Intelligent Data Analytics

Predicting Airbnb house price
Project: Final Report
Project Group No. 35

Submitted by:
Bhairavsingh Ghorpade OU Id: 113435739
Abhishek Kumar Gupta OU Id: 113434764
Dharani Kumar OU Id: 113420414

Contents

Executive Summary

Today travelling is way more comfortable and enjoyable than earlier days. It's not only due to ease of travelling or transportation but also a warm and home like stay makes it even more perfect. Many hospitalization chains of well-known hotels offer various types of accommodations and a home like feeling. But eventually it somehow burdensome our pocket. In recent years, a pocket friendly yet comfortable and homely accommodation option has been growing. The option is Airbnb.

As the business grows it gives a lot of space to move based on its performance. Same with the Airbnb as well. Since it's throughout the United States, we can get enormous number of data for predicting or analyzing various business attributes. One of them is predicting price for a house. The prediction of price is welcomed by not only the travelers but also the hosts as they can predict their generating revenue.

Hence in our project we are keen to analyze the data we got from ***Kaggle.com*** and predict the house price at a given location and with given specification.

In our project we are visualizing the data and then transforming it to appropriate structure. Some of feature engineering such as calculating the days of a house since it started hosting the guests. Later, we are using models such as **Linear Regression**, **Ridge Regression**, **Lasso Regression**, **Elastic Net Regression, PLRS regression** and **Random Forest** for predicting the house price.

Project Description

Airbnb is a privately owned global company with headquarter in San Francisco. The company operates online marketplace hospitality service which can be access through mobile or websites. The company does not own any of the real estate listings but receives commission from every booking.

The dataset used in this study uses the details from the Airbnb data set which contains several predictor variables such as number of ID of the house listed on the website, number of the bedrooms, bathrooms, beds, accommodation, host identity, customer review and ratings etc. The dataset also gives the logarithm of the housing price listed on the Airbnb.

The aim of this project is to identify the key factors affecting the Airbnb price and fit a best model to predict the house rent based on the different listed parameters. We are predicting logarithmic value of house rent.

Dataset Description

The dataset consists of 74111 observations form different city and locations with 29 attributes. The features/attributes are basically of different types including text, categorical and text. We will discuss the variables in detail according to its type.

Numerical attributes

As shown in table 01, we can see that, there are 13 numerical variables. All 13 variables are quantitative variables and we did not find any qualitative variables in those.

Categorical attributes

As shown in table 02, we can see that, there are 10 categorical variables.

Date and text attributes

As shown in table 03, we can see that, there are 7 date and text variables.

Data Analysis/cleansing

The dataset consists of several attributes with missing values. The dataset was analyzed to see the number of missing observations. We found variables and the missing values which are listed in table 04.

From table. 04, Since the dataset has lots of missing observations, data imputation has been done to impute values of larger missing observations. Before that, we have removed many of above attributes which has higher number of missing values, we removed attributes like "review_score_raitng" and "host_response_rate" which has around 20% of missing values, imputing by such a big number of values will cause a biased prediction hence we removed such variables. Also, considering the large number of observations present in the dataset, attributes with lower (<300) missing values were simply deleted instead imputation as they constitute less than 1% of the dataset. We have used the Predictive Mean Imputation (PMM) method to impute values in missing fields. In this method, what is done that the missingness is removed by imputing the value of the observation by the predictive mean of the rest of the observation.

Feature selection

The dataset has around 29 variables. However, looking at the factor levels of some attributes like *id, description, name, thumbnail, url etc,* we decided to only select those features which have lower factor levels. Initial feature selection is done to select only 21 attributes out of 29.

Most of the variables like "***id***" doesn't make any since for the prediction. Even though it's a numerical variable, its uniqueness will not help in the prediction of price. Variable like "***amenities***", "***decreiption***", "***name***", "***thumbnail_url***" has many factor levels also, these are long text attributes which cannot be used efficiently and effectively for prediction or improving the existing model. Some attributes like "***first review date***" and "***last review date***" are dates, and the dates of review does not help us for predicting the price. Hence, we decided to remove these attributes.

Feature Transformation

After selecting the appropriate features next step is to do necessary transformation. After visual inspecting the dataset and its structure, we found that, "***host_response rate***" has numerical followed by percentage (%) symbol, Hence we used ***as.numeric*** function in R to remove it and make it pure numerical variable.

Feature Engineering

Attribute like "***host_since***" has date of first hosted guest. We need this to convert to days. We have subtracted the host_since date through 2018 – 12 – 01, to get the number of days until first of December 2018. This plays an important role in prediction because

Test/train dataset

The whole dataset has been partitioned in the 70-30 ratio. The 70% of observations have been used for training purpose while the rest 30% used for testing the validity of the model.

Modelling

We have used different models including linear regression, ridge regression, LASSO, elastic net and PLS. We also ran random forest on the dataset for more robust modelling. The model is made to predict the log_price using the attributes discussed in the data preparation. Each model was run with train dataset containing 70% of the total data and validated using the test data consisting of rest of the 30% of the whole dataset. The root mean square error has been used as a main parameter to determine the efficacy of each model. Different hyper parameters which has been tuned to get most optimized models have been discussed in the respective model descriptions.

Selection of appropriate features was the key for modelling. Therefore, we looked in to different co-relation plots between the attributes and selected the features which are pretty well co-related with the value we are predicting i.e., log_price and having low inter co-relation with the rest of the attributes.

