NYC CITADEL DATATHON REPORT

Predicting the average prices and number of properties for

each small area in Airbnb listings

Team 14
Sep. 30, 2017

Abstract

In this project, we aim to predict the average prices and numbers of listed properties for each small area (by zip code) in Airbnb listings using area-specific economics, demographics and real-estate data. In the meantime, we show which factors have large predictive power on the Aribnb rental postings and how they influence the listing prices or numbers.

1. Introduction

Airbnb started in 2007 by two recent college graduates to achieve the goal of affordable rent in an expensive city. It has now become a very popular online platform to link the hosts and renters, which benefits both sides enormously: hosts can easily rent out their vacant rooms and renters can enjoy an affordable stay in an expensive city. As both hosts and renters use the Airbnb platform individually, they need guidance on prices to set for their rooms (for hosts), prices to take for their accommodation (for renters) and whether or not to use Airbnb for a stay (for renters). The questions naturally occur that how are the Airbnb listed prices for each small area depend on the that area's economics, demographics and real-estate information. By predicting the average listed prices on Airbnb for each small area using that area's geographical features, we provide a reference price for both the hosts and the renters when they intend to make or accept an accommodation price in one area through Airbnb. The other question is how is Airbnb's presence level in each small area depends on that area's geographical features. By predicting the number of listed properties on Airbnb for each small area, we provide the potential renters the popularity degree of Airbnb in each area, which information facilitates their accommodation selections.

Our project intends to capture the influence of economics, demographics and real-estate data on the average listed prices and number of listed properties on Airbnb for each small area (one zip code). Through capturing their relationship, we predict the average listed prices and number of listed properties using each area's geographical features, which largely helps Airbnb users who have less knowledge about the whole Airbnb listings data. Our economics data include the most recent GDP, income per capita and unemployment rate for each state. These variables measure how 'expensive' each area's living standard is and their resulting influence on Airbnb activities show how listed prices and popularity of Airbnb change with the 'expensive' level of each area. Our demographics data contain the population, number of people in each age bracket, total households, percentage of households in each given income bracket and median/mean household income. These variables capture how 'rich' the people in each area are. The real-estate variables include Zillow Home Value Index (ZHVI) rank, Zillow Rent Index (ZRI) rank and latest ZHVI value. These variables measure how 'expensive' housing is in each area.

Our final price prediction model has very close in sample and out of sample performances in both re square and root mean squared error, which is desirable. We use descent gradient boosting to predict listed properties numbers, which also works fine.

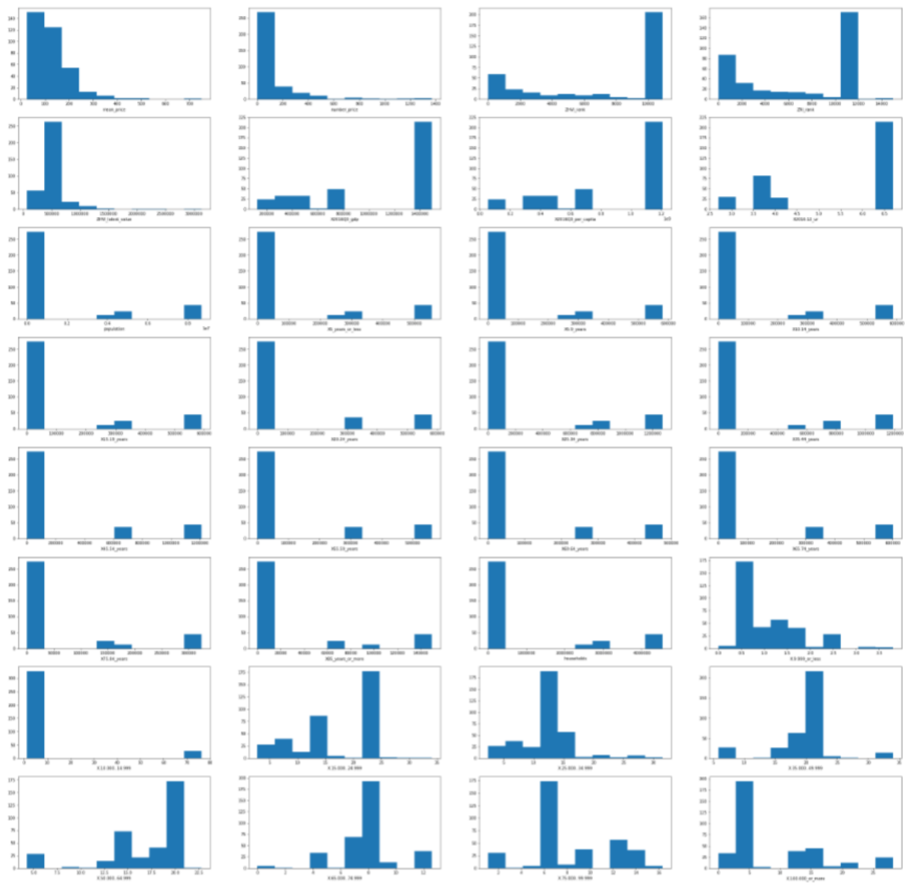
The remainder of this report is organized as follows. In section 2 and 3, we discuss the data manipulation process and exploratory data analysis respectively. In section 4, we describe the statistical modelling process we conduct to make predictions starting from data preprocessing, feature selection and including linear regression model, support vector machine, gradient descent boosting and random forest. In section 5, we show the summary statistics of our results and draw conclusions.

