Copyright

by

Shiyao Cai

2017

The Report Committee for Shiyao Cai Certifies that this is the

approved version of the following report:

Predicting rental listing popularity-2 Sigma connect Renthop

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

____
Tim Keitt

____
Paul Robbins

Predicting rental listing popularity-2 Sigma connect Renthop

By

Shiyao Cai, B.S.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of science in Statistics

THE UNIVERSITY OF TEXAS AT AUSTIN

MAY 2017

Abstract

Renting a perfect apartment can be a hassle. There are plenty of features people care about when it comes to finding the apartment, such as price, hardwood floor, dog park, laundry room, etc. Being able to predict people's interest level on an apartment will help the rental agency better handle fraud control, identify potential listing quality issues, and allow owners and agents to understand renters' needs and preferences.

RentHop, an apartment search engine, along with 2 Sigma, introduced this multiple classification problem in the Kaggle community. It provides the opportunity to use owners' data to predict the interest level of their apartments on its website.

This report attempts to find a pattern of people's interest level towards rental listing on the website using the dataset from the Kaggle competition. Multiple features are derived from the original dataset. Several common data mining and machine learning techniques are used to improve the accuracy of the predicting model. The final result is evaluated using Log loss function.

Table of Contents

List of Tables

List of Figures

Introduction

Data driven solutions have become important to many businesses. For instance, companies use costumers' data to recommend new products, cast specific advertisement and detect fraud. This paper will focus on using statistical models and data mining methods to predict the level of interest of customers towards a rental listing online. Being able to predict people's interest level on an apartment will help the rental agency better handle fraud control, identify potential listing quality issues, and allow owners and agents to understand renters' needs and preferences.

Kaggle is a community to solve the most interesting and sensitive business problems using machine learning and data mining for large business corporations. RentHop, an apartment search engine, introduced this multiple classification problem in the Kaggle community. It provides the opportunity to use owners' data to predict the interest level of their apartments on its website.

This project aims to predict rental listings' interest levels using the data provided in the competition. There are 3 possible outcomes for the interest levels, namely 'high', 'medium' and 'low'. Therefore, it is a multiclass classification problem. Several features are provided to develop machine learning models including price, number of bedroom, number of bathroom, time the listing created, manager ID, description of the apartment, photos, longitude, latitude, displayed address on the website, the actual detailed address, key features about the apartment, etc. The feature names and corresponding types are provided in Table 1.

Table 1. Features in the dataset

Feature name	Feature type
bathrooms	Numerical
Bedrooms	Numerical
Building ID	Categorical
Created	Numerical
Description	Text
Display Address	Text
Features	Text
Latitude	Numerical
Longitude	Numerical
Manager ID	Categorical
Photos	Image
Price	Numerical
Street Address	Text
Interested Level	Categorical

CHAPTER 1: Data Exploration

1. Interest Level

The variable interest level is the variable we are going to predict. Examining this feature closely will help us determine the prediction baseline. From Figure 1. Distribution of Interest Level***Figure 1***, it is noted that the interest level 'low' takes 69% of all the interest levels, while the interest level 'high' only takes 7.8%. This indicates that most of the apartments in the website are of low interest for users.Only a few apartments that really draw high attention from the customers.

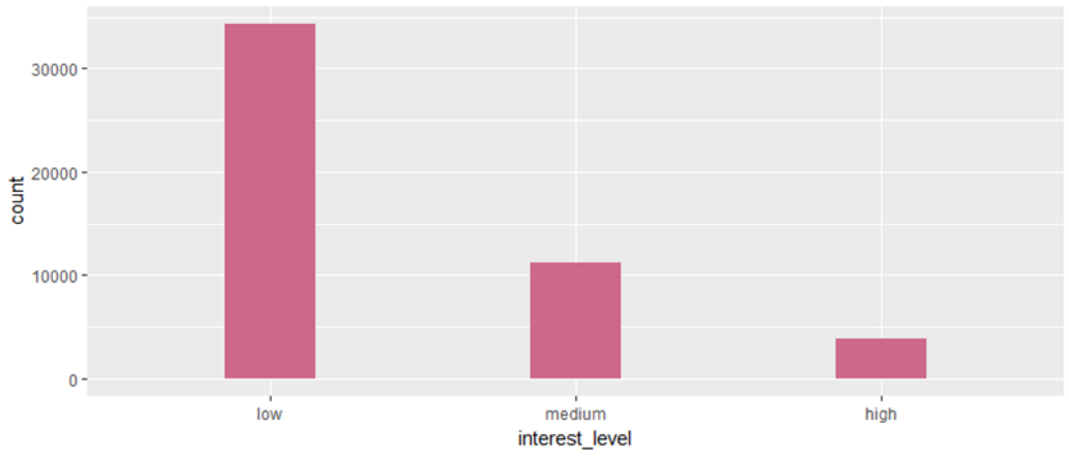


Figure 1. Distribution of Interest Level

2. Price

The price of an apartment in the website represents the rental price the tenants need to pay per month.   This is an important indicator of the interest level for potential customers. Analyzing the price distribution of the dataset will give us an overview of the price in the data set.

The distribution of price with outliers removed is shown in Figure 2.Distribution ***of Listing Price with outliers removed*** The outliers in the distribution have prices larger than 20000 USD per month. From the distribution, the average price for renting an apartment is around 2700 per month. It has a long right tail, which indicates that there are several cases the prices for renting is very large. Most data points fall in the range 2000 to 8000 USD per month.