Linear regression

Linear regression is applied after the data preparation and data imputation of missing values. Model was run on train data and validated using test data. The rmse value was then calculated to determine the efficacy of the model. Several residual analyses were further done which is discussed below:

```
Call:
lm(formula = log_price ~ ., data = trainData)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.8160 -0.2486 -0.0189  0.2134  3.6421 


Residual standard error: 0.4242 on 48617 degrees of freedom
Multiple R-squared:  0.6549,	Adjusted R-squared:  0.6502 
F-statistic: 138.9 on 664 and 48617 DF,  p-value: < 2.2e-16
```

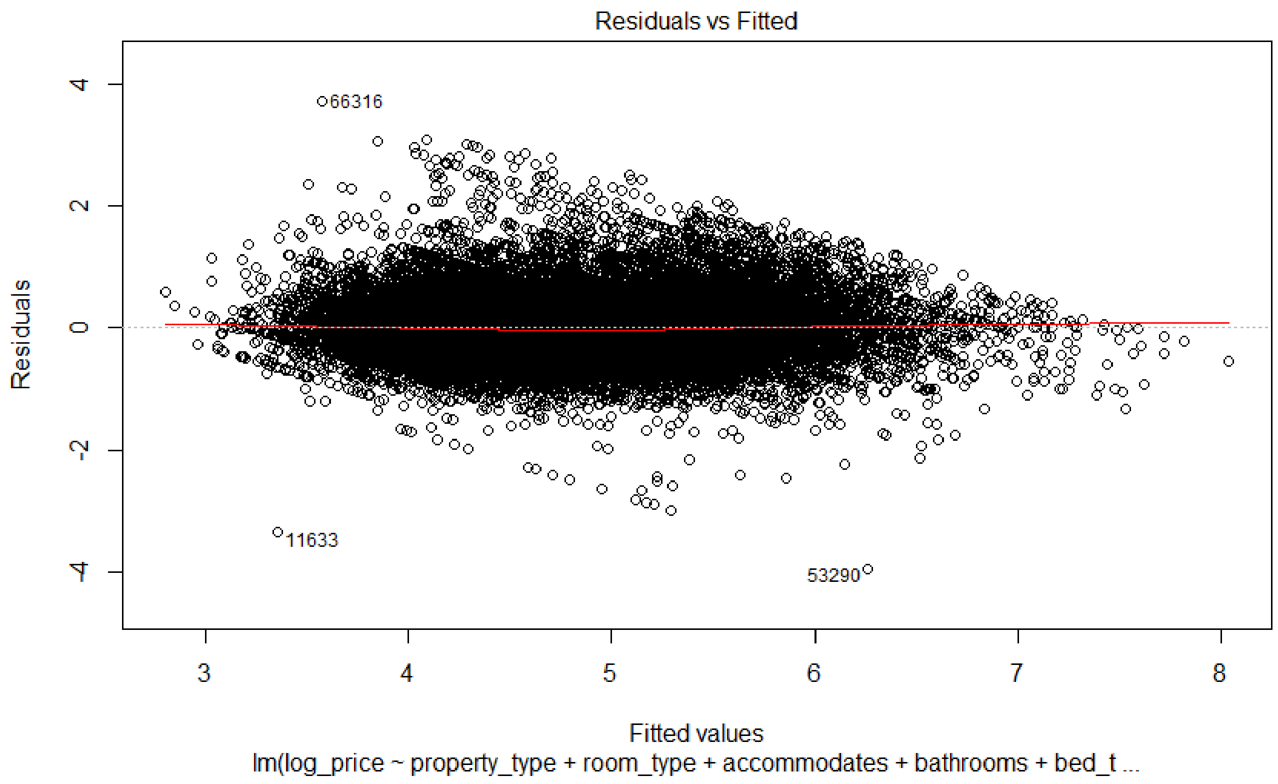
The coefficients corresponding to each attributes have been shown in the appendix. From the regression model we get the adjusted r^2 value as 0.65 which suggest that model is able to explain the 65% variance in the dataset and rmse value on the test data is 0.42. The AIC value is 56001 and BIC value is 61865.

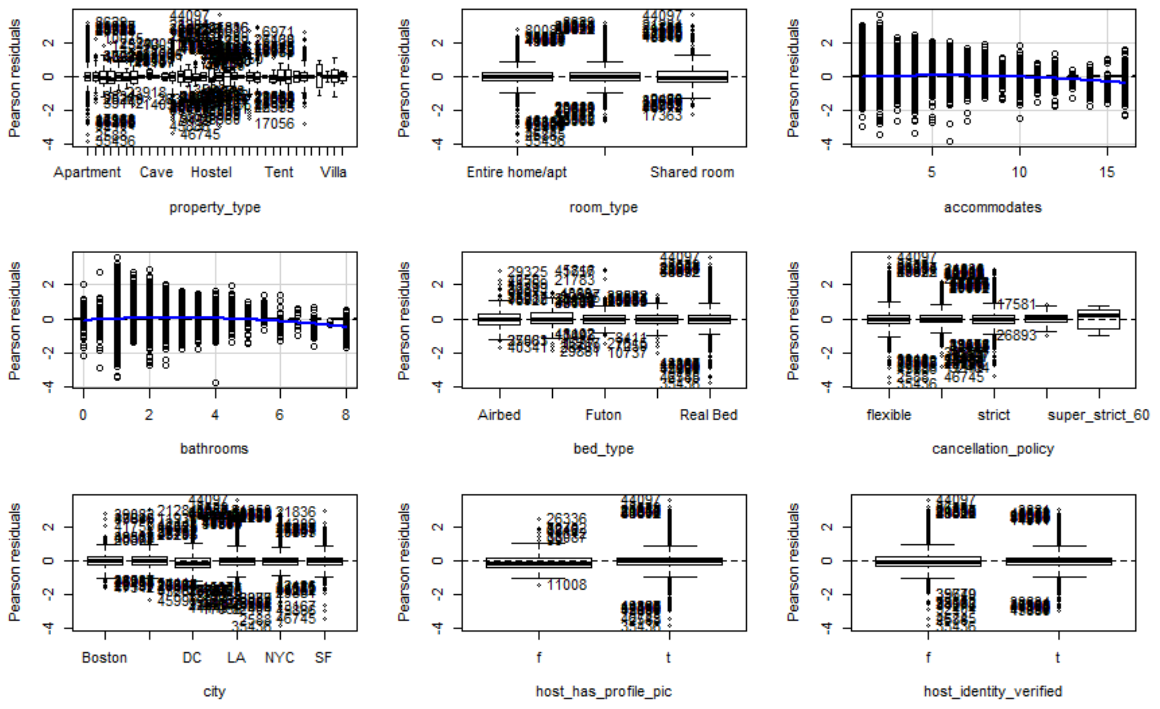
We also tried the stepwise regression with attributes. However, it didn't show any improvement in the R^2 and rmse values.