2. Data Manipulation

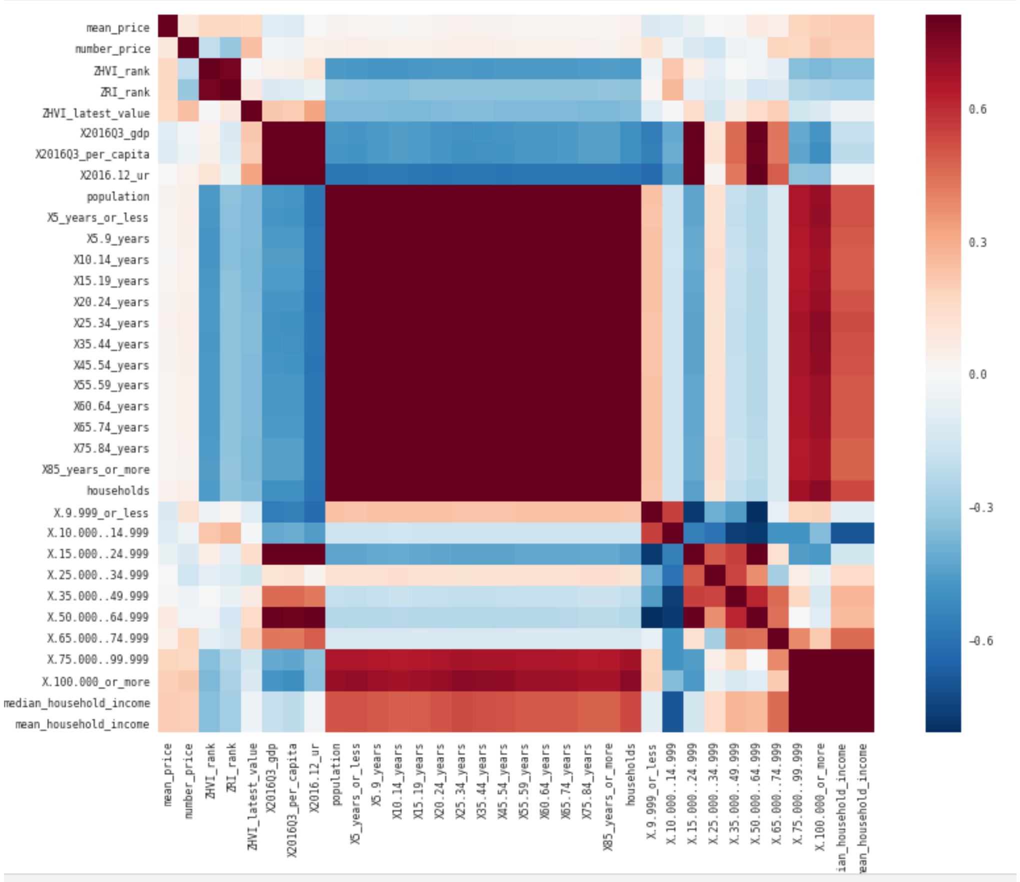
We construct our response variable y from the listings csv file. For each zip code, we calculate the average of one-night rental price (price) among all listed properties as one response variable; we also calculate the number of listed properties for each zip code as another response variable. We take the latest gdp (2016 Q3), per_capita_income (2016 Q3) and unemployment_rate (2016.12) from econ_state dataset and merge with the y variables by state. We take all the variables in the demographics dataset and merge with the y variables by city and state. Finally we take ZHVI rank, ZRI rank and latest ZHVI value from real_estate dataset and merge with y variables by zipcode.