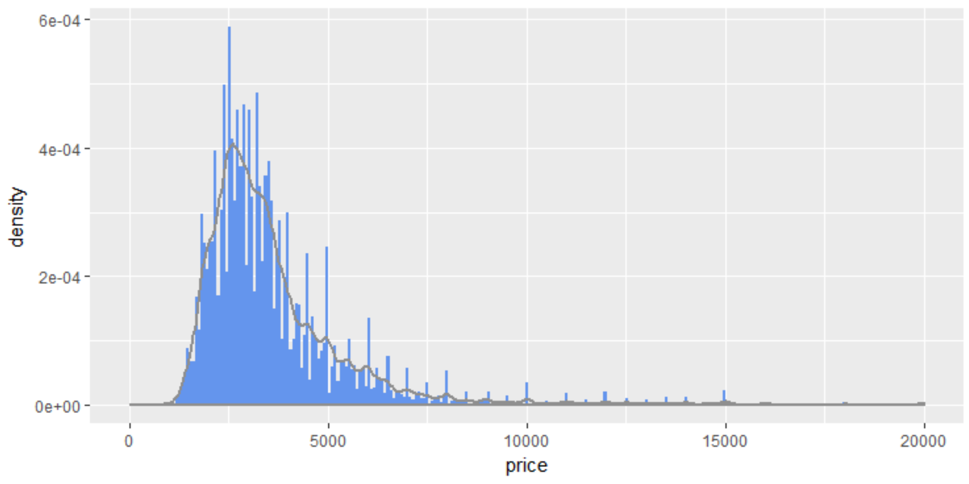


Figure 2.Distribution of Listing Price with outliers removed

By plotting the prices across different interest levels, we can see if prices and interest levels are correlated. To validate their correlation, an ANOVA test is conducted to see if there is a significant effect between prices and interest level.

From Figure 3. ***Listing Prices across Interest Level***, we can see the price distributions are different for different interest levels. For 'low' Interest level, the mean and median price are apparently higher than that of 'high' Interest level. The distribution indicates people have high interest towards listings that have lower prices. To find out whether the prices are truly different across different interest levels, ANOVA test is conducted.

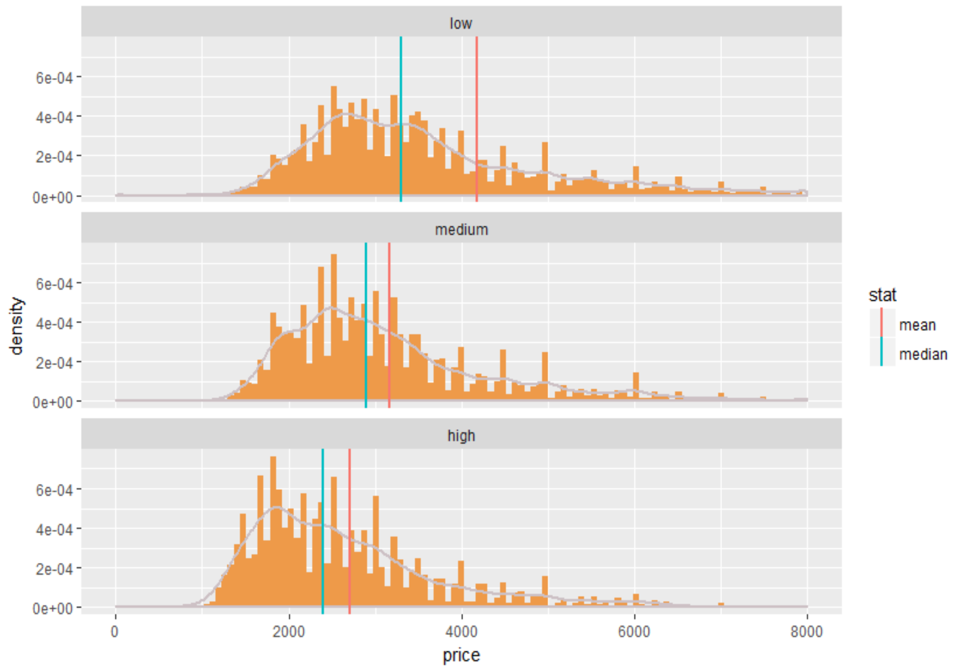


Figure 3. Listing Prices across Interest Level

A one-way ANOVA was conducted to compare the effect of Interest Level on prices in 'low', 'medium', 'high' interest level. The result of ANOVA test is shown in Table 2. There is a significant effect of Interest Level on prices at the $p < .05$ level for the three conditions $[F(2, 49349) \; = \; 14.46, p = 5.26 * e - 07]$.

Table 2.   One-way ANOVA result

	Sum of squares	DF	Mean Square	F	Sig.
Between Groups	1.4e+10	2	7.039e+09	14.46	5.26e-07
Within Groups	2.4e+13	49349	4.867e+08		
Total	2.4e+13	49351			

This result confirms the assumption that apartment with lower prices will be of higher interest to the customers browsing the website.

3. Longitude & Latitude

The dataset contains the longitude and latitude for every apartment on the website. Specifically, the apartments are all from New York city. By plotting longitude and latitude of apartments, we can get the location information of the apartments in the dataset. From Figure 4. Interest levels on locations, we can see that most apartments of low interest are gathered around the center of the figure, while some apartments of high interest are spread with some of them located on the outside of the center.

