$longitude, nymalistings$property_type, nymalistings$room_type,
 nymalistings$accommodates,nymalistings$bathrooms, nymalistings$bedrooms,nymalistings$beds, nymalistings$bed_type, nymalistings$review_sc
ores_rating, nymalistings$price)
> DF <- na.omit(DF)
```

Creating Matrix is a crucial step towards Lasso Regression.

```
x<- data.frame(DF$nymalistings.id, DF$nymalistings.host_id, DF$nymalistings.street, DF$nymalistings.neighbourhood_cleansed, DF$nyma
```

```
> x<- data.frame(DF$nymalistings.id, DF$nymalistings.host_id, DF$nymalistings.street, DF$nymalistings.neighbourhood_cleansed, DF$nymalist
ings.city, DF$nymalistings.state, DF$nymalistings.zipcode, DF$nymalistings.latitude, DF$nymalistings.longitude, DF$nymalistings.property_
type, DF$nymalistings.room_type, DF$nymalistings.accommodates, DF$nymalistings.bathrooms, DF$nymalistings.bedrooms, DF$nymalistings.beds,
 DF$nymalistings.review_scores_rating)
> 
```

The matrix has the following 6379 observations.

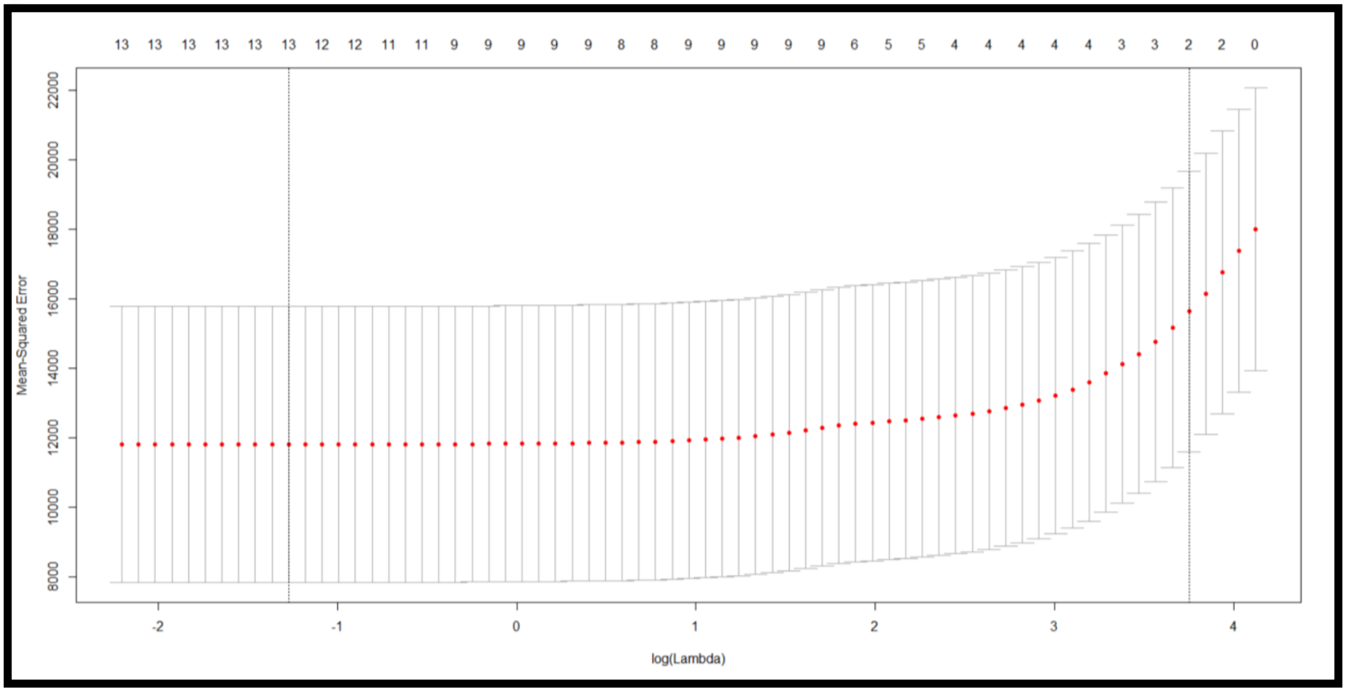
	DF.nymalistings.id	DF.nymalistings.host_id	DF.nymalistings.street	DF.nymalistings.neighbourhood_cleansed	DF.nymalistings.city	DF.nymalistings.state	DF.nymalistings.zipcode	DF.nymalistings.latitude	DF.nymalistings.longitude	DF.nymalistings.prope
1	12147973	31303940	113	124	7	1	163	42.28262	-71.13307	
2	3075044	2572247	924	124	7	1	163	42.28624	-71.13437	
3	6976	16701	45	124	7	1	163	42.29244	-71.13577	
4	1436513	6031442	139	124	7	1	1	42.28111	-71.12102	
5	7651065	15396970	379	124	7	1	163	42.28451	-71.13626	
6	12386020	64200298	1169	124	7	1	163	42.29169	-71.13189	
7	5706985	6570877	674	124	7	1	163	42.28139	-71.13119	
8	2843445	3164508	319	124	7	1	163	42.28195	-71.14102	
9	753446	3962517	1097	124	54	1	163	42.28588	-71.12491	

Setting up the model as follows:

```
#Setting up the model
lasso_model <- cv.glmnet(Mx , y, alpha = 1)
lasso_model1 <- glmnet(Mx,y)
```

Plotting the cross-validation error according to log(lambda) values:

```
#Plot of cross valildation error according to log lambda values
plot.cv.glmnet(lasso_model)
```



The plot shows the cross-validation error according to log(lambda) values.

The left most dashed vertical line indicates the log of optimal value of lambda, i.e., -1 approximately. It minimizes the prediction error. This value of lambda gives the most accurate model. Minimum error is observed at this point.

Generally, the purpose of regularization is to balance accuracy and simplicity. This means a model with the smallest number of predictors that also give accuracy of model.

The function cv.glmnet (lasso_model), finds value of lambda that gives simplest of model but also lies within one standard error of optimal value of lambda. This value is lambda.1se (largest value of lambda such that error is within one standard error of the minimum.)

1. When lambda.min is used as the best value of lambda, then the following results are obtained:

```
#When lambda is minimum
fit <- glmnet(x=Mx, y=y, alpha = 1, lambda= lasso_model$lambda.min)
fit$beta
```

```
> fit <- glmnet(x=Mx, y=y, alpha = 1, lambda= lasso_model$lambda.min)
> fit$beta
17 x 1 sparse Matrix of class "dgCMatrix"
                                        s0
DF.data.id                   -1.229755e-06
DF.data.host_id               .
DF.data.street               -3.223656e-03
DF.data.neighbourhood_cleansed  1.058787e-01
DF.data.city                 -9.879065e-02
DF.data.state                 .
DF.data.zipcode              -6.467644e-01
DF.data.latitude              4.936165e+01
DF.data.longitude             .
DF.data.property_type         .
DF.data.room_type            -7.578142e+01
DF.data.accommodates          1.135662e+01
DF.data.bathrooms             4.689165e+01
DF.data.bedrooms              3.230800e+01
DF.data.beds                 -9.898345e-01
DF.data.bed_type              2.441050e-02
DF.data.review_scores_rating  8.673622e-01
>
```

It is observed that when lambda.min is used, then four variables, namely, host_id, state, longitude and property_type have shrunk to zero.

2. When lambda.1se (max lambda value) is used as best lambda, then following results are obtained:

```
#When lambda is maximum
fit1 <- glmnet(x=Mx, y=y, alpha = 1, lambda= lasso_model$lambda.1se)
fit1$beta
```

```
17 x 1 sparse Matrix of class "dgCMatrix"
                                       s0
DF.data.id                             .
DF.data.host_id                        .
DF.data.street                         .
DF.data.neighbourhood_cleansed         .
DF.data.city                           .
DF.data.state                          .
DF.data.zipcode                        .
DF.data.latitude                       .
DF.data.longitude                      .
DF.data.property_type                  .
DF.data.room_type              -19.957642
DF.data.accommodates             9.236519
DF.data.bathrooms                      .
DF.data.bedrooms                 2.402951
DF.data.beds                           .
DF.data.bed_type                       .
DF.data.review_scores_rating           .
```

Here, fourteen (14) variables have shrunk to zero. Thus, only three variables, namely, room_type, accommodates and bedrooms have non-zero coefficients.

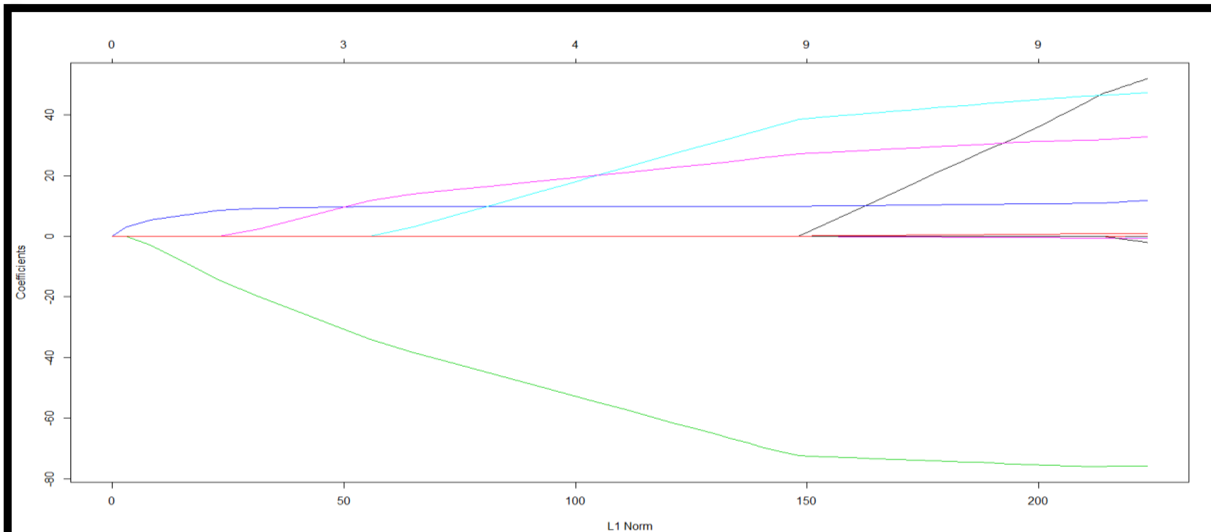
The coefficients of all other variables have been set to zero by the lasso algorithm, thereby reducing the complexity of the model.

It can be concluded that by setting lambda= lambda.1se, a simpler model is obtained compared to lambda.min but model from lambda.1se might be a little less accurate than the one obtained with lambda.min.

Thus, the most effective model with accuracy in prediction and easy interpretability would be with three variables: room_type, accommodates and bedrooms.

The following plot shows how each curve responds to change in lambda. Each curve is representative of a variable. The path shows coefficients against L1 norm as lambda varies. In a way it is the evolution of the value of the different coefficients while lambda is growing.