3. Exploratory Data Analysis

Skewed Data: we perceive that the majority distributions are heavily skewed to either direction. Since our later prediction model such as SVM or regression requires normality of the input data distribution to achieve accurate prediction, we decide to take standardization on our data.



Collinearity: The correlation heat map is used to detect collinearity issue across all variables. We find that there is a high and positive correlation among the variables regarding to economics and income. This prompts us to use regulation techniques later in our model development.



4. Statistical Modelling

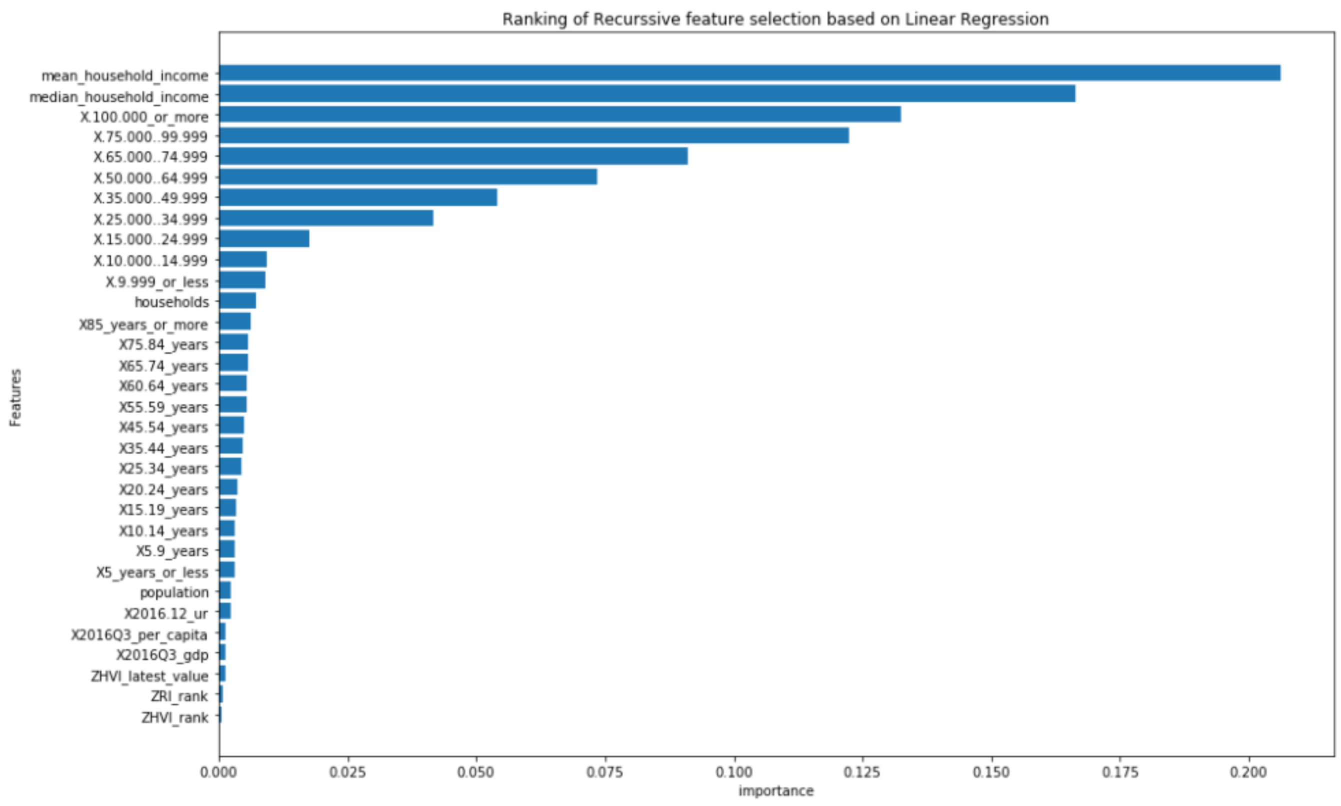
4.1 Data Preprocessing

We first clean data by removing rows with NA entries. Then we normalize each independent x variable by dividing by each column's standard deviation. After that we randomly split the data into training data (75%) and testing data (25%).