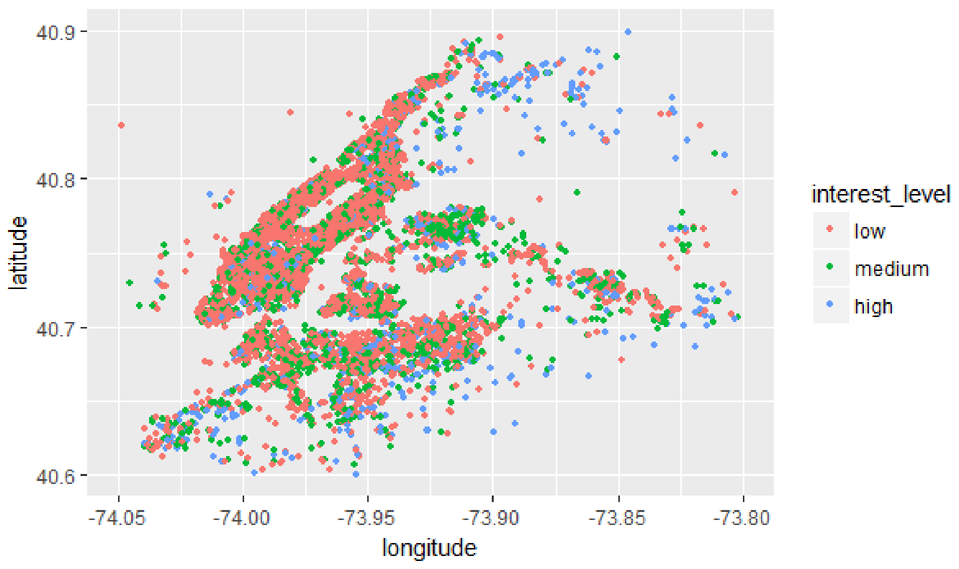


Figure 4. Interest levels on locations

4. Number of Bedroom

In this dataset, the number of bedroom indicates how many bedrooms an apartment has. The distribution of number of bedroom is plotted in Figure 5. It shows the patterns of number of bedroom is generally consistent across different interest levels. There are more apartments with 1 or 2 bedrooms for low interest listings. For listing of high interest, the largest group are the apartments with 2 bedrooms.

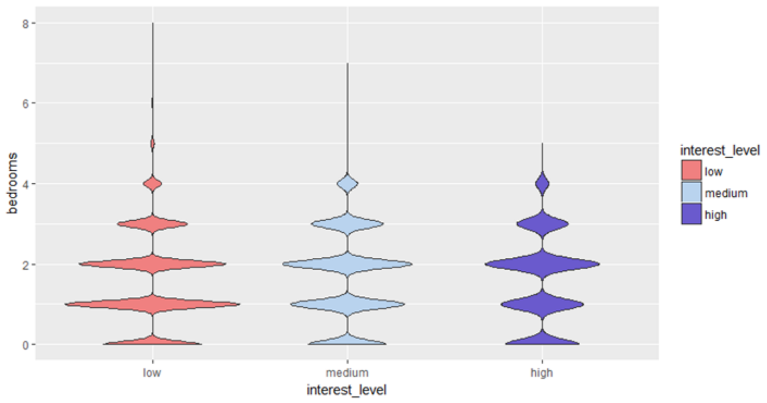


Figure 5.Number of bedrooms across Interest Level

5. Number of Bathroom

By plotting number of bathroom across different interest levels, we can see that most apartments of high interest contain 1 bathroom. The bathroom numbers of apartment of low and medium interest contains a larger range of numbers of bathroom. This correspond to the finding that apartments with high interest are generally in lower price, which excluded fancy apartment with multiple bathrooms.

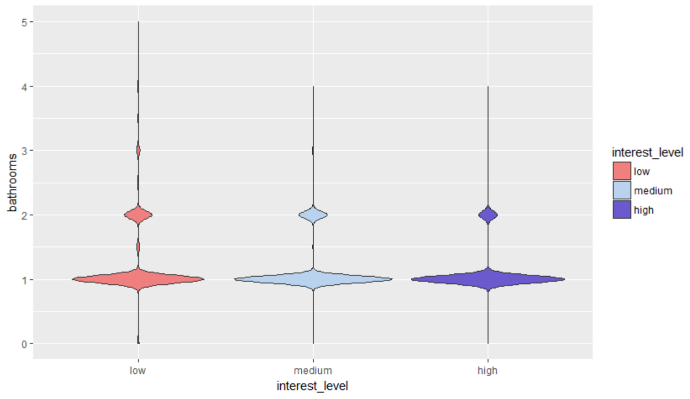


Figure 6.Number of bathrooms across Interest Level

CHAPTER 2: Feature Engineering

1. High-cardinality Features

High cardinality refers to the features that contain a large amount of categorical values. Typical examples of high cardinality features include email address, zip code, ID, etc. There are 2 features that are of high cardinality, manger ID and building ID. The following table states the number of unique values for each variable.

Table 3. The number of unique values in Manager ID and Building ID

	Manager ID	Building ID
Number of unique values	*3481*	*7585*
Proportion of unique values	*7.05%*	*15.37%*

From Table 3, we can see that the cardinality of each feature is not that high. These features are informative, since probably some managers are good at management that the rental listing he/she manages intrigue more interest from customers. Or some buildings are at good locations that appealing to customers.

There are several common methods to deal with high cardinality features, including one-hot encoding, semantic grouping and supervised ratio.

a. One hot encoding/Dummy coding

One hot encoding or dummy coding is a common method to deal with nominal data. The method transforms the original feature with N categories into N binary columns. These new columns contain either 0 or 1, with 0 means there is no such category in the original feature, and vice versa. This method preserves the information of original feature, but it adds more dimensions to the original dataset.

b. Semantic Grouping

The aim of semantic grouping is to identify logical groups from high cardinality data. One benefit is that it reduces the number of unique values from the feature. In this dataset, semantic grouping can be applied to the longitude and latitude features. This will be included in the Geospatial analysis in the later section.

c. Supervised ratio

This technique will use the target outcome, the variable we are trying to predict, along with the high cardinality feature to calculate new continuous features. Since the new variables contain the information from outcome variable, it is considerably easy to overfit the data.

$$OP_i = \frac{A_i}{A_i + B_i + C_i}$$

One assumption made when using this technique is that rental listing with the same manager will have similar percentage of outcome.