```
#Plot showing path of coefficient of variables against L1-norm as lambda varies
plot.glmnet(lasso_model1)
```



4. Data Mining

Data mining refers to extracting useful information from vast amounts of data. Many other terms are being used to interpret data mining, such as knowledge mining from databases, knowledge extraction, data analysis, and data archaeology. It is also called as knowledge discovery process, knowledge mining from data, knowledge extraction or data /pattern analysis.

The goal of this data mining is to find patterns that were previously unknown. Once these patterns are found they can further be used to make certain decisions for development of their businesses. Three steps involved are:

- Exploration: In this step, data is cleaned and transformed into another form, and important variables and problem statement is defined.
- Pattern Identification: Once data is explored and defined for the specific variables, the second step is to form pattern identification. Identify and choose the patterns which make the best prediction.
- Deployment: Here, in this step, we deploy the predicted model.

In our project, we will be using two techniques of data mining, namely classification and clustering on Airbnb dataset.

4.1 Classification:

Classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. Fraud detection and credit-risk applications are particularly well suited to this type of analysis. The classification algorithms used in this project are linear discriminant analysis and K nearest neighbors.

4.1.1 Linear Discriminant Analysis:

LDA models the distribution of predictors separately in each of the response classes, and then it uses Bayes' theorem to estimate the probability. It makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.
In this project, we have used LDA classification on Airbnb dataset for New York and Massachusetts. The predicted or the response variable used here is room type and the predictor used is price. We have done separate classification for New York and Massachusetts and our aim is to find the best classification method based on the data.

Packages and Libraries used for classification are listed below:

```
install.packages("class") #package to perform classification
install.packages("MASS") #package to perform classification
```

```
library(class) #package to perform classification
library(MASS) #package to perform classification
```

First, we start by dividing our data set into two different data frames, one for New York and other for Massachusetts. The same can be achieved using the following code:

```
data_ny <- subset(nymalistings, state == 'NY') #Data for New York
data_ma <- subset(nymalistings, state == 'MA') #Data for Massachusetts
```

Next step is to create two separate data frames for training and testing data. Training data is the subset of our dataset which is used to train our model whereas the testing data is used for testing the model to ensure we do not have overfitting or underfitting. Following code helps us subset our data set for Massachusetts into training and testing data:

```
train_data<- data_ma[1:2000,c(12,18)]
test_data <- data_ma[2001:3583, c(12,18)]
```

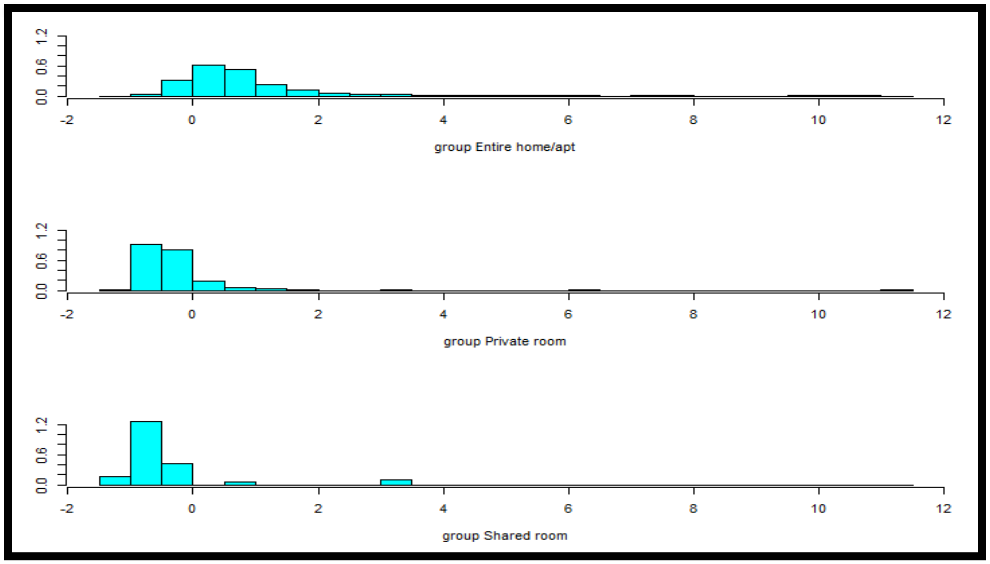
Now that we have our data ready, our next step is to create a fit model for LDA. The qualitative variable used in the fitting model is the room type that defines the type of room i.e. private room, entire apartment and shared room. The quantitative attribute is the price of each room. The following is the code to create a fitting model using R mentioned along with the screenshots of output.

```
m1 <- lda(room_type ~ price, data = train_data)
plot(m1)
```

Output:

```
> m1
Call:
lda(room_type ~ price, data = train_data)

Prior probabilities of groups:
Entire home/apt    Private room     Shared room
         0.6565          0.3245          0.0190

Group means:
                    price
Entire home/apt 227.41432
Private room    102.21880
Shared room      88.52632

Coefficients of linear discriminants:
              LD1
price 0.008843019
```

The above output gives the prior probabilities of each class i.e. 65.65% of our training data has entire home as the room type. Similarly, for private room and shared room, we have 32.45% and 1.9%, respectively. The LDA model also gives us group means that gives us the average of the quantitative variable corresponding to each group. From the above output, we can interpret that for entire home/apt the average price is $227.41. Similarly, for private and shared room the average price is 102.21 and 88.52, respectively. The linear discriminant coefficients give the linear value for price which is further used as a basis for the LDA decision rule. It can also be described as the multipliers of a given predictor value that also describe its magnitude or slope with respect to the response variable.



The graphs above display how many standard deviations away (left or right) observations in entire home/apt, private room and shared room are from the mean i.e. 0. In other words, the graphs of entire home/apt, private room and shared room represent the density function that represent these three classes. From a careful analysis, we can see that there is some overlap between the observations of three regions which can cause some uncertainty about the class to which those observations belong. To overcome this problem, we can assume that each observation stands an equally likely chance to belong to either of three classes.

Now our next step is to predict the model using our testing data. We use the predict() function in R to achieve this. Following is the code for the same along with its output:

```
predict_lda <-predict(m1,test_data)
```

The first input parameter to the above code is the fitting model we had created in the previous step and the second parameter the predict function is the testing data frame.

Output:

```
> predict_lda$class
  [1] Entire home/apt Entire home/apt Entire home/apt Entire home/ap
  [9] Entire home/apt Entire home/apt Private room    Entire home/ap
 [17] Entire home/apt Entire home/apt Entire home/apt Entire home/ap
 [25] Entire home/apt Entire home/apt Entire home/apt Entire home/ap
 [33] Private room    Private room    Entire home/apt Entire home/ap
```

```
> predict_lda$posterior
     Entire home/apt Private room  Shared room
2001       0.5524105 4.234416e-01 2.414795e-02
2002       0.7676206 2.210532e-01 1.132614e-02
2003       0.8789098 1.156510e-01 5.439179e-03
```

In the output above, we can see that the class attribute of the predict() function contains all the predicted room types based on the price. The posterior attribute of the predict() function gives us the posterior probability corresponding to each class.

Now, the next step is to calculate the accuracy of our model. The table() function produces a confusion matrix that gives the number of observations falling under each class that were predicted correct and the ones that were incorrect. We have then calculated the accuracy of our training data prediction on the test data, using the mean() function as shown in the screenshot below:

```
table(test_data$room_type, predict_lda$class)

mean(test_data$room_type == predict_lda$class )
```

Output:

```
> table(test_data$room_type, predict_lda$class)

                  Entire home/apt Private room Shared room
  Entire home/apt             774           40           0
  Private room                212          515           0
  Shared room                  13           29           0
>
```