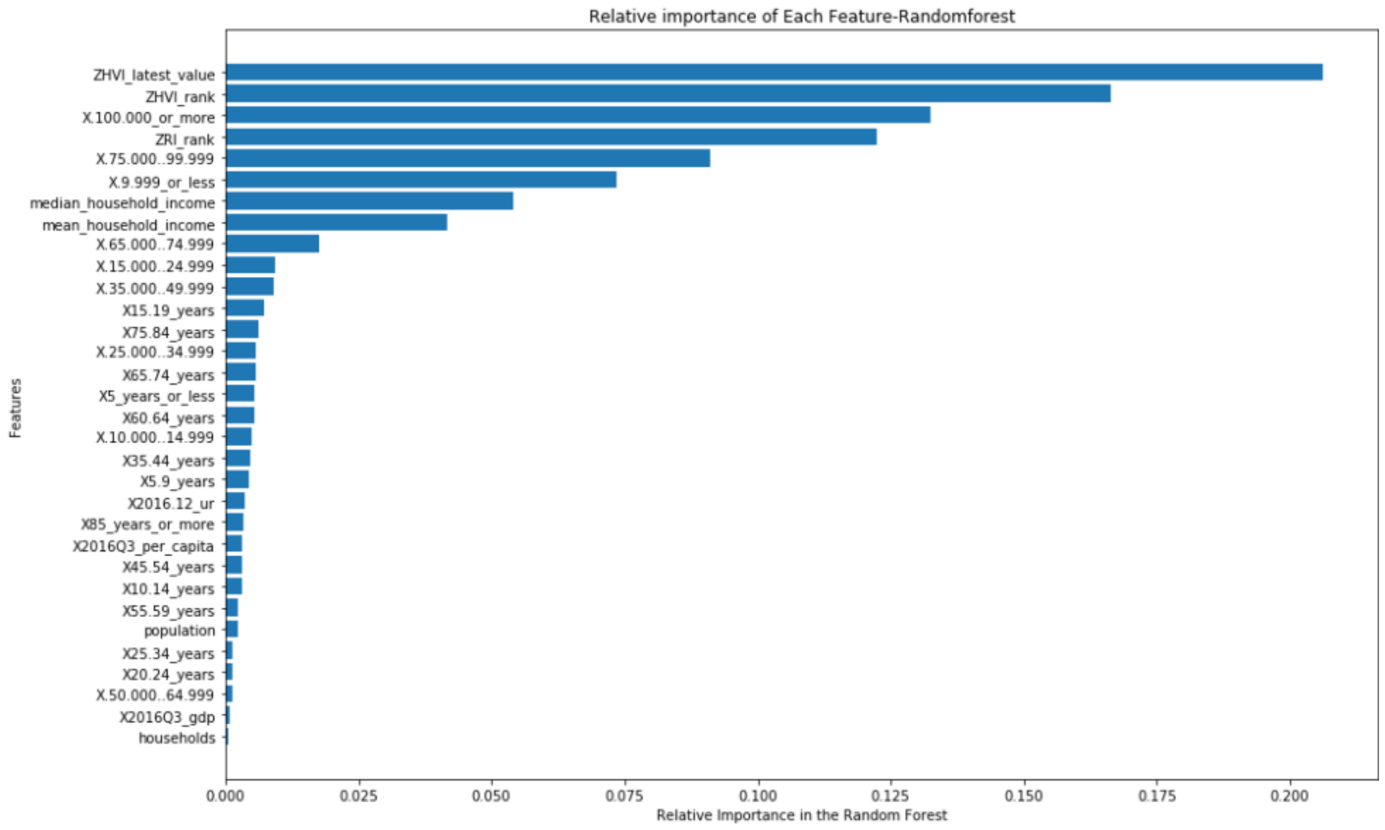
4.2 Feature Selection

Feature ranking can be incredibly useful in a number of machine learning and data mining scenarios. We rank our features based on three robust methods: RFE on Linear Regression, Random Forest and Correlation. The overall ranking score is then calculated as the average of standardized scores from three methods.