Table 4. Example of Supervised Ratio

Manager ID	Manager Level Low	Manager Level Medium	Manager Level High
565	0.361	0.397	0.241
2817	0.44	0.52	0.04
3959	0.679	0.251	0.069
4237	0.88	0.11	0
2054	0.98	0.019	0

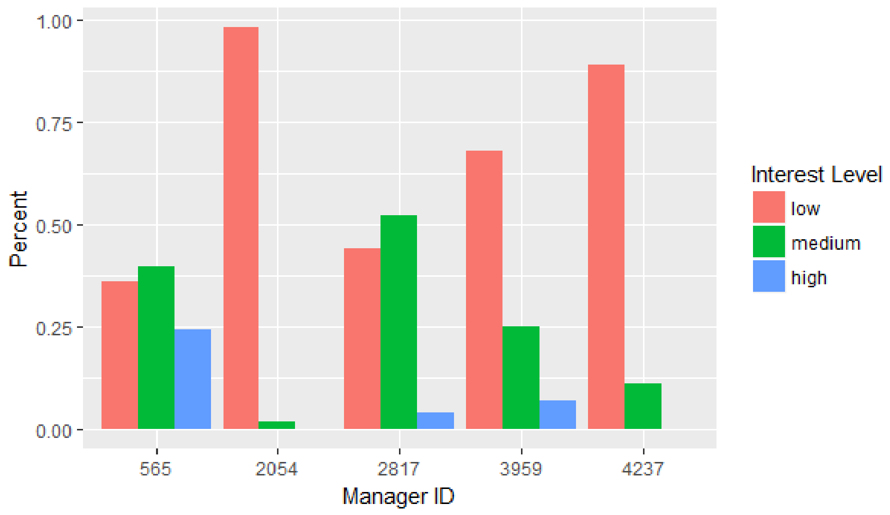


Figure 7. Interest Level ratio of 5 managers

2. Geospatial features

There are two features in our dataset, longitude and latitude, that cannot be directly used in our model since its actual meaning is beyond these numerical variables. Since the dataset contains rental listings only in New York, one way to deal with it is to use neighborhood information extracted from longitude and longitude. 7 categories are derived from the original longitude and latitude features: Uptown, Queens, Midtown, Kings, Downtown, Between midtown and uptown, others. Dummy encoding is used to indicate which neighborhood each apartment listing belongs to. We are going to use the new derived features about district instead of latitude and longitude in the model.

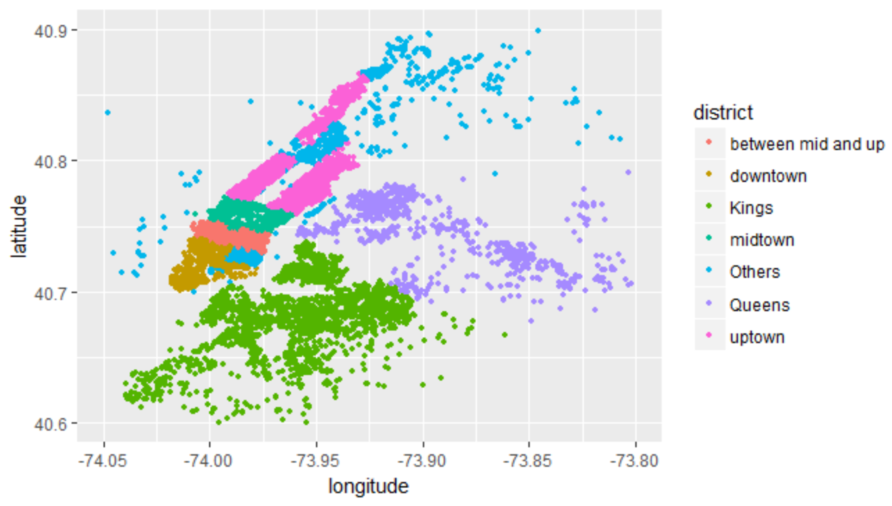


Figure 8. New features district on the map

Table 5. New Neighborhood features derived from latitude and longitude

Listing ID	Uptown	Queens	Midtown	Kings	Downtown	Between midtown and uptown
7170325	0	0	0	1	0	0
7092344	0	0	1	0	0	0
7158677	0	0	1	0	0	0
7140668	0	0	0	0	1	0
7126989	0	0	0	0	0	0

3. Difference Between Address

One interesting thing in the dataset is that there are 2 features represent address of the listing apartments. One is displayed address, the address shown on the website. The other one is street address, which contains the actual detailed address of the listing apartments. Since both addresses are provided, a variable describing their difference is created and used as a new feature in the model. The technique used to describe their difference between is Levenshtein distance. Mathematically, the Levenshtein distance between two strings a, b is given by $Lev(a, b)$,where

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

The new feature 'similarity' captures the difference between street address and display address, which is shown in Table 6.

Table 6. Similarity Features Derived from Street Address and Display Address

Street address	Display address	Similarity
1661 york avenue	york avenue	0.6875000
410 east 13th street	east 13th street	0.8000000
170 east 18th street	18th street	0.550000
145 borinquen place	145 borinquen place	1.000000

4. Sentimental Analysis

Sentiment analysis is a method using natural language processing to identify whether a piece of information is positive, negative or neutral. Since the feature "Description" in the data set contains very rich information about the apartment, sentiment analysis is used to evaluate whether description about the apartment on listing is positive or negative.

The sentiment analysis showed that in the dataset, anticipation, joy, and trust take the majority portion of the whole sentiment result. This aligns with the fact that the description for the apartment is generally positive and aims to be appealing to customers.

Table 7. Sentimental Features Derived from Description

Llisting ID	anticipation	disgust	fear	joy	sadness	surprise	trust	anger
7170325	0	0	0	1	0	0	0	1
7092344	2	1	1	4	0	1	6	0
7158677	2	0	0	2	0	0	1	0
7211212	3	0	0	1	0	2	4	0
7225292	2	0	1	2	2	0	1	0

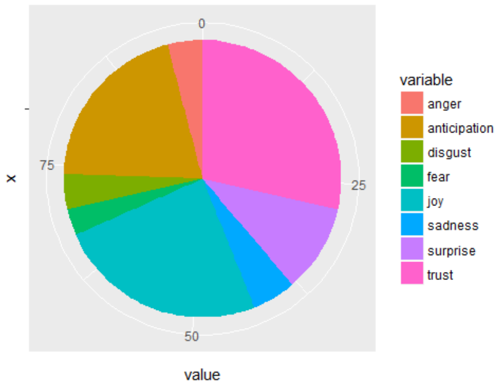


Figure 9. Pie chart of sentiment features

5. Basic numerical feature

In addition to the methods used above, some basic features are derived from the original dataset. Features Photos, Features, Descriptions contain numbers of items, so new features derived from calculating how many items are there in each feature. 3 features, namely number of photos, number of features, number of description words, are calculated from the original ones to describe the number of things in one listing.