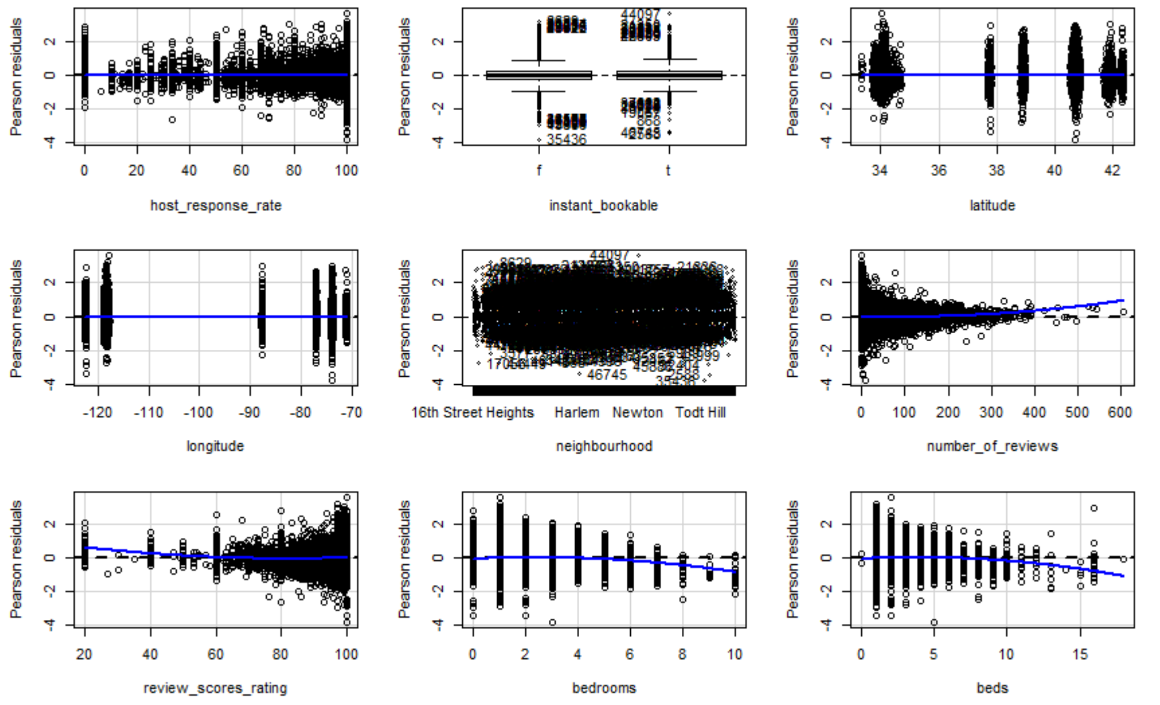
Residual analysis

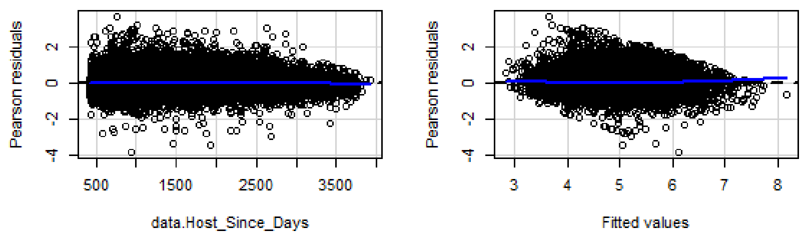
Standard residual vs fitted value

Residual plot vs predictor and fitted values shows that most of the points surrounds the 0 line. There is no specific pattern or change indication of cone.

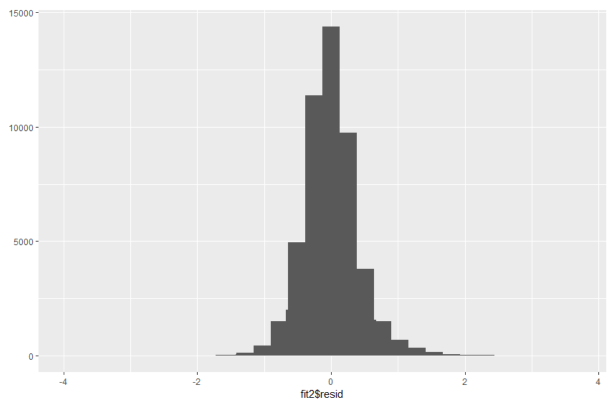


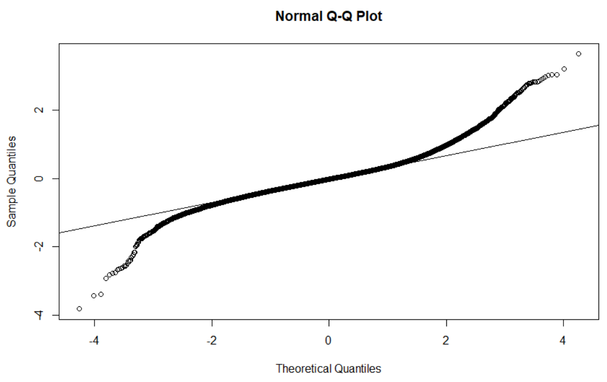

Pearson residuals
property_type
Apartment
Cave
Hostel
Tent
Villa
room_type
Entire home/apt
Shared room
accommodates
bathrooms
bed_type
Airbed
Futon
Real Bed
cancellation_policy
flexible
strict
super_strict_60
city
Boston
DC
LA
NYC
SF
host_has_profile_pic
f
t
host_identity_verified