```
> mean(test_data$room_type == predict_lda$class )
[1] 0.8142767
```

From the above output, we can interpret that 774 entire home/apt were correctly identified as such but 40 of entire home/ apt were identified as private rooms. 515 private rooms were correctly identified whereas 212 of those where identified as entire home/apt. We can also interpret that no shared rooms apartments were correctly identified. 13 of those were identified as entire home/apt and 29 as private room. This is because there are not many records in the training and test data for shared rooms. Also, we can see from the output of mean() function that the accuracy of our model is 81.42 percent.

The analysis above was done for Massachusetts data and, similarly, now we can perform the same analysis on New York’s data.

First, we divide our dataset into training and testing data. Following is the code to subset data into training and testing data.

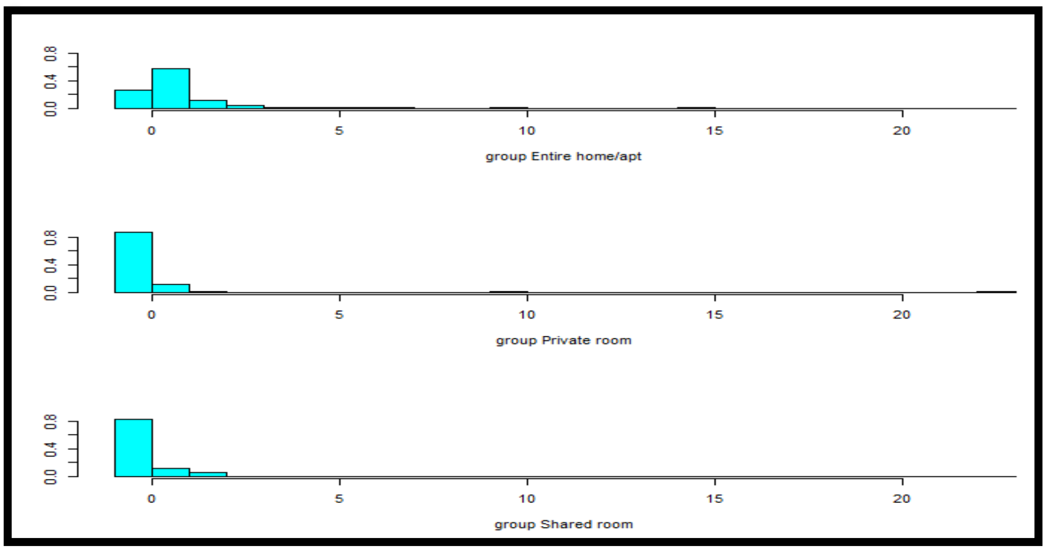
```
train_data_ny<- data_ny[1:2000,c(12,18)]
test_data_ny <- data_ny[2001:3999, c(12,18)]
```

Next, we create a fitting model using the lda() function based on the training data. Following is the output and screenshots of the fitting model.

```
m1_ny <- lda(room_type ~ price, data = train_data_ny)
plot(m1_ny)
```

```
> m1_ny
Call:
lda(room_type ~ price, data = train_data_ny)

Prior probabilities of groups:
Entire home/apt    Private room     Shared room
         0.6045          0.3870          0.0085

Group means:
                    price
Entire home/apt 198.05955
Private room     93.80362
Shared room      97.52941

Coefficients of linear discriminants:
              LD1
price 0.007720688
```

The above output gives the prior probabilities of each class i.e. 60.46% of our training data has entire home as the room type. Similarly, for private room and shared room, we have 38.70% and 0.85% respectively. The LDA model also gives us group means that gives us the average of the quantitative variable corresponding to each group. From the above output, we can interpret that for entire home/apt the average price is $198.05. Similarly, for private and shared room the average price is 93.80 and 97.52 respectively. The linear discriminant coefficients give the linear value for price which is further used as a basis for the LDA decision rule. It can also be described as the multipliers of a given predictor value that also describe its magnitude or slope with respect to the response variable.



The graphs above display how many standard deviations away (left or right) observations in entire home/apt, private room and shared room are from the mean i.e. 0. In other words, the graphs of entire home/apt, private room and shared room represent the density function that represent these three classes. From a careful analysis, we can see that there is some overlap between the observations of three regions which can cause some uncertainty about the class to which those observations belong. To overcome this problem, we can assume that each observation stands an equally likely chance to belong to either of three classes.

Now our next step is to predict the model using our testing data. We use the predict() function in R to achieve this. Following is the code for the same along with its output:

```
predict_lda_ny <-predict(m1,test_data_ny)
```

The first input parameter to the above code is the fitting model we had created in the previous step and the second parameter the predict function is the testing data frame.

```
> predict_lda_ny$class
  [1] Entire home/apt Private room    Entire home/a
  [9] Entire home/apt Entire home/apt Private room
 [17] Entire home/apt Entire home/apt Entire home/a
 [25] Private room    Entire home/apt Entire home/a
 [33] Entire home/apt Private room    Entire home/a
```

```
> predict_lda_ny$posterior
     Entire home/apt Private room  Shared room
5584       0.4910505 4.807874e-01 2.816211e-02
5585       0.4179215 5.488784e-01 3.320005e-02
5586       0.8735761 1.207160e-01 5.707869e-03
5587       0.5645511 4.120746e-01 2.337424e-02
```

In the output above, we can see that the class attribute of the predict() function contains all the predicted room types based on the price. The posterior attribute of the predict() function gives us the posterior probability corresponding to each class.

Now the next step is to calculate the accuracy of our model. The table() function produces a confusion matrix that gives the number of observations falling under each class that were predicted correct and the ones that were incorrect. We have then calculated the accuracy of our training data prediction on the test data, using the mean() function as shown in the screenshot below:

```
table(test_data_ny$room_type, predict_lda_ny$class)

mean( test_data_ny$room_type == predict_lda_ny$class )
```

```
> table(test_data_ny$room_type, predict_lda_ny$class)

                  Entire home/apt Private room Shared room
  Entire home/apt            1130          100           0
  Private room                232          512           0
  Shared room                   3           22           0
>
> mean( test_data_ny$room_type == predict_lda_ny$class )
[1] 0.8214107
```

From the above output, we can interpret that 1130 entire home/apt were correctly identified as such but 100 of entire home/ apt were identified as private rooms. 512 private rooms were correctly identified whereas 232 of those where identified as entire home/apt. We can also interpret that no shared rooms apartments were correctly identified. 3 of those were identified as entire home/apt and 22 as private room. This is because there are not many records in the train and test data for shared rooms. Also, we can see from the output of mean() function that the accuracy of our model is 82.14 percent.

Now we will perform K nearest neighbor classification on both New York's and Massachusetts data and compare the accuracy of our models.

4.1.2 K Nearest Neighbor (KNN):

A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in, by looking at the data points around it. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and then all those neighbors vote, so the maximum votes for whichever neighbor is, becomes the new data point. K is the number of neighbors it checks).

In R, the syntax and method of a KNN classification is different from other methods such as LDA and linear regression. In other models, we first predict our model using the training data and then using the predict() function and testing data, we predict our model to ensure there is not overfitting and underfitting whereas in the case of KNN classification, we just use a simple KNN() function which take input both the training and testing data along with the qualitative field.

First, we divide our data set into training and testing data based on Massachusetts data. Here, in KNN classification, we separate the qualitative fields with the quantitative fields. Following is the code to subset the training and testing data:

```
train_data_knn <- data_ma[1:2000,c(18)]
test_data_knn <- data_ma[2001:3583,c(18)]
train_data_ql <-data_ma[1:2000,c(12)]
test_data_ql <-data_ma[2001:3583,c(12)]
```

The train_data_knn vector contains training data for the price field. Similarly, test_data_knn vector contains test data for the price field. Now similarly, we create another vector for the qualitative field i.e. room type. The train_data_ql vector contains training data for the room type field. Similarly, test_data_ql vector contains testing data for the room type field.

Now that the data is ready, we will perform KNN classification using the knn() function in R. Following is the output and screenshot for the KNN classification function.

```
knn_model <- knn(data.frame(train_data_knn), data.frame(test_data_knn), train_data_ql , k = 1)
```

In the above code, the first parameter to the knn() function is the training data which contains the price vector. Here, we are converting it to a data frame using the data.frame() function. The next parameter is the test data for the price vector which is also being converted to a data frame. The third parameter is the qualitative field i.e. our response variable(room type). The fourth and the final parameter is the value of K(the number of neighbors to vote). In this case, the value of K is 1.

```
> knn_model
  [1] Private room    Entire home/apt Entire home/apt Entire home/apt
  [9] Entire home/apt Entire home/apt Private room    Entire home/apt
 [17] Entire home/apt Private room    Entire home/apt Entire home/apt
 [25] Private room    Entire home/apt Entire home/apt Entire home/apt
 [33] Private room    Shared room     Entire home/apt Entire home/apt
 [41] Entire home/apt Entire home/apt Entire home/apt Entire home/apt
```

The output of the KNN function returns predictions based on the testing data. Now our next step is to find the accuracy of model. The table() function produces a confusion matrix that gives the number of observations falling under each class that were predicted correct and the ones that were incorrect. We have then calculated the accuracy of our training data prediction on the test data, using the mean() function as shown in the screenshot below:

```
table(test_data_ql,knn_model)

mean(test_data_ql == knn_model)
```

```
> table(test_data_ql,knn_model)
                 knn_model
test_data_ql      Entire home/apt Private room Shared room
  Entire home/apt             703          110           1
  Private room                117          607           3
  Shared room                   8           34           0
>
> mean(test_data_ql == knn_model)
[1] 0.8275426
>
```

From the above output, we can interpret that 703 entire home/apt were correctly identified as such but 110 of entire home/ apt were identified as private rooms and 1 was identified as shared room. 607 private rooms were correctly identified whereas 117 of those where identified as entire home/apt and 3 were identified as shared rooms. We can also interpret that no shared rooms apartments were correctly identified. 8 of those were identified as entire home/apt and 34 as private room. This is because there are not many records in the train and test data for shared rooms. Also, we can see from the output of mean() function that the accuracy of our model is 82.75 percent.

Now we will increase the value of K. Below is the code for knn() function.