Another feature derived is the price per room. It is calculated using the equation:

$$Price\ \ per\ \ room = \frac{\text{Price}}{\text{Number of bathrooms} + \text{Number of bedrooms}}$$

This feature captures the price for a listing per room. It is probably a better way to describe the value customers are going to get by paying certain price. The newly derived features are shown in Table 8. Basic numerical features

Table 8. Basic numerical features

Listing ID	Price per room	Number of photos	Number of features	Number of Description words
7170325	2400	12	7	77
7092344	1900	6	6	131
7158677	1747.5	6	6	119
7211212	1000	5	0	95
7225292	Inf	4	4	41

CHAPTER 3: Modeling

1. Random forest

Random forest is a popular ensemble methods of bootstrap aggregating(bagging). It grows many classification trees and chooses the classification having the most votes to be the final prediction, which eliminates the overfitting issue of decision trees. The algorithm contains 2 major part, tree bagging and feature bagging.

Random forest builds each tree by taking a subsample with replacement from the original dataset. Suppose there are N samples from the original dataset, random forest will take M samples from N original data with replacement. It will them use these M samples to build one decision tree. And it will continue to do it until it has a forest.

This bootstrapping method will decrease the variance of the model without increasing bias. One single decision is very likely to be overfitting due to the noise from the dataset, while having multiple trees will eliminate this issue.

To reduce the noise, it is optimal to have trees that are not correlated. If there are several features that are strong predictors of the outcome, many trees will choose them as the input variable. This will cause trees to be highly correlated and will not reduce the noise from the original data. To solve this problem, random forest will select a subset of features when building each tree, this method is known as feature bagging.

2. Gradient boosting

a. Introduction

There are 3 common ensemble methods in machine learning: Bagging, Boosting, Stacking. Bagging, stands for ‘Bootstrap Aggregating’, is an algorithm creating multiple models using sub-samples from the original dataset. Boosting is a method that combine weak models into a strong one. Stacking is a way that use the predicted values from several models as the training set to train a new better performing model using new machine learning methods.

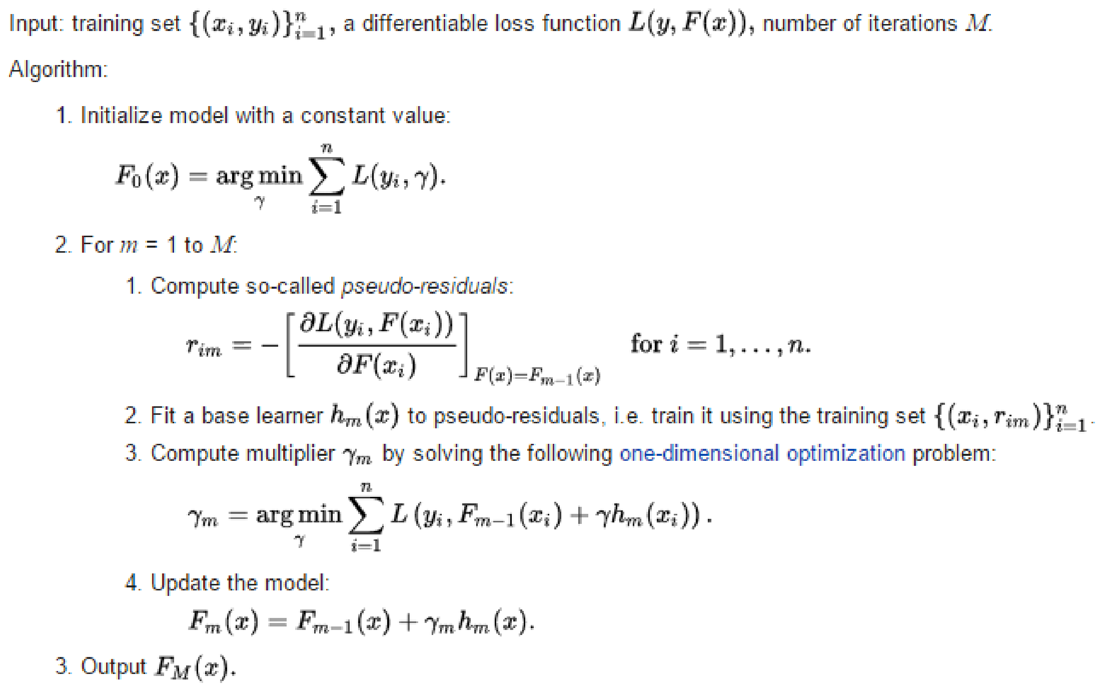
Gradient boosting is an algorithm in combination of gradient descent and boosting. Gradient descent is an algorithm to find a local minimum of a function by taking steps proportional to the negative of the gradient of the function at the current point. Boosting is a family of machine learning algorithms that combine weak learners to a strong one. This family of algorithm can reduce bias and variance at the same time in supervised learning.