Pearson residuals
host_response_rate
instant_bookable
f
t
latitude
longitude
neighbourhood
16th Street Heights
Harlem
Newton
Todt Hill
number_of_reviews
review_scores_rating
bedrooms
beds

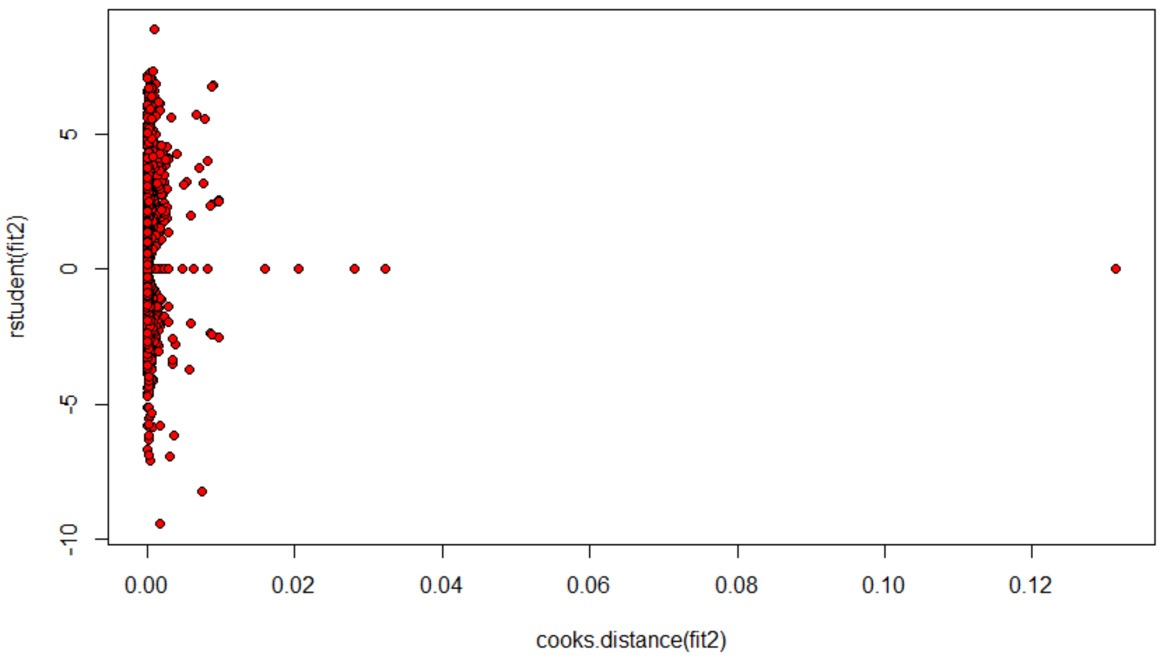


Below histogram shows residuals which looks fairly normally distributed and the residuals follows the linear QQ line.

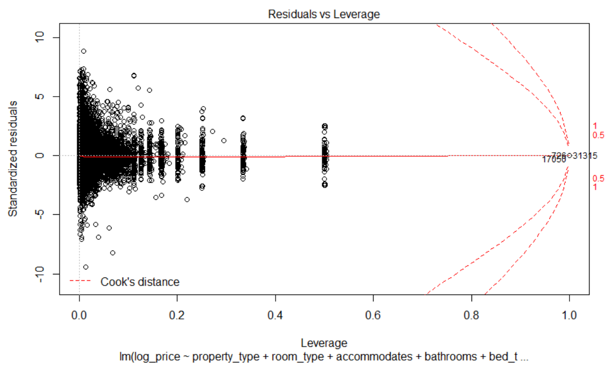


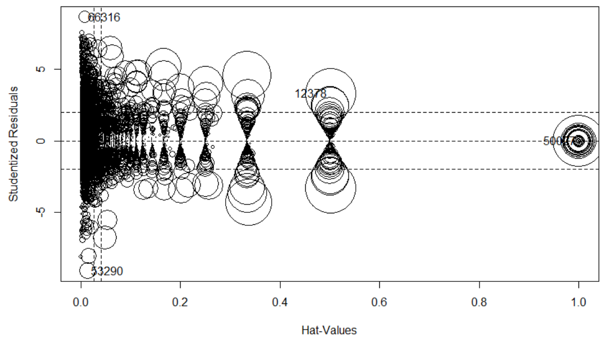


Cook Distance vs Studentized Residual: We have just few observations that have large Cook distance but it small in term of studentized residual value. We also have some studentized residual value that is large but they are small relative to the Cook's distance



Studentized Residual vs Hat values:





On the left side is the Studentized Residual vs Leverage plot none of the point go beyond 0.5 in leverage indicated that we don't have any influence point respect to Cook's Distance. On the right side is the Studentized Residual vs Hat values, we see few data point which might be potentially demarcation for the hat value and have the potential to be studentized residual outliers.

Ridge regression

The second model which has been used is ridge regression. We used the value of lambda as a hyper parameter which has been tuned to get the optimized results. The rmse value obtained in this case is 0.797.

```
> summary(regridge)
            Length Class      Mode
a0            100  -none-     numeric
beta        68100  dgCMatrix  S4
df            100  -none-     numeric
dim             2  -none-     numeric
lambda        100  -none-     numeric
dev.ratio     100  -none-     numeric
nulldev         1  -none-     numeric
npasses         1  -none-     numeric
jerr            1  -none-     numeric
offset          1  -none-     logical
call            5  -none-     call
nobs            1  -none-     numeric
lambdaOpt       1  -none-     numeric
xNames        681  -none-     character
problemType     1  -none-     character
tuneValue       2  data.frame list
obsLevels       1  -none-     logical
param           0  -none-     list


> #Calculating RMSE value
> rmse(trainData$log_price, testingridge)
[1] 0.7972118
```