```
> knn_model <- knn(data.frame(train_data_knn), data.frame(test_data_knn), train_data_ql , k = 10)
>
> table(test_data_ql,knn_model)
                 knn_model
test_data_ql      Entire home/apt Private room Shared room
  Entire home/apt             706          108           0
  Private room                116          611           0
  Shared room                   8           34           0
>
> mean(test_data_ql == knn_model)
[1] 0.8319646
```

From the above about we can interpret that the accuracy of model increased as we increase the value of K. The accuracy of model with k = 10 is 83.196%.

Now we will do a similar KNN classification analysis on New York's data. Following is the code to subset the training and testing data:

```
train_data_knn_ny <- data_ny[1:2000,c(18)]
test_data_knn_ny <- data_ny[2001:3999,c(18)]
train_data_ql_ny <-data_ny[1:2000,c(12)]
test_data_ql_ny <-data_ny[2001:3999,c(12)]
```

The train_data_knn_ny vector contains training data for the price field. Similarly, test_data_knn_ny vector contains test data for the price field. Now similarly, we create another vector for the qualitative field i.e. room type. The train_data_ql_ny vector contains training data for the room type field. Similarly, test_data_ql_ny vector contains testing data for the room type field.

Now that the data is ready, we will perform KNN classification using the knn() function in R. Following is the output and screenshot for the KNN classification function.

```
knn_model_ny <- knn(data.frame(train_data_knn_ny), data.frame(test_data_knn_ny), train_data_ql_ny , k = 1)
```

In the above code, the first parameter to the knn() function is the training data which contains the price vector. Here, we are converting it to a data frame using the data.frame() function. The next parameter is the test data for the price vector which is also being converted to a data frame. The third parameter is the qualitative field i.e. our response variable(room type). The fourth and the final parameter is the value of K(the number of neighbors to vote). In this case, the value of K is 1.

```
> knn_model_ny
  [1] Private room    Private room    Entire home/apt
  [9] Entire home/apt Entire home/apt Private room
 [17] Entire home/apt Entire home/apt Entire home/apt
 [25] Private room    Entire home/apt Entire home/apt
 [33] Entire home/apt Private room    Entire home/apt
```

The output of the KNN function returns predictions based on the testing data. Now our next step is to find the accuracy of model. The table() function produces a confusion matrix that gives the number of observations falling under each class that were predicted correct and the ones that were incorrect. We have then calculated the accuracy of our training data prediction on the test data, using the mean() function as shown in the screenshot below:

```
table(test_data_ql_ny,knn_model_ny)

mean(test_data_ql_ny == knn_model_ny)
```

```
> table(test_data_ql_ny,knn_model_ny)
                 knn_model_ny
test_data_ql_ny   Entire home/apt Private room Shared room
  Entire home/apt            1045          185           0
  Private room                167          574           3
  Shared room                   3           22           0
>
> mean(test_data_ql_ny == knn_model_ny)
[1] 0.809905
```

From the above output, we can interpret that 1045 entire home/apt were correctly identified as such but 185 of entire home/ apt were identified as private rooms. 574 private rooms were correctly identified whereas 167 of those where identified as entire home/apt and 3 were identified as shared rooms. We can also interpret that no shared rooms apartments were correctly identified. 3 of those were identified as entire home/apt and 22 as private room. This is because there are not many records in the train and test data for shared rooms. Also, we can see from the output of mean() function that the accuracy of our model is 80.99 percent.

Now we will increase the value of K. Below is the code for knn() function.

```
> knn_model_ny <- knn(data.frame(train_data_knn_ny), data.frame(test_data_knn_ny), train_data_ql_ny , k = 10
> table(test_data_ql_ny,knn_model_ny)
                 knn_model_ny
test_data_ql_ny   Entire home/apt Private room Shared room
  Entire home/apt            1059          171           0
  Private room                165          579           0
  Shared room                   3           22           0
>
> mean(test_data_ql_ny == knn_model_ny)
[1] 0.8194097
```

From the above about we can interpret that the accuracy of model increased as we increase the value of K. The accuracy of model with k = 10 is 81.94 percent.

4.2 Clustering:

Clustering refers to finding subgroups in a dataset such that it is divided into distinct groups where each group consists of similar observations or have similar characteristics and different groups consist of distinct observations or have different characteristics. Clustering is a part of unsupervised learning as we attempt to discover a structure or pattern among observations contained in a dataset based on which distinct clusters or subgroups are created. In other words, clustering methods aim at finding homogeneous subgroups among observations contained in a dataset. In this project, we will be using K means clustering algorithm to find distinct groups.

K-means clustering method divides the observations into a pre-specified number of clusters, K. The K-means algorithm, then, assigns each observation to one of the K clusters. K-means clustering must satisfy the following properties:

1. $C1 \cup C2 \cup \ldots \cup CK = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $Ck \cap Ck' = \emptyset$ for all $k \neq k'$. In other words, the clusters are nonoverlapping: no observation belongs to more than one cluster.

Where, $C1 \cup C2 \cup \ldots \cup CK$ = sets or clusters (from 1 to K) containing observations
K-means clustering minimizes the variances within each cluster while it maximizes the variance in the overall dataset. Therefore, the clusters have a low-variability that makes them compact.

In our Airbnb's dataset, we aim to make clusters based on number of bedrooms and price. We have used the **'factoextra'** package and library to create plots for clusters in the section below.

First, we will analyze Massachusetts dataset by creating a subset of our original dataset. Following is the code to create a subset of dataset:

```
data_kmeans_ma <- data_ma[,c(15,18)]
```

We aim to divide this data into clusters and compare it with our original data, however, the comparison would become difficult as different numerical attributes lie within different range of values. Hence, we begin by, first, normalizing our data using the scale() function in R as shown below:

```
data_kmeans_ma_scaled <- scale(data_kmeans_ma)
```

	bedrooms	price
1	0.9876091	0.512579586
2	-0.3399783	-0.732931658
3	-0.3399783	-0.732931658
4	-0.3399783	-0.665606726
5	-0.3399783	-0.638676753
6	-0.3399783	-0.665606726
7	-0.3399783	-0.497294395
8	-0.3399783	-0.665606726
9	-0.3399783	-0.780059110
10	0.9876091	0.371197228

As we can see that the values of each attribute has been scaled with a mean of 0. Each value, hence, represents the number of standard deviations away it lies from the center of the data or from the mean.

To begin clustering, we need to pre-define the number of clusters that we want to divide our data into. We take K = 4. Similarly we can take the value of K to 2 and 3 and check for variance. We use the kmeans() function to perform clustering on our dataset in R as shown below:

```
set.seed(123)              #setting seed for kmeans
final <- kmeans(data_kmeans_ma_scaled, 4, nstart = 25)  #Kmeans clustering to make 3 clusters
attributes(final) #to get the attributes of final variable
```

Output:

```
> final
K-means clustering with 4 clusters of sizes 14, 748, 2598, 213

Cluster means:
    bedrooms      price
1  0.7031261  9.1210339
2  0.8846678  0.6179143
3 -0.4866363 -0.3121819
4  2.7826569  1.0382830

Clustering vector:
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
   2    3    3    3    3    3    3    3    3    2    3    3    2    2    2
  28   29   30   31   32   33   34   35   36   37   38   39   40   41   42
   3    3    3    3    2    3    3    3    3    3    4    4    3    3    3
  55   56   57   58   59   60   61   62   63   64   65   66   67   68   69
```

```
> attributes(final)
$`names`
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