Gradient boosting optimizes a cost function over function space by iteratively choosing a function that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

That means, within each iteration, gradient boosting introduces a weak learner to compensate the shortcomings of existing weak learners, hence, dramatically improved the model performance. It is an efficient algorithm to solve regression, classification, and ranking problems. For this classification problem, the gradient boosting tree(GBT) model is used for prediction.

b. Implementation

The algorithm trains weak learners sequentially. It initiates constant values as prediction of y_i, which is apparently a weak learner. After initiation, within each iteration, instead of prediction the outcome y_i, each weak learner is trained to predict r_i , which is called pseudo-residual, meaning the residual from last iteration. Each weak learner will be added to the original model to reduce the bias. The weight of the weak learners is calculated by solving a one-dimensional optimization problem.



```
Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.
Algorithm:
    1. Initialize model with a constant value:
        $$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$
    2. For $m$ = 1 to $M$:
        1. Compute so-called pseudo-residuals:
            $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$
        2. Fit a base learner $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
        3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:
            $$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$
        4. Update the model:
            $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$
    3. Output $F_M(x)$.
```

Figure 10. Gradient boosting algorithm

c. XGBoost

Among the machine learning methods used in practice, XGBoosting is the one that shines in many data science challenges. This famous open source implementation of GBT is introduced by Tianqi Chen from Washington University. The success of the XGBoost is due to its scalability in all scenarios. The scalability of XGBoost is the result from several systems and algorithm optimizations. It enables data scientists to process millions of records on a desktop. In this report, XGBoost is used to predict the interest level of customers towards rental listing online.

CHAPTER 4: Evaluation

To evaluation the final predications, this Kaggle competition will use log loss to measure the final predictions. This measurement metric the same of negative the log likelihood of each prediction.

$$Logloss = -\frac{1}{N}\sum_{i=0}^{N}\sum_{i=0}^{m}\sum y_{i,j}\log(p_{i,j})$$

In the equation, N is the total observations of the dataset. M is the number of class labels. $y_{i,j}$ is 1 when the observation i is in class j, is 0 otherwise. $p_{i,j}$ is the probability of prediction that this observation i is in class j.

Another measurement used to test models' performance is accuracy. Accuracy is calculated by the number of correctly predicted observations over the total number of observations in the dataset.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

As for the baseline of accuracy, the percentage of class 'low' will be used since it is the most frequent class in the dataset. The baseline of Logloss will be calculated by predicting every class using the same probability. In this case, the number of classes we are predicting is 3. Thus the baseline is calculated using the Log loss equation:

$$Baseline\ \ of\ \ logloss = -\ln\left(\frac{1}{3}\right) = \ln 3 = 1.098$$

After getting the baseline of the prediction, random forest and XGBoost are applied to training set and validation set. In this case, cross validation with 5 folds is used to reduce any noise in measurement. From the result, XGBoost outperforms Random forest, achieving better accuracy and lower logloss. This result may be due to the difference between the way of generating trees in random forest and XGBoost. Random forest generates trees by select subset of features and samples from the original data, while

XGBoost focus on training weak trees to predict error. The final prediction is generated using XGBoost, resulting top 20% in the leader board of the competition.

Table 9. Model Evaluation

	Baseline	Random Forest	XGBoost
Log-loss (training)	1.098	0.560	0.335
Log-loss (validation)	1.098	0.614	0.524
Accuracy (training)	0.694	0.794	0.887
Accuracy (validation)	0.697	0.736	0.765

Appendix

1. R code: Data warehouse

```
options(continue = "    ")
#loading packages
   packages = c("jsonlite", "dplyr", "purrr")
   library(syuzhet)
   library(DT)
   purrr::walk(packages, library, character.only = TRUE, warn.conflicts = FALSE)

   suppressMessages(library("jsonlite"))
   suppressMessages(library("dplyr"))
   suppressMessages(library("plotly"))
   suppressMessages(library("purrr"))
   suppressMessages(library("RecordLinkage"))

   #Loading packages for further analysis of the data.
   libs <- c("lubridate", "nnet")
   lapply(libs, require,    character.only = T)
   library(mice)

#load the data

dataware=function(t){
    setwd("C:/Users/ellie/Dropbox/2 sigma/input")
    if(t=='train'){ mydata = fromJSON("train.json")}
    if(t=='test'){mydata = fromJSON("test.json")}
    vars <- setdiff(names(mydata), c("photos", "features"))
    mydata = map_at(mydata, vars, unlist) %>% tibble::as_tibble(.)
    mydata$id=seq(1:length(mydata$building_id)) #numerical ids!
    return(mydata)
}

mytrain=dataware('train')
mytest=dataware('test')



#**********************************
    #Feature------extract
#**************************************

features_extract=function(data){
    #calculate the similarity
    vec.addressSimilarity                                                          <-
        levenshteinSim(tolower(data$street_address),tolower(data$display_address))

    data$sim=vec.addressSimilarity
    #sentimental analysis
    sentiment <- get_nrc_sentiment(data$description)
    datatable(head(sentiment))
    sentiment$id<-seq(1:nrow(sentiment))
    data<-merge(data,sentiment, by.x="id", by.y="id", all.x=T, all.y=T)

    #add other basic features
    data$num_room=data$bathrooms+data$bedrooms
    data$price_per_bedroom=data$price/data$bedrooms
    data$price_per_bathroom=data$price/data$bathroom
    data$price_per_room=data$price/data$num_room

    data$num_features=lapply(data$features, function(x)length(unlist(x)) )
    data$num_photos=lapply(data$photos, function(x)length(unlist(x)) )
    data$num_word_desciption=lapply(data$description,                    function(x)
        length(strsplit(gsub(' {2,}','',x),' ')[[1]]) )

    data$date_time = strptime(data$created, format = "%Y-%m-%d %H:%M:%S")

    data$data_month = month(data$date_time)
    data$data_hour = hour(data$date_time)

    return(data)
}
```

```
mytrain=features_extract(mytrain)

mytest=features_extract(mytest)



#**********************************

#central park

#**************************************


library(geosphere)

for(i in 1:nrow(mytrain)){

  mytrain$central[i]=distm(c(mytrain$longitude[i], mytrain$latitude[i]),

                         c(-73.9654, 40.7829), fun = distHaversine)

}


for(i in 1:nrow(mytest)){

  mytest$central[i]=distm(c(mytest$longitude[i], mytest$latitude[i]),

                        c(-73.9654, 40.7829), fun = distHaversine)

}


#**********************************

#write the data

#**************************************
```

```
#colnames(mytrain)

#mytrain> mytrain[order(mytrain$id),]

#mytest> mytest[order(mytest$id),]


mytrain$date_time=as.character(mytrain$date_time)

mytest$date_time=as.character(mytest$date_time)



mytrain<- apply(mytrain,2,as.character)

mytest<- apply(mytest,2,as.character)




setwd("C:/Users/ellie/Dropbox/2 sigma/feature_basic")

write.csv(mytrain, file = "train_ready_feature1_basic.csv",row.names=FALSE)

write.csv(mytest, file = "test_ready_feature1_basic.csv",row.names=FALSE)
```