LASSO

Lasso is used for the another regression model for the dataset. The rmse value we get from this model is~0.71 The summary of the results is:

```
summary(reglasso)
            Length Class      Mode
a0             88  -none-     numeric
beta        59928  dgCMatrix  S4
df             88  -none-     numeric
dim             2  -none-     numeric
lambda         88  -none-     numeric
dev.ratio      88  -none-     numeric
nulldev         1  -none-     numeric
npasses         1  -none-     numeric
jerr            1  -none-     numeric
offset          1  -none-     logical
call            5  -none-     call
nobs            1  -none-     numeric
lambdaOpt       1  -none-     numeric
xNames        681  -none-     character
problemType     1  -none-     character
tuneValue       2  data.frame list
obsLevels       1  -none-     logical
param           0  -none-     list
> #Predicting the log_price
> testinglasso <- predict(reglasso, testData)
> #Calculating RMSE value
> rmse(trainData$log_price, testinglasso)
[1] 0.7172321
```

Elastic net regression

In elastic net the rmse value we get is~0.79. The summary of the results are below:

```
> summary(regelastic)
            Length Class      Mode
a0            100  -none-     numeric
beta        68100  dgCMatrix  S4
df            100  -none-     numeric
dim             2  -none-     numeric
lambda        100  -none-     numeric
dev.ratio     100  -none-     numeric
nulldev         1  -none-     numeric
npasses         1  -none-     numeric
jerr            1  -none-     numeric
offset          1  -none-     logical
call            5  -none-     call
nobs            1  -none-     numeric
lambdaOpt       1  -none-     numeric
xNames        681  -none-     character
problemType     1  -none-     character
tuneValue       2  data.frame list
obsLevels       1  -none-     logical
param           0  -none-     list
> #Predicting the log_price
> testingelastic <- predict(regelastic, testData)
> #Calculating RMSE value
> rmse(trainData$log_price, testingelastic)
[1] 0.7972118
```

PLS regression

We performed PLS regression and looked into the different cases with different principal components. We first ran a default model with max number of PC, the reults are below:

```
> #Performing PLS.
> #using randomly all principal components to model data
> regpls <- plsr(log_price ~ ., data = trainData, method = "oscorespls", vali
dation = "CV")
> summary(regpls)
Data:   X dimension: 49282 681
        Y dimension: 49282 1
Fit method: oscorespls
Number of components considered: 681


#Predicting the log_price.
> testingpls <- predict(regpls, trainData)
> #Calculating RMSE value
> rmse(trainData$log_price, testingpls)
[1] 6.337018e+31
```

Looking at the above results, we see the model uses~ 681 PC and the rmse is not optimized.

```
TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 c
omps  9 comps  10 comps  11 comps  12 comps
X          99.4846   99.773   99.902    99.92
```

We looked at the results and found that only 4 PC explains~ 99.92% variance of the data. So, we ran the model again, this time limiting it to juts 4 PC. The rmse value obtained this time is~0.69. The summary of the results is below:

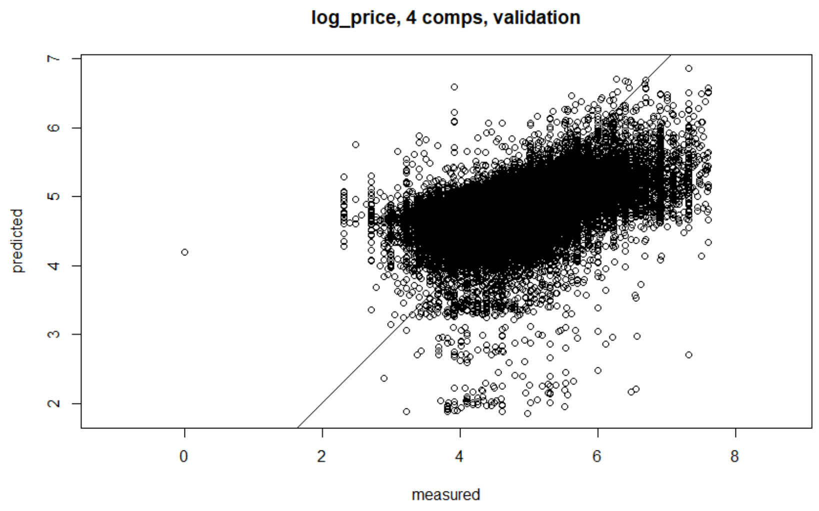
```
Form the plot, we see 4 components are sufficient to explain most of the vari
ance
> regpls1 <- plsr(data =trainData, log_price ~ .,4, method = "oscorespls", va
lidation = "CV")
> summary(regpls1)
Data:   X dimension: 49282 681
        Y dimension: 49282 1
Fit method: oscorespls
Number of components considered: 4

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
```

```
CV          0.7182   0.7161   0.7126    0.703   0.6347
adjCV       0.7182   0.7161   0.7126    0.703   0.6346

TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps
X          99.4846   99.773   99.902    99.92
log_price   0.5881    1.565    4.194    21.98
> plot(regpls, ncomp = 4, asp = 1, line = TRUE)
> plot(regpls, plottype = "scores", comps = 1:3)
> testingpls1 <- predict(regpls1, trainData)
> #Calculating RMSE value
> rmse(trainData$log_price, testingpls1)
[1] 0.6922688
```

From the plot below, we see that most of the points are concentrated along the unit slope line indicating the good model match with the predicted data and measured data. It should be noted that this result is with 4 principle components which explains~99.92 variation in dataset.



Random forest

It is believed that random forest is the most robust model for the classification and regression. However, with such a huge dataset having more than 75,000 variables, running random forest with 1000 trees would take forever with 8 gm Ram we had. Through internet research, we came across the H2O platform. H2O is an open source machine learning platform and they provide GUI platforms to perform faster data computations.

H2O has a feature which connects the R tool of the computer and channelize more memory, processing power for making faster computations and allows computation s to take at 100% CPU capacity. The added advantage it has is; it can also be connected with clusters at cloud platforms for doing faster computations.