$class
[1] "kmeans"
```

The output above shows the data that has been clustered into 3 subgroups along with their respective within cluster sum of squares that represents the measure of variability within each cluster. It is given by the sum of squared deviations from each observation and the cluster centroid which is calculated using the Euclidean distance between each observation and its respective cluster center. A cluster having a small sum of squares is a more compact cluster than the one that has a larger sum of squares.

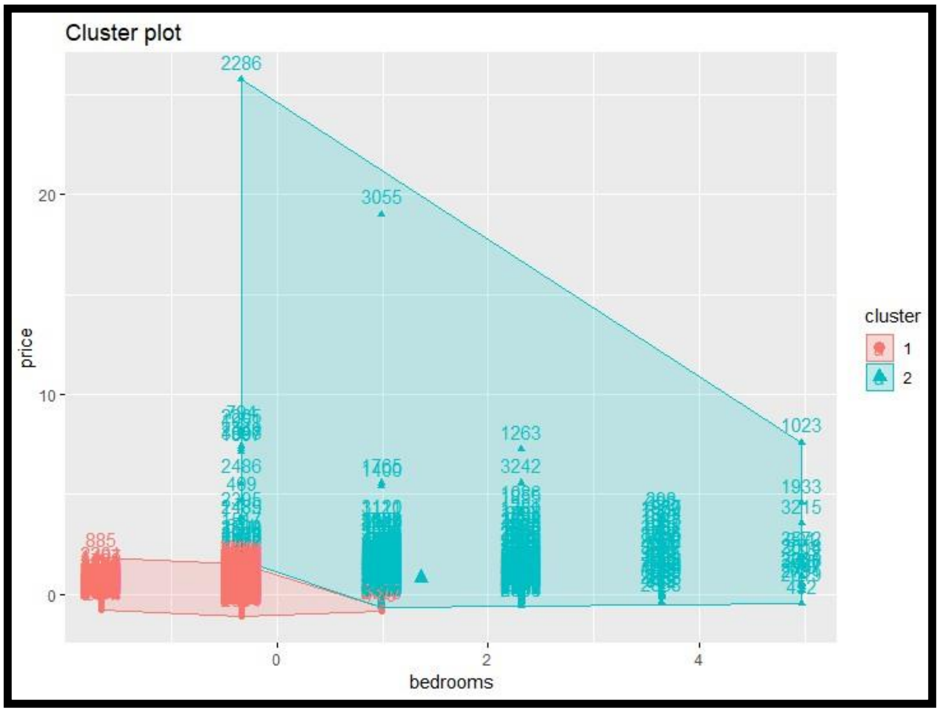
The clustering is performed in a loop till all observations are allocated to its nearest cluster center. The iteration process stops once the result can no longer be changed and when the maximum number of repetitions have been performed.

The attributes() function displays the different characteristics of our clustered data that involves cluster,centers, totss (total_ss), etc.

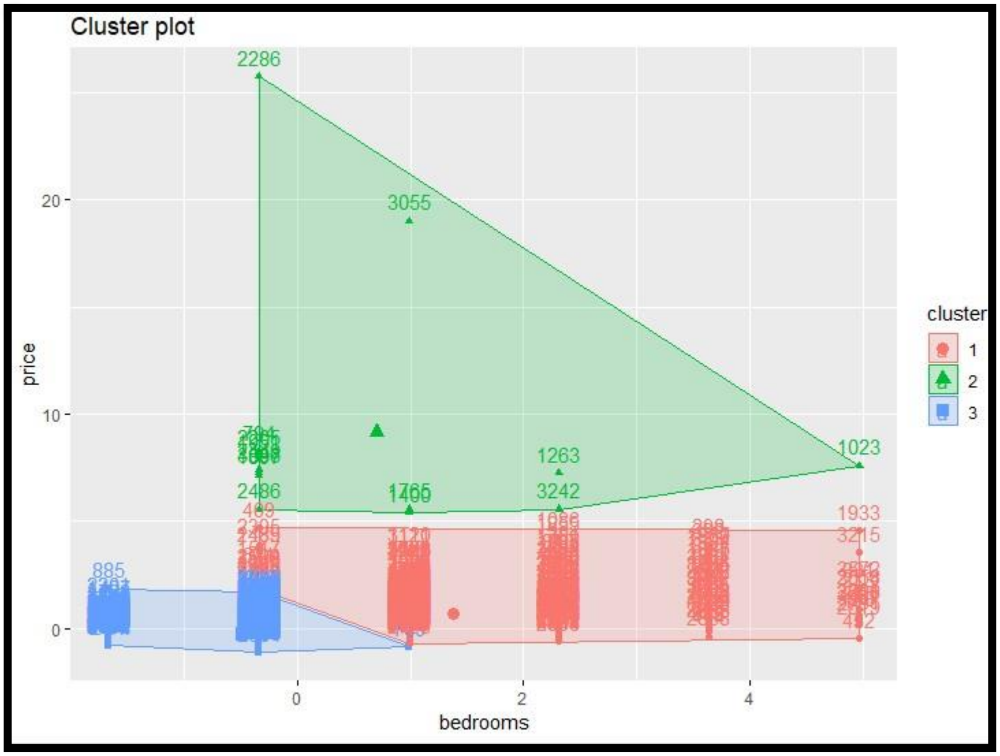
Next, we visualize our clustered data using cluster plot to get a better understanding of our new grouped data.

```
fviz_cluster(final, data = data_kmeans_ma_scaled)                #plot for clustering
```

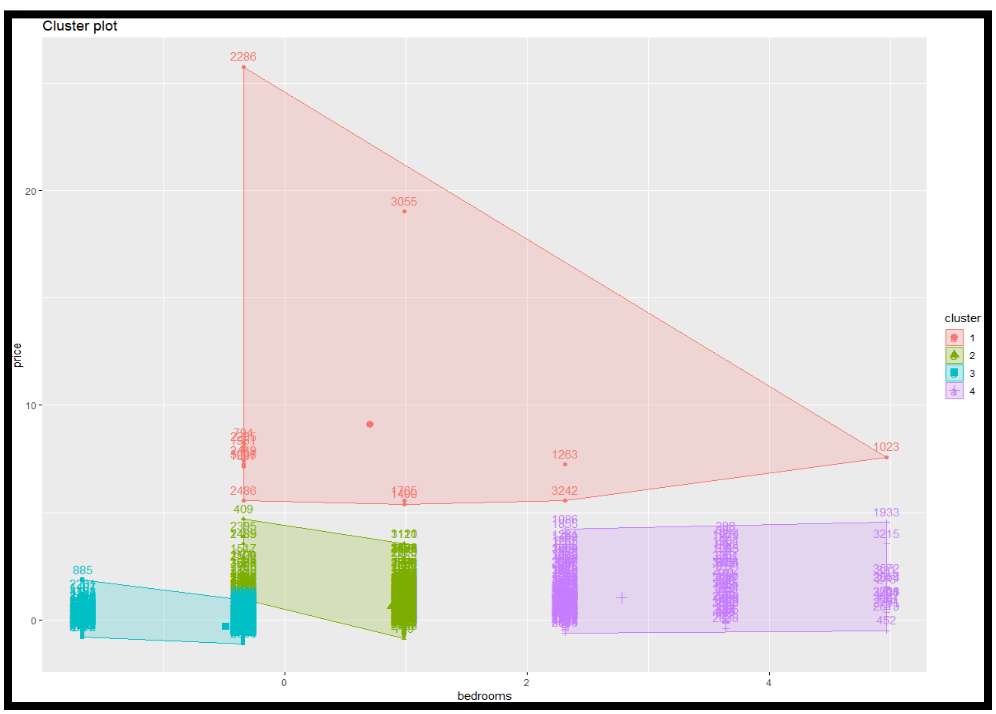
Output: For k = 2 :



For k = 3:


Cluster plot
price
bedrooms
cluster
1
2
3
2286
3055
1263
1023
3242
2486
1933
3215
885

For k =4:


Cluster plot
price
bedrooms
cluster
1
2
3
4
2286
3055
1263
1023
2486
3242
1933
3215
885
409
3120
452

We install the 'factoextra' package and library to plot our clustered data using the fviz_cluster() command in R. Using the cluster plot, we can also confirm the goodness of our clustered data. When we divide our data into 4 clusters, we can observe that we satisfy all properties of k-means such that each observation belongs to exactly one of the 4 clusters and the variance within each cluster is minimized. From the above plots, we can interpret that when there are 4 distinct clusters, the variance between each cluster is minimized which is not the case with 2 and 3 clusters. In the plot for 4 clusters,the green and blue cluster looks like they are overlapped but that is not the case. There are different distinct clusters. There are a lot of records in the dataset, hence the data labels are all scatter together at the boundaries of cluster 2 and 3 which is why they look like they are overlapping. From the above plot we can also interpret that high priced apartments are in cluster 1(red).

Now the next step is to analyze New York's dataset. We start by creating a subset of data for New York. Following is the code:

```
data_kmeans_ny <- data_ny[,c(15,18)]
```

We aim to divide this data into clusters and compare it with our original data, however, the comparison would become difficult as different numerical attributes lie within different range of values. Hence, we begin by, first, normalizing our data using the scale() function in R as shown below:

```
data_kmeans_ny_scaled <- scale(data_kmeans_ny)
```

Next, to begin clustering, we need to pre-define the number of clusters that we want to divide our data into. We take K = 4 . Similarly we can take the value of K to 2 and 3 and check for variance. We use the kmeans() function to perform clustering on our dataset in R as shown below:

```
## for k =4
final_ny <- kmeans(data_kmeans_ny_scaled, 4, nstart = 25)  #Kmeans clustering to make 4 clusters
attributes(final_ny) #to get the attributes of final variable
fviz_cluster(final_ny, data = data_kmeans_ny_scaled)          #plot for clustering
```

```
> final_ny
K-means clustering with 4 clusters of sizes 138, 8, 3138, 677

Cluster means:
     bedrooms      price
1  2.58513539  2.5318601
2  0.04691692 13.7916537
3 -0.41208022 -0.2008069
4  1.38254609  0.2517020

Clustering vector:
3584 3585 3586 3587 3588 3589 3590 3591 3592 3593 3594 3595 35
   3    3    3    3    3    3    3    3    3    3    3    4
3613 3614 3615 3616 3617 3618 3619 3620 3621 3623 3624 3625 36
   3    3    3    3    4    3    3    3    3    3    3    3
```

```
> attributes(final_ny)
$`names`
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