2. R code: Data visualization

```
#Loading packages for further analysis of the data.

libs <- c("lubridate", "nnet")

lapply(libs, require,    character.only = T)

library(mice)

library(ggplot2)

library(plyr)

library(e1071)

library(reshape2)


#load the data


dataware=function(t){

   setwd("C:/Users/ellie/Dropbox/master report/data exploration")

   if(t=='train'){ mydata = fromJSON("train.json")}

   if(t=='test'){mydata = fromJSON("test.json")}

   vars <- setdiff(names(mydata), c("photos", "features"))

   mydata = map_at(mydata, vars, unlist) %>% tibble::as_tibble(.)

   mydata$id=seq(1:length(mydata$building_id)) #numerical ids!

   return(mydata)

}


mytrain=dataware('train')
```

```
mytest=dataware('test')


mytrain=as.data.frame(mytrain)

mytest=as.data.frame(mytest)



#Then turn it back into an ordered factor

mytrain$interest_level<- as.character(mytrain$interest_level)

mytrain$interest_level<-    factor(mytrain$interest_level,    levels=c("low",    "medium",
        "high"))

#**********************************

#Plotting

#**************************************


ggplot(data=mytrain,

           aes(interest_level))+

   geom_bar( stat="count" , width = 0.3,fill = "palevioletred3")


#table(mytrain$interest_level)




ggplot(data=mytrain,

           aes(mytrain$price))+

   geom_histogram( breaks=seq(0,20000, by=50),fill = "steelblue3")
```

```
ggplot(data=mytrain, aes(x=price, y=..density..)) +
   geom_histogram(binwidth=70,    fill='cornflowerblue')+
   geom_density(size=1, colour='gray56')+
   xlim(0,20000)




#*********************************
#3 pricing distribution
#*************************************
   df_vline$stat <- rep(c("mean", "median"), each = nrow(df_vline) / 2)
colnames(df_vline)=c('interest_level','x','stat')


mm=    ggplot(data=mytrain, aes(x=price, y=..density..)) +
   geom_histogram(binwidth=70, fill='tan2')+
   geom_density(size=1, colour='lavenderblush3')+
   geom_vline(data=df_vline, mapping=aes(xintercept=x, colour = stat),
                      linetype = 1, size=1.5, show.legend = T)+
   xlim(0,8000)+    facet_wrap(~ as.factor(interest_level),nrow = 3)


print(mm)
```

```
an=lm(price~interest_level,data=mytrain)

summary(an)

#**********************************

# Bedroom and bathroom

#**************************************



ggplot(mytrain, aes(x = interest_level, y = bedrooms)) +

  geom_violin(aes(fill = interest_level)) +

  scale_fill_manual(values=c("lightcoral", "slategray2",'slateblue'))


 #scale_fill_brewer(palette="RdYlGn")


ggplot(mytrain, aes(x = interest_level, y = bathrooms)) +

  geom_violin(aes(fill = interest_level)) +

  scale_fill_manual(values=c("lightcoral", "slategray2",'slateblue'))+

ylim(0,5)


#**********************************

# latitude and longitude

#**************************************


ggplot(data=mytrain) +

  geom_point(size = 1,
                    aes(x = longitude,
                        y = latitude,
                        color = interest_level)) +
xlim(-74.05, -73.8) +ylim(40.6, 40.9)



ggplot(data = mytrain) +
    geom_point(aes(x = longitude, y = latitude,
                        color = interest_level), alpha = 0.2)



#*********************************
# onehot encoding graphs
#*************************************


setwd("C:/Users/ellie/Dropbox/master report/data exploration")

mydata=read.csv('READY_TO_USE_train.csv')
mydata1=mydata[order(-mydata$manager_count),]

myd=subset(mydata,select=c(manager_level_low,manager_level_medium,
   manager_level_high, manager_count,manager_id))




ll=rbind(mydata1[6222,],mydata1[29540,],mydata1[123,],mydata1[40000,],mydata1[10
        000,])


library(reshape2)


dfm                                                                            <-
        melt(ll[,c('manager_id','manager_level_low','manager_level_medium','manager_
        level_high')],
                id.vars = 1)


ggplot(dfm , aes(as.factor(manager_id),    value)) +
   geom_bar(aes(fill=variable), position = "dodge", stat="identity")+
   labs(x="Manager ID", y="Percent")+
   scale_fill_discrete("Interest Level",
                              labels=c("low", "medium",'high'))
```