The summary of the results is below:

```
=================================================

Variable Importances:
                 variable relative_importance scaled_importance percentage
1               room_type       163653.500000          1.000000   0.265159
2           neighbourhood       140032.093750          0.855662   0.226887
3                bedrooms        88510.429688          0.540840   0.143409
4            accommodates        53588.902344          0.327453   0.086827
5               bathrooms        34448.164062          0.210495   0.055815
6                    beds        31158.919922          0.190396   0.050485
7     data$Host_Since_Days        17787.294922          0.108689   0.028820
8    review_scores_rating        13399.546875          0.081878   0.021711
9           property_type        11993.679688          0.073287   0.019433
10              longitude        10142.878906          0.061978   0.016434
11                   city         9904.601562          0.060522   0.016048
12     host_response_rate         8178.743164          0.049976   0.013252
13      number_of_reviews         7769.519043          0.047475   0.012589
14               latitude         7194.763184          0.043963   0.011657
15    cancellation_policy         6226.384277          0.038046   0.010088
16           cleaning_fee         4318.464355          0.026388   0.006997
17 host_identity_verified         3323.883789          0.020310   0.005386
18       instant_bookable         3191.572754          0.019502   0.005171
19               bed_type         1884.820679          0.011517   0.003054
20   host_has_profile_pic          480.804260          0.002938   0.000779
```

```
H2ORegressionMetrics: drf
** Reported on training data. **
** Metrics reported on Out-Of-Bag training samples **

MSE:  0.172469
RMSE:  0.4152939
MAE:  0.2993402
RMSLE:  0.07115799
Mean Residual Deviance :  0.172469
```

```

H2ORegressionMetrics: drf
** Reported on validation data. **

MSE:  0.1709782
RMSE:  0.4134951
MAE:  0.2976
RMSLE:  0.07102662
Mean Residual Deviance :  0.1709782
```

```
> perf <- h2o.performance(rf1, test)
> perf
H2ORegressionMetrics: drf

MSE:  0.1715643
RMSE:  0.4142032
MAE:  0.2983739
RMSLE:  0.07062328
Mean Residual Deviance :  0.1715643
```

From the results, we see the respective importance of each attributes. As we can expect intuitively, from the airBNB dataset, the attributes like rom type, neighborhood, bedrooms, accommodates, bathrooms, beds are of higher importance as compared to least importance attributes like host_has_profile_pic, bed_type, instant_bookable etc. This shows the model gives a very good work in classification and regression. The rmse we get ~0.41 which is also least of all the models above.

Summary & Conclusions

To summarize, we would say, we looked the dataset in huge details and did several data preparation/cleansing work given the huge volume of dataset we had. Thereafter, we analyzed the efficacy of different models and found the random forest is giving the best results. From the above model, the rmse on train dataset we had is the least which is 0.41. This model can be further used to predict the house price listed at airBnb based on the different attributes which has been used in the model preparation.

Appendix

Tables

Attributes	Description	Type
id	Id of the house	Numeric
log_price	Logarithm of the house price	Numeric
accommodates	number of people can be accomodated	Numeric
bathrooms	number of bathroom	Numeric
host_response_rate	Response rate of the host of the house	Numeric
latitude	Latitude co-ordinate of the location	Numeric
longitude	Longitude co-ordinate of the location	Numeric
number_of_reviews	Number of reviews received by the property	Numeric
review_scores_rating	review score rating of the property	Numeric
zipcode	zipcode of the property location	Numeric
bedrooms	number of bedrooms available	Numeric
beds	number of bed available	Numeric

Table. 01

Attributes	Description	Type
property_type	Type of the property: loft, apartment,townhouse, house, etc.	Categorical
room_type	Entire home, private room	Categorical
amenities	TV, kitchen, wifi, internet	Categorical
bed_type	Type of the bed: real bed, airbed	Categorical
cancellation_policy	Strict, moderate, flexible	Categorical
cleaning_fee	True for yes, false for no	Categorical
city	City located: NYC, LA Chicago etc	Categorical
host_has_profile_pic	True for yes, false for no	Categorical
host_identity_verified	True for yes, false for no	Categorical
instant_bookable	True for yes, false for no	Categorical

Table. 02

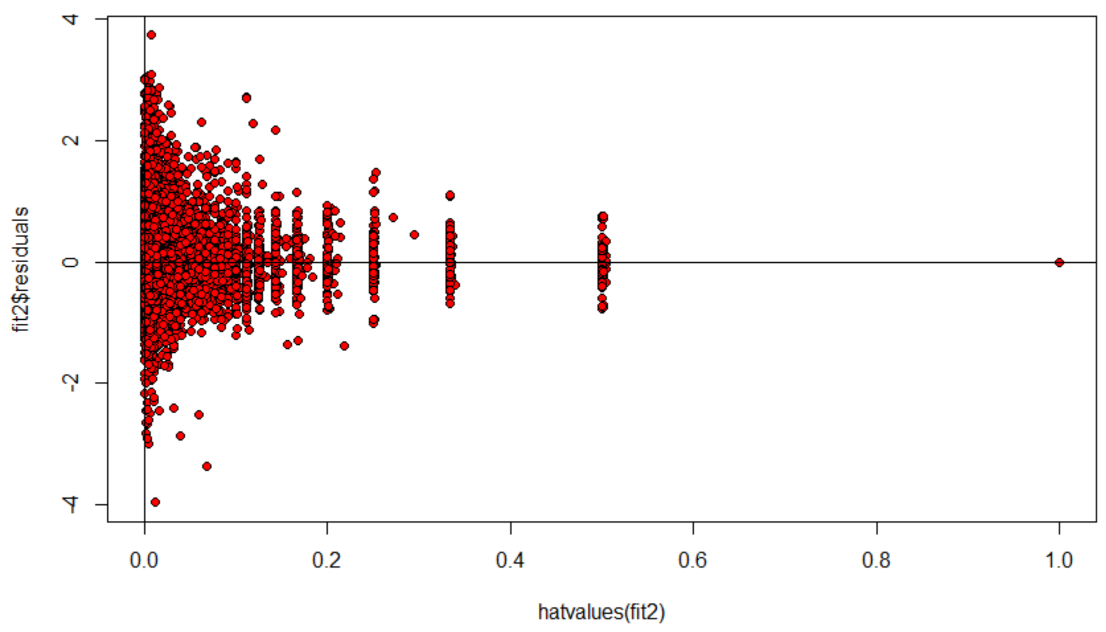
Attributes	Description	Type
description	Description of the house	Text
first_review	When the first review was given	date
host_since	Start date of the host housing the property	date
last_review	Date of last review received	date
name	name of the property	text
neighbourhood	Neighbourhood where the property located	text
thumbnail_url	url to access the property	text