$class
[1] "kmeans"
```

The output above shows the data that has been clustered into 4 subgroups along with their respective within cluster sum of squares that represents the measure of variability within each cluster. It is given by the sum of squared deviations from each observation and the cluster centroid which is calculated using the Euclidean distance between each observation and its respective cluster center. A cluster having a small sum of squares is a more compact cluster than the one that has a larger sum of squares.

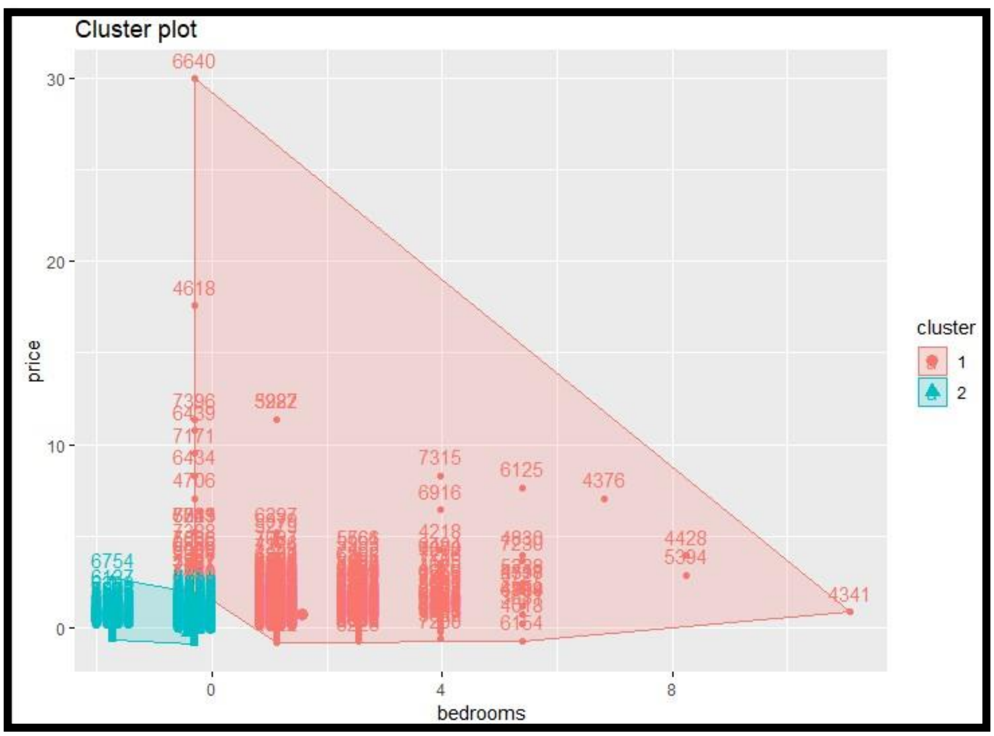
The attributes() function displays the different characteristics of our clustered data that involves cluster,centers, totss (total_ss), etc.

Next, we visualize our clustered data using cluster plot to get a better understanding of our new grouped data. Following is the code to visualize our clusters:

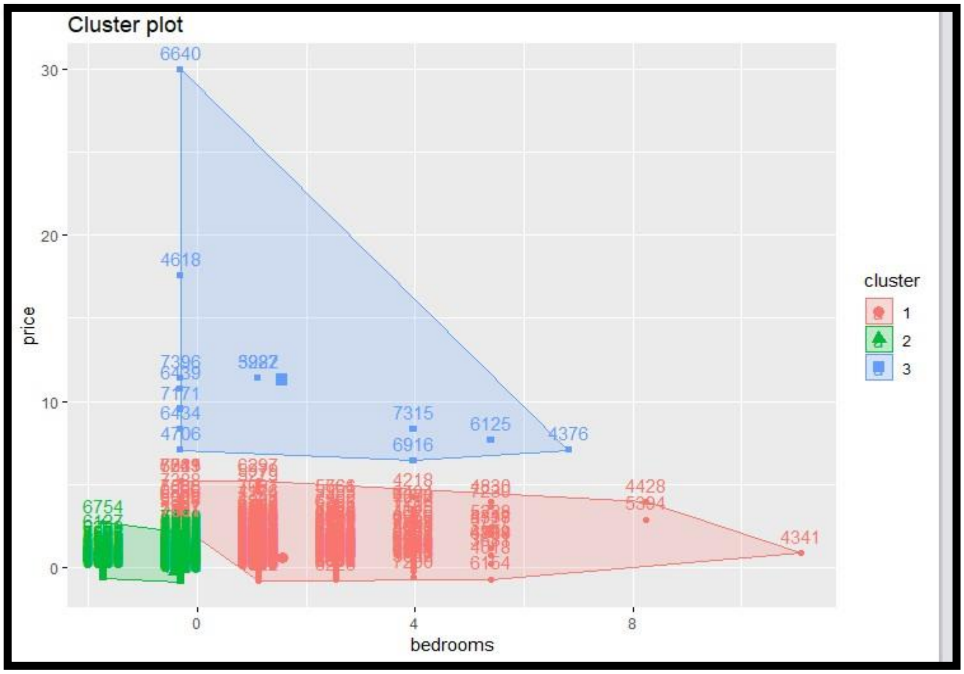
```
fviz_cluster(final_ny, data = data_kmeans_ny_scaled)            #plot for clustering
```

Output:

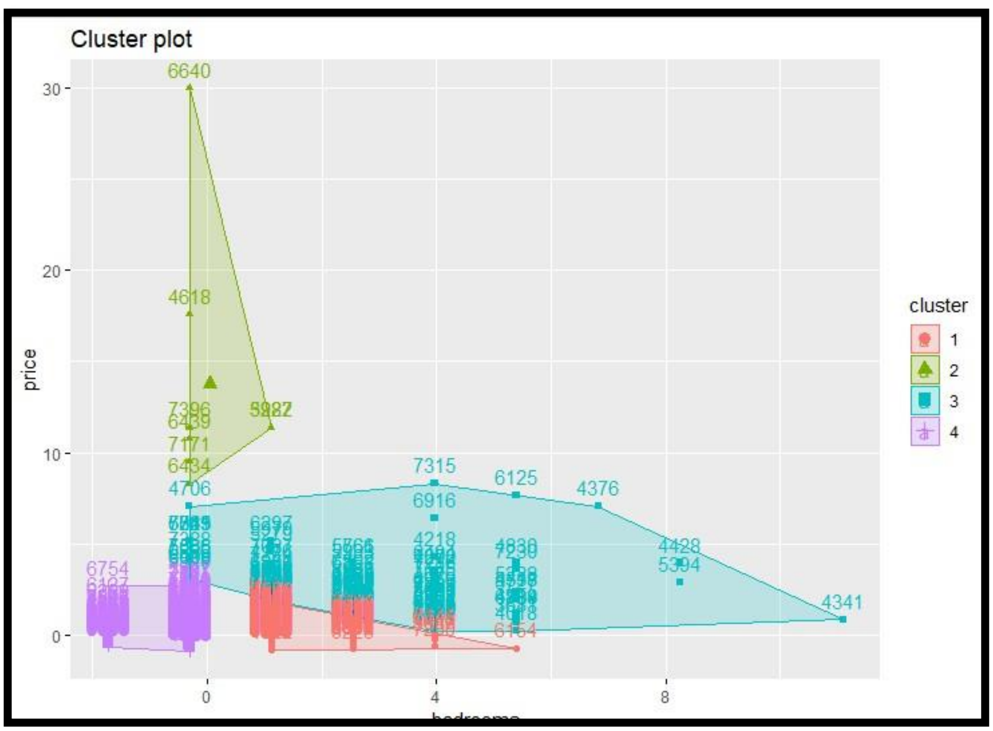
For k = 2:



For K =3:



For k = 4:



We install the ‘factoextra’ package and library to plot our clustered data using the fviz_cluster() command in R. Using the cluster plot, we can also confirm the goodness of our clustered data. When we divide our data into 4 clusters, we can observe that we satisfy all properties of kmeans such that each observation belongs to exactly one of the 4 clusters and the variance within each cluster is minimized. From the above plots, we can interpret that when there are 4 distinct clusters, the variance between each cluster is minimized which is not the case with 2 and 3 clusters. In the plot for 4 clusters, the red, purple,blue cluster looks like they are overlapped but that is not the case. They are different distinct clusters. There are a lot of records in the dataset, hence the data labels are all scatter together at the boundaries of cluster 1,3 and 4 which is why they look like they are overlapping. From the above plot we can also interpret that high priced apartments are in cluster 2(green).

Conclusion

- There exists no difference between the mean overnight prices of 3-bedrooms in NY and MA.
- There exists no difference between the sample variance in price of a 3 -bedroom listing in NY and MA.
- The mean price of a 3-bedroom listing in NY is not significantly different from $227 i.e. the claim stated by Airbnb on their website is true.
- The mean price of an Airbnb listing in MA is not significantly different from $338 or the claim made by Airbnb website regarding prices of 3-bedroom listings in Massachusetts is true.
- Country and currency variables were identified as variables having a single value (one-level) for all records and, hence, they were dropped from the dataset to perform linear regression.
- From the results of the linear model function, we note that the residuals of the price are calculated by comparing the estimated values with the actual values of the response variable, price. The minimum value is -298.1, max being the 4623.7 with the median being 0. This means that some value in the middle of the list is predicted correctly.
- The values which are marked with an asterisk corresponding to them, in the output of the different R-codes, denote that they are important for the model while considering Pr(>|t|) column. The asterisk signifies the p-values less than alpha are significant, while the p-values greater than alpha (level of significance) are statistically insignificant.
- The R-squared value is 0.4844, which implies that approx. 48.44% of variation in price (response variable) can be explained using the independent variables
- The analyses of various plots show us that the outliers exist in almost every large dataset but their significance on the response variable is determined using cook's distance and the linearity of the graph.
- The predict() and resid() function provided an insight that there are variables (predictor variables) which might not be important for the accuracy and analysis of our linear regression model. Thus, there arises a need to perform regularization using LASSO to shrink the coefficient of the irrelevant variables to 0 and consider only those variables that are relevant and determine the price (response variable) correctly.
- Regularization technique using the LASSO model, shrinks the coefficients of the irrelevant variables to zero and gives us 3 most relevant variables affecting the price, that are: room_type, accommodates and bedrooms.
- For Massachusetts dataset, LDA model gave us an accuracy of 81.42% whereas KNN classification model gave an accuracy of 83.196%. Hence, KNN classification method is better suited on Massachusetts dataset.
- For New York’s dataset, LDA model gave us an accuracy of 82.14% whereas KNN classification model gave an accuracy of 81.94%. Hence, LDA classification method is better suited on New York’s dataset.
- We can infer(for both MA and NY) that when the number of clusters are 4, all the properties of k-means clustering are satisfied such that each observation belongs to exactly one of the 4 clusters and the variance within each cluster is minimized.

References

- *Airbnb. (2016, November 17). Boston Airbnb Open Data. Retrieved from* https://www.kaggle.com/airbnb/boston/version/1#listings.csv
- *New York listings.csv Retrieved from* http://insideairbnb.com/get-the-data.html
- *Vacation Homes & Condo Rentals. (n.d.). Retrieved from* https://www.airbnb.com/s/homes?refinement_paths[]=/homes&query=Massachusetts, United States &adults=1&children=0&infants=0&guests=0&checkin=2019-02-16&checkout=2019-02-17&min_bedrooms=3&allow_override[]=&s_tag=5laHK5U4
- *New York 2019 (with Photos): Top 20 Places to Stay in New York - Vacation Rentals, Vacation Homes - Airbnb New York, New York, United States. (n.d.). Retrieved from* https://www.airbnb.com/s/New-York--NY--United-States/homes?refinement_paths[]=/homes&query=New York, NY, United States&adults=0&children=0&infants=0&guests=0&place_id=ChIJOwg_06VPwokRYv534QaPC8g &min_bedrooms=3&checkin=2019-02-20&checkout=2019-02-22&allow_override[]=&s_tag=fSSWqQDt
- *Hypothesis Testing. (n.d.). Retrieved from* https://www.statisticssolutions.com/hypothesis-testing/
- *Prabhakaran, S. (n.d.). Eval(ez_write_tag([[728,90],'r_statistics_co-box-3','ezslot_4']));Outlier Treatment. Retrieved from* http://r-statistics.co/Outlier-Treatment-With-R.html
- *Yurtoğlu, N. (2018).* http://www.historystudies.net/dergi//birinci-dunya-savasinda-bir-asayis-sorunu-sebinkarahisar-ermeni-isyani20181092a4a8f.pdf. *History Studies International Journal of History*,*10*(7), 241-264.
- *Wetherill, C. (2018, January 29). How to Perform T-tests in R. Retrieved from* https://datascienceplus.com/t-tests/
- *Testing a Variance in R. (n.d.). Retrieved from* https://www.dummies.com/education/math/statistics/testing-variance-r/
- *Hmisc. (n.d.). Retrieved from* https://www.rdocumentation.org/packages/Hmisc/versions/4.2-0/topics/summarize
- *Bommae. (n.d.). University of Virginia Library Research Data Services Sciences. Retrieved* from https://data.library.virginia.edu/diagnostic-plots/