3. R code: Modeling

```
library(xgboost)

library(stringr)

library(caret)

library(car)

library(mice)

library(data.table)

library(readr)

library(data.table)

library(xgboost)

library(caret)

library(stringr)

library(lubridate)

library(stringr)

library(Hmisc)

library(dplyr)

library(Matrix)


#***************

#read data

#***************


setwd("C:/Users/ellie/Dropbox/2 sigma/TRY24 EVERYTHING")
```

```
train_id=read.csv("train_id.csv")
test_id=read.csv("test_id.csv")

train_ready=read.csv("train_ready.csv")
test_ready=read.csv("test_ready.csv")
md.pattern(train_ready)

train_ready$manager_level_high[is.na(train_ready$manager_level_high)]=
  mean(train_ready$manager_level_high,na.rm=TRUE)
mean(train_ready$manager_level_medium,na.rm=TRUE)
mean(train_ready$manager_level_low,na.rm=TRUE)



mana_fun=function(d){
  d$manager_level_high[is.na(d$manager_level_high)]=
    mean(d$manager_level_high,na.rm=TRUE)
  d$manager_level_medium[is.na(d$manager_level_medium)]=
    mean(d$manager_level_medium,na.rm=TRUE)
  d$manager_level_low[is.na(d$manager_level_low)]=
    mean(d$manager_level_low,na.rm=TRUE)
  return(d)
}
```

```
train_ready=mana_fun(train_ready)

test_ready=mana_fun(test_ready)


#***************

#merge data

#***************


time_stamp=read.csv('listing_image_time.csv')

colnames(time_stamp)=c("listing_id",'time_stamp')


train_ready=merge(train_ready,time_stamp,by="listing_id",all.x = TRUE)

test_ready=merge(test_ready,time_stamp,by="listing_id",all.x = TRUE)


datatrain=merge(train_ready,train_id,by="listing_id",all.x = TRUE)

datatest=merge(test_ready,test_id,by="listing_id",all.x = TRUE)

#--------

datatest=datatest[order(datatest$id),]

datatrain=datatrain[order(datatrain$id),]



trainlabel=read.csv('train_label.csv')

trainlabel$id=NULL

datatrain=merge(datatrain,trainlabel,by="listing_id",all.x = TRUE)
```

```
datatest=datatest[order(datatest$id),]

datatrain=datatrain[order(datatrain$id),]




#***************

#Preparing data functions

#***************

mytrain=datatrain

mytest=datatest


#make the outcome be numeric

numoutcome=function(data){

   data$outcome[data$interest_level=='low']=0

   data$outcome[data$interest_level=='medium']=1

   data$outcome[data$interest_level=='high']=2

   return(data)

}


#select numerica features and impute missing value


selectnum_imputemiss=function(d){

   nums <- sapply(d, is.numeric)
   d=d[,nums]
   #d$sim[is.na(d$sim)==TRUE]=0
   return(d)
}

#***************
#Prepare DATA
#***************
mytrain=numoutcome(mytrain)
mytrain=selectnum_imputemiss(mytrain)

#split outcome
mytrain_outcome=mytrain$outcome
mytrain$outcome=NULL

mytest=selectnum_imputemiss(mytest)

#*****************************
#xbg_1
#*****************************
#xgb boost

dtrain1 <- xgb.DMatrix(data=as.matrix(mytrain), label=mytrain_outcome)
dtest1 <- xgb.DMatrix(data=as.matrix(mytest))
```

```
numberOfClasses <- length(unique(mytrain_outcome))

xgb_params <- list(booster="gbtree",
                   objective="multi:softprob",
                   eval_metric="mlogloss",
                   nthread=13,
                   num_class=numberOfClasses,
                   eta = .03,
                   gamma = 1,
                   max_depth = 6,
                   min_child_weight = 1,
                   subsample = .7,
                   colsample_bytree = .7
)

nround    <- 1000 # number of XGBoost rounds
cv.nfold  <- 5
# Fit cv.nfold * cv.nround XGB models and save OOF predictions
cv_model1 <- xgb.cv(params = xgb_params,
                    data = dtrain1,
                    nrounds = nround,
                    nfold = cv.nfold,
                    verbose = TRUE,
                    prediction = TRUE)
```

```
mean(cv_model1$dt$train.mlogloss.mean)
mean(cv_model1$dt$test.mlogloss.mean)



#***************
#training
xgb1<- xgb.train(data = dtrain1,
                 params = xgb_params,
                 # watchlist=watch,
                 # nrounds = cv_model1$best_ntreelimit
                 nrounds = nround
)

# compute feature importance matrix

imp <- xgb.importance(names(mytrain),model = xgb1)
print(imp)



#***************
#Saving result
#***************
```

```
#***select features



sPreds <- as.data.table(t(matrix(predict(xgb1, dtest1), nrow=3, ncol=nrow(dtest1 ))))

class <- data.table(interest_level=c("low", "medium", "high"), class=c(0,1,2))
colnames(sPreds) <- class$interest_level



setwd("C:/Users/ellie/Dropbox/2 sigma/TRY24 EVERYTHING")

write.csv(data.table(listing_id=mytest$listing_id, sPreds[,list(high,medium,low)]),
          "submission_apr24-1.csv",row.names=FALSE)
```

Reference

Han, J., Kamber, M., & Pei, J. (2012). Data mining: Concepts and techniques. Waltham, MA: Morgan Kaufmann.

Chen, T., & Guestrin, C. (2016). XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16.

Kulkarni, S., Harman, G., & Wiley InterScience (Online service). (2011). *An elementary introduction to statistical learning theory*. Hoboken, NJ: Wiley.

In Perner, P. (2015). *Machine Learning and Data Mining in Pattern Recognition: 11th International Conference, MLDM 2015, Hamburg, Germany, July 20-21, 2015, Proceedings*.

Two Sigma Connect: Rental Listing Inquiries | Kaggle. (n.d.). Retrieved from https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries

Tune Machine Learning Algorithms in R (random forest case study) - Machine Learning Mastery. (n.d.). Retrieved from http://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

Gradient descent - Wikipedia. (n.d.). Retrieved May 3, 2017, from https://en.wikipedia.org/wiki/Gradient_descent

Boosting (machine learning) - Wikipedia. (n.d.). Retrieved May 3, 2017, from https://en.wikipedia.org/wiki/Boosting_(machine_learning)

Gradient descent - Wikipedia. (n.d.). Retrieved May 3, 2017, from https://en.wikipedia.org/wiki/Gradient_descent

Log Loss | Kaggle. (n.d.). Retrieved from https://www.kaggle.com/wiki/LogLoss