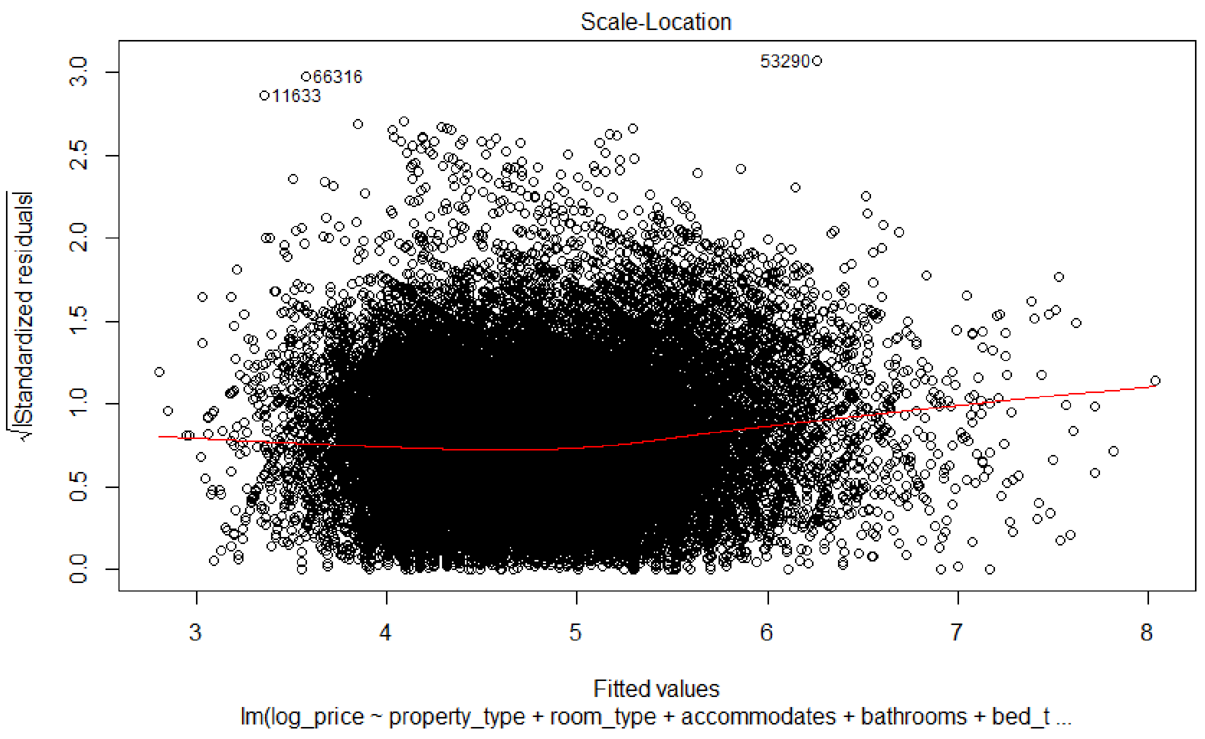
Table. 03

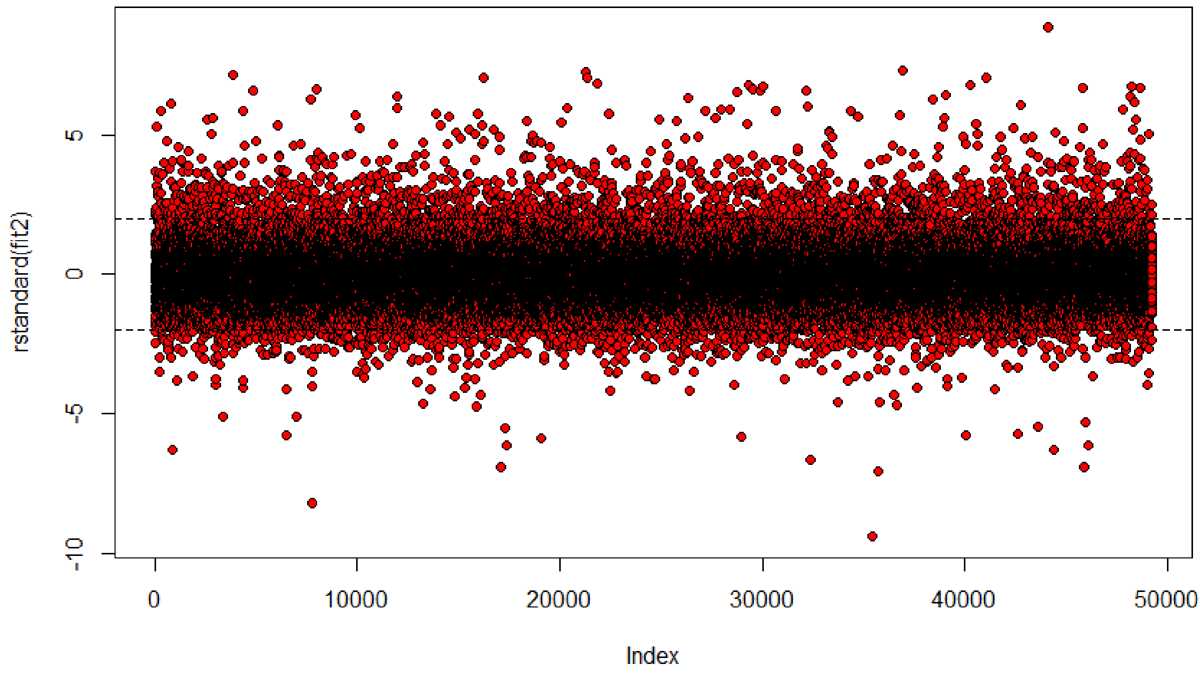
Variable Names	Number of Missing Values
id	0
log_price	0
property_type	0
room_type	0
amenities	0
accommodates	0
bathrooms	200
bed_type	0
cancellation_policy	0
cleaning_fee	0
city	0
description	0
first_review	15864
host_has_profile_pic	188
host_identity_verified	188
host_response_rate	18299
host_since	188
instant_bookable	0
last_review	15827
latitude	0
longitude	0
name	0
neighbourhood	6872
number_of_reviews	0
review_scores_rating	16722
thumbnail_url	8216
zipcode	966
bedrooms	91
beds	131