- *Prabhakaran, S. (n.d.). Eval(ez_write_tag([[728,90],'r_statistics_co-box3','ezslot_5']));Linear Regression. Retrieved from* http://r-statistics.co/LinearRegression.html
- *R Tutorial Series: Simple Linear Regression. (2015, February 12). Retrieved from* https://www.r-bloggers.com/r-tutorial-series-simple-linear-regression/
- *Stats. (n.d.). Retrieved from* https://www.rdocumentation.org/packages/stats/versions/3.5.2/topics/residuals
- *(n.d.). Retrieved from* https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.lm.html
- *HDCI. (n.d.). Retrieved from* https://www.rdocumentation.org/packages/HDCI/versions/1.0-2/topics/Lasso
- *KaleabKaleab 147113, JohnKJohnK 12k74086, EdMEdM 21.5k23396, & William ChiuWilliam Chiu 348211. (n.d.). Lasso Regression for predicting Continuous Variable Variable Selection? Retrieved from* https://stats.stackexchange.com/questions/188753/lasso-regression-for-predicting-continuous-variable-variable-selection

- *R Data Frame (Create, Access, Modify and Delete Data Frame in R). (2018, October 08). Retrieved from* https://www.datamentor.io/r-programming/data-frame/
- *Nagpal, A., & Nagpal, A. (2017, October 13). L1 and L2 Regularization Methods – Towards Data Science. Retrieved from* https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
- *R Matrix (Create and Modify Matrix, and Access Matrix Elements). (2018, October 08). Retrieved from* https://www.datamentor.io/r-programming/matrix/
- *Moulines, E. (n.d.). Retrieved from* https://rstudio-pubs-static.s3.amazonaws.com/224739_0b86cafbb6d5421ab92c9dc1dd982690.html
- *Curious 2, JPico6 212, Clarinetist 1, MamboSC4649mamboSC4649 413, & Danny Wong 558. (n.d.). Intepretation of crossvalidation result - cv.glm(). Retrieved from* https://stats.stackexchange.com/questions/48766/intepretation-of-crossvalidation-result-cv-glm
- *Linear discriminant analysis. (2019, January 26). Retrieved from* https://en.wikipedia.org/wiki/Linear_discriminant_analysis
- *Classification with Linear Discriminant Analysis. (2016, December 23). Retrieved from* https://www.r-bloggers.com/classification-with-linear-discriminant-analysis/
- *Computing and visualizing LDA in R. (2014, January 15). Retrieved from* https://www.r-bloggers.com/computing-and-visualizing-lda-in-r/
- *Srivastava, T. (2018, March 27). Introduction to KNN, K-Nearest Neighbors: Simplified. Retrieved from* https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
- *KNN Classification using Scikit-learn. (n.d.). Retrieved from* https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
- *Stats.(n.d.). Retrieved from* https://www.rdocumentation.org/packages/stats/versions/3.5.2/topics/kmeans
- *Galili, T. (2013, August 07). K-means Clustering (from 'R in Action'). Retrieved from* https://www.r-statistics.com/2013/08/k-means-clustering-from-r-in-action/

Appendix

1 Hypothesis testing

nymalistings <- read.csv(file.choose(), header = TRUE, na.string = "") #Loading data

#commands to install the packages

install.packages("outliers") #package to detect the outliers

install.packages("rcompanion") #package to create normal distribution cure, histogram

install.packages("lars") #package to perform linear and lasso regression

install.packages("glmnet") #package to perform linear and lasso regression

install.packages("ggplot2") #package to plot graphs

install.packages("class") #package to perform classification

install.packages("MASS") #package to perform classification

install.packages("factoextra") #package to create clustering plots

library(outliers) #package to detect the outliers

library(rcompanion) #package to create normal distribution cure, histogram

library(lars) #package to perform linear and lasso regression

library(glmnet) #package to perform linear and lasso regression

library(ggplot2) #package to plot graphs

library(class) #package to perform classification

library(MASS) #package to perform classification

library(factoextra) #package to create clustering plots

nylistings <- subset(nymalistings, state=='NY'& bedrooms=='3') #Filtering data by state and no. of bedrooms for NY

malistings <- subset(nymalistings, state=='MA' & bedrooms=='3') # Filtering data by state and no. of bedrooms for MA

cleannylistings <- rm.outlier(nylistings$price, fill = FALSE, median = FALSE, opposite = FALSE) #Remove outliers NY

cleanmalistings <- rm.outlier(malistings$price, fill = FALSE, median = FALSE, opposite = FALSE) #Remove outliers MA qqnorm(cleannylistings,main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)') # Check normality NY

qqline(cleannylistings,main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)') # Check normality NY

plotNormalHistogram(cleannylistings, main = "3 bedroom overnight Prices in NY", xlab = 'Price (in dollars)', xlim = c(10,900)) # Check normality NY

qqnorm(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)') #Check normality MA

qqline(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)') #Check normality MA

plotNormalHistogram(cleanmalistings, main = "3 bedroom overnight Prices in MA", xlab = 'Price (in dollars)', xlim = c(10,750)) #Check normality MA

nysample <- sample(cleannylistings,30) #Taking a random sample of 30 from NY, 3 bedroom listings

masample <- sample(cleanmalistings,30) #Taking a random sample of 30 from MA, 3 bedroom listings

t.test(nysample, masample, alternative = 'two.sided') #Testing the difference between Mean prices of 3 bedrooms in NY and MA

var.test(nysample, masample, alternative = "two.sided") #Testing the difference in variances in 3 bedroom prices in NY and MA

t.test(nysample, alternative = 'two.sided', mu=227) #Testing claim mean price in NY for 3 bedrooms = 227

t.test(masample, alternative = 'two.sided', mu=338) #Testing claim mean price in MA for 3 bedrooms = 338

###2 Multile Linear Regression

summary(nymalistings) #summary of the dataset represents the min, max, and quartile values

ggplot(nymalistings, aes(x=price)) + geom_histogram(binwidth=10) + labs(x="price") #plotting the minimum,max,1st quartile,3rd quartile,mean etc for price variable

(l <- sapply(nymalistings, function(x) is.factor(x))) #sapply function is used to determine which columns/variables are factors and return a vector

m <- data.frame(nymalistings[, l]) #function to create a data frame ifelse(n <- sapply(m, function(x) length(levels(x))) == 1, "DROP", "NODROP") #This command is used to determine which of the variables in the dataset are factors or characters which take a single value for every input

which(sapply(m, function(x) length(unique(x))<2)) #command to calculate the number of level/length and return only unique value

linearMod <- lm(price ~ id + host_id + street + neighbourhood_cleansed + city + state + zipcode + latitude + longitude + property_type + room_type + accommodates + bathrooms + beds +bed_type + review_scores_rating, data=nymalistings) #function to implement linear regression for price(dependent variable) over other independent variables

summary(linearMod) #summary of the linear model

plot(linearMod) #function to plot the linear model ##continue pressing enter(return) button in the console window to view more plots. There are 4 plots related linear mod.

predict_price <- predict(linearMod) #predict() function to predict the values of the price variable and compare it with the original dataset