Table. 04

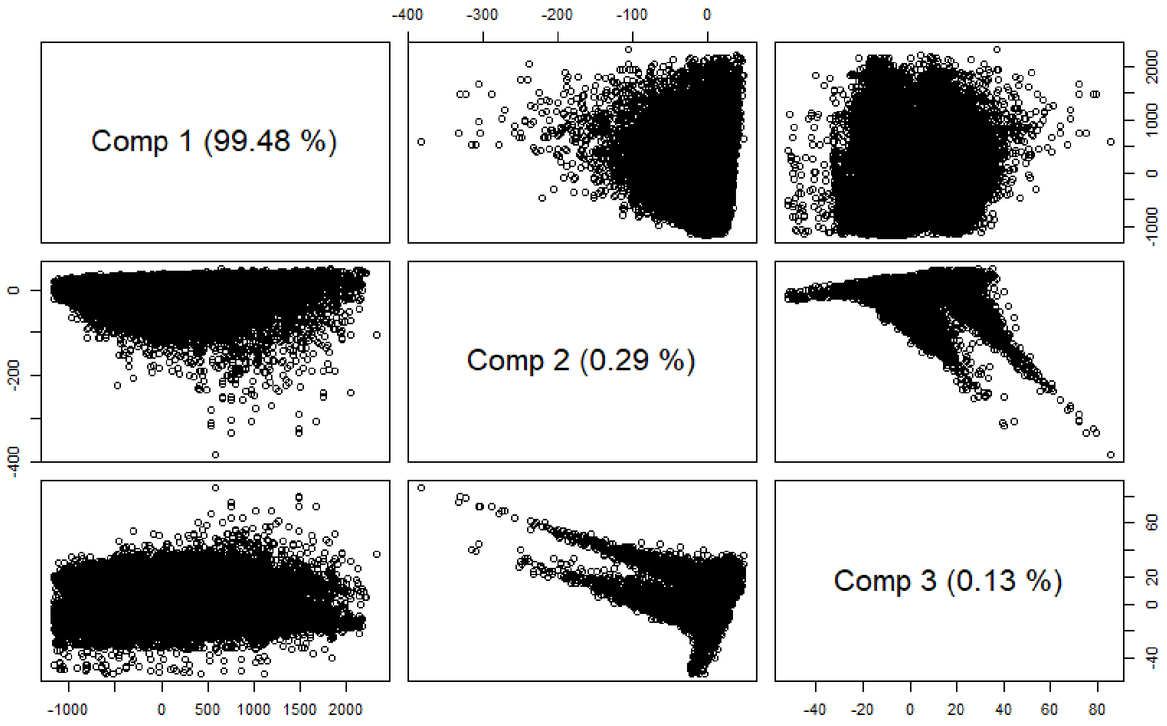
Residual plots







Plots from PLS regression



Detailed summary of Random Forest

```
> summary(rf1)
Model Details:
==============
H2ORegressionModel: drf
Model Key:  rf_covType_v1
Model Summary:
  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_
depth mean_depth min_leaves max_leaves
1              39                       39            15140141        20
20   20.00000      13896      16379
  mean_leaves
1 15301.84600
Scoring History:
            timestamp   duration number_of_trees training_rmse training_mae t
raining_deviance validation_rmse
1 2018-12-14 14:15:44  0.015 sec               0            NA           NA
NA               NA
2 2018-12-14 14:15:45  1.324 sec               1       0.55510      0.39746
0.30813          0.55145
3 2018-12-14 14:15:46  2.151 sec               2       0.53705      0.38599
0.28842          0.48461
4 2018-12-14 14:15:47  2.766 sec               3       0.52171      0.37490
0.27218          0.46116
5 2018-12-14 14:15:47  3.279 sec               4       0.50886      0.36619
0.25894          0.44954
  validation_mae validation_deviance
1             NA                  NA
2        0.39874             0.30410
3        0.35242             0.23485
4        0.33448             0.21267
5        0.32526             0.20208
---
             timestamp   duration number_of_trees training_rmse training_mae
training_deviance validation_rmse
35 2018-12-14 14:16:01 16.618 sec              34       0.41742      0.30108
0.17424          0.41400
36 2018-12-14 14:16:01 17.114 sec              35       0.41695      0.30062
0.17385          0.41383
37 2018-12-14 14:16:02 17.595 sec              36       0.41654      0.30029
0.17351          0.41369
38 2018-12-14 14:16:02 18.102 sec              37       0.41591      0.29991
0.17298          0.41346
39 2018-12-14 14:16:03 18.557 sec              38       0.41563      0.29970
0.17275          0.41344
40 2018-12-14 14:16:03 19.046 sec              39       0.41529      0.29934
0.17247          0.41350
   validation_mae validation_deviance
35        0.29811             0.17140
36        0.29796             0.17125
37        0.29769             0.17114
38        0.29754             0.17095
39        0.29760             0.17094
40        0.29760             0.17098
```