predict_price

residual_price <- data.frame(resid(linearMod)) #resid() function to calculate the residue, which is the difference between the original price data and the predicted value

format(residual_price, scientific=FALSE)

3. Lasso Regularization

DF <-data.frame(nymalistings$id, nymalistings$host_id, nymalistings$street, nymalistings$neighbourhood_cleansed, nymalistings$city, nymalistings$state, nymalistings$zipcode, nymalistings$latitude, nymalistings$longitude, nymalistings$property_type, nymalistings$room_type, nymalistings$accommodates,nymalistings$bathrooms, nymalistings$bedrooms,nymalistings$beds, nymalistings$bed_type, nymalistings$review_scores_rating, nymalistings$price) #Created a dataframe

DF <- na.omit(DF) #omitted the null/missing values from the dataframe

x<- data.frame(DF$nymalistings.id, DF$nymalistings.host_id, DF$nymalistings.street, DF$nymalistings.neighbourhood_cleansed, DF$nymalistings.city, DF$nymalistings.state, DF$nymalistings.zipcode, DF$nymalistings.latitude, DF$nymalistings.longitude, DF$nymalistings.property_type, DF$nymalistings.room_type, DF$nymalistings.accommodates, DF$nymalistings.bathrooms, DF$nymalistings.bedrooms, DF$nymalistings.beds, DF$nymalistings.bed_type, DF$nymalistings.review_scores_rating) #Combining data together of all variables

y <- DF$nymalistings.price #price variable for lasso model

Mx <- as.matrix(as.data.frame(lapply(x, as.numeric))) #creating a matrix

lasso_model <- cv.glmnet(Mx , y, alpha = 1) #Setting up the model for lasso

lasso_model1 <- glmnet(Mx,y) #setting up the model

plot.cv.glmnet(lasso_model) #Plot of cross valildation error according to log lambda values

fit <- glmnet(x=Mx, y=y, alpha = 1, lambda= lasso_model$lambda.min) #When lambda is minimum

fit$beta #diplay the important features

fit1 <- glmnet(x=Mx, y=y, alpha = 1, lambda= lasso_model$lambda.1se) #When lambda is maximum

fit1$beta #diplay the important features

plot.glmnet(lasso_model1) #Plot showing path of coefficient of variables against L1-norm as lambda varies

###4 Data Mining

###4.1 Classification

###4.1.1 LDA CLassification

##LDA Classification on MA dataset

data_ny <- subset(nymalistings, state == 'NY') #Data for New York

data_ma <- subset(nymalistings, state == 'MA') #Data for Massachusetts

train_data<- data_ma[1:2000,c(12,18)]   #training data for MA

test_data <- data_ma[2001:3583, c(12,18)]   #testing data for MA

m1 <- lda(room_type ~ price, data = train_data) #lda model for MA dataset

plot(m1) #lda plot for MA dataset

predict_lda <-predict(m1,test_data) #prediction based on test data for MA dataset

predict_lda$class #class variables for prediction

predict_lda$posterior #posterior probabilities based on predictions

table(test_data$room_type, predict_lda$class) #cross table to find accuracy mean(test_data$room_type == predict_lda$class) #find mean of matched data to find accuracy

##LDA CLassification on NY dataset

train_data_ny<- data_ny[1:2000,c(12,18)]   #training data for NY

test_data_ny <- data_ny[2001:3999, c(12,18)]   #testing data for NY

m1_ny <- lda(room_type ~ price, data = train_data_ny) #fitting model creation

plot(m1_ny) #plot of fit model

predict_lda_ny <-predict(m1,test_data_ny) #prediction on testing data

predict_lda_ny$class #class variables i.e. predicted variables

predict_lda_ny$posterior #posterior probability of classes

table(test_data_ny$room_type, predict_lda_ny$class) #cross table for accuracy

mean(test_data_ny$room_type == predict_lda_ny$class) #finding mean of matched values

###4.1.2 K nearest Neighbor Classification

##KNN for MA dataset

train_data_knn <- data_ma[1:2000,c(18)] #training data for quantitative variables

test_data_knn <- data_ma[2001:3583,c(18)]# testing data for quantitative variables

train_data_ql <-data_ma[1:2000,c(12)] #training data for qualitative variable

test_data_ql <-data_ma[2001:3583,c(12)]#testing data for qualitative variable

knn_model <- knn(data.frame(train_data_knn), data.frame(test_data_knn), train_data_ql , k = 1) #creating a knn model with k = 1

knn_model <- knn(data.frame(train_data_knn), data.frame(test_data_knn), train_data_ql , k = 10) #creating a knn model with k = 10

table(test_data_ql,knn_model) #cross table to check accuracy

mean(test_data_ql == knn_model) #to find the matched values by the predicted system

##KNN for NY dataset

train_data_knn_ny <- data_ny[1:2000,c(18)] #training data for quantitative variables

test_data_knn_ny <- data_ny[2001:3999,c(18)] #training data for quatitative variables train_data_ql_ny <-data_ny[1:2000,c(12)] #training data for qualitative variables

test_data_ql_ny <-data_ny[2001:3999,c(12)]#training data for qualitative variables

knn_model_ny <- knn(data.frame(train_data_knn_ny), data.frame(test_data_knn_ny), train_data_ql_ny , k = 1) #creating a knn model with k = 1

knn_model_ny <- knn(data.frame(train_data_knn_ny), data.frame(test_data_knn_ny), train_data_ql_ny , k = 10) #creating a knn model with k = 10

table(test_data_ql_ny,knn_model_ny) #cross table to check accuracy

mean(test_data_ql_ny == knn_model_ny) #to find the matched values by the predicted system

###4.2 Clustering

##Kmeans clustering on MA

data_kmeans_ma <- data_ma[,c(15,18)] #data for kmeans(bedrooms and price variables)

data_kmeans_ma <- na.omit(data_kmeans_ma) #removing NA values from dataset

data_kmeans_ma_scaled <- scale(data_kmeans_ma) #normalizing data using scale function

set.seed(123) #setting seed for kmeans

for k =2

final <- kmeans(data_kmeans_ma_scaled, 2, nstart = 25) #Kmeans clustering to make 3 clusters

attributes(final) #to get the attributes of final variable

fviz_cluster(final, data = data_kmeans_ma_scaled) #plot for clustering

###for k =3

final <- kmeans(data_kmeans_ma_scaled, 3, nstart = 25) #Kmeans clustering to make 3 clusters

attributes(final) #to get the attributes of final variable

fviz_cluster(final, data = data_kmeans_ma_scaled) #plot for clustering

for k = 4

final <- kmeans(data_kmeans_ma_scaled, 4, nstart = 25) #Kmeans clustering to make 3 clusters

attributes(final) #to get the attributes of final variable

fviz_cluster(final, data = data_kmeans_ma_scaled) #plot for clustering

##Kmeans clustering on NY

data_kmeans_ny <- data_ny[,c(15,18)] #data for kmeans(bedrooms and price variables)

data_kmeans_ny <- na.omit(data_kmeans_ny) #removing NA values from dataset

data_kmeans_ny_scaled <- scale(data_kmeans_ny) #normalizing data using scale function

set.seed(123) #setting seeds for k means

for k =2

final_ny <- kmeans(data_kmeans_ny_scaled, 2, nstart = 25) #Kmeans clustering to make 2 clusters

attributes(final_ny) #to get the attributes of final variable

fviz_cluster(final_ny, data = data_kmeans_ny_scaled) #plot for clustering

for k =3

final_ny <- kmeans(data_kmeans_ny_scaled, 3, nstart = 25) #Kmeans clustering to make 3 clusters

attributes(final_ny) #to get the attributes of final variable

fviz_cluster(final_ny, data = data_kmeans_ny_scaled) #plot for clustering

for k =4

final_ny <- kmeans(data_kmeans_ny_scaled, 4, nstart = 25) #Kmeans clustering to make 4 clusters

attributes(final_ny) #to get the attributes of final variable

fviz_cluster(final_ny, data = data_kmeans_ny_scaled) #plot for clustering

###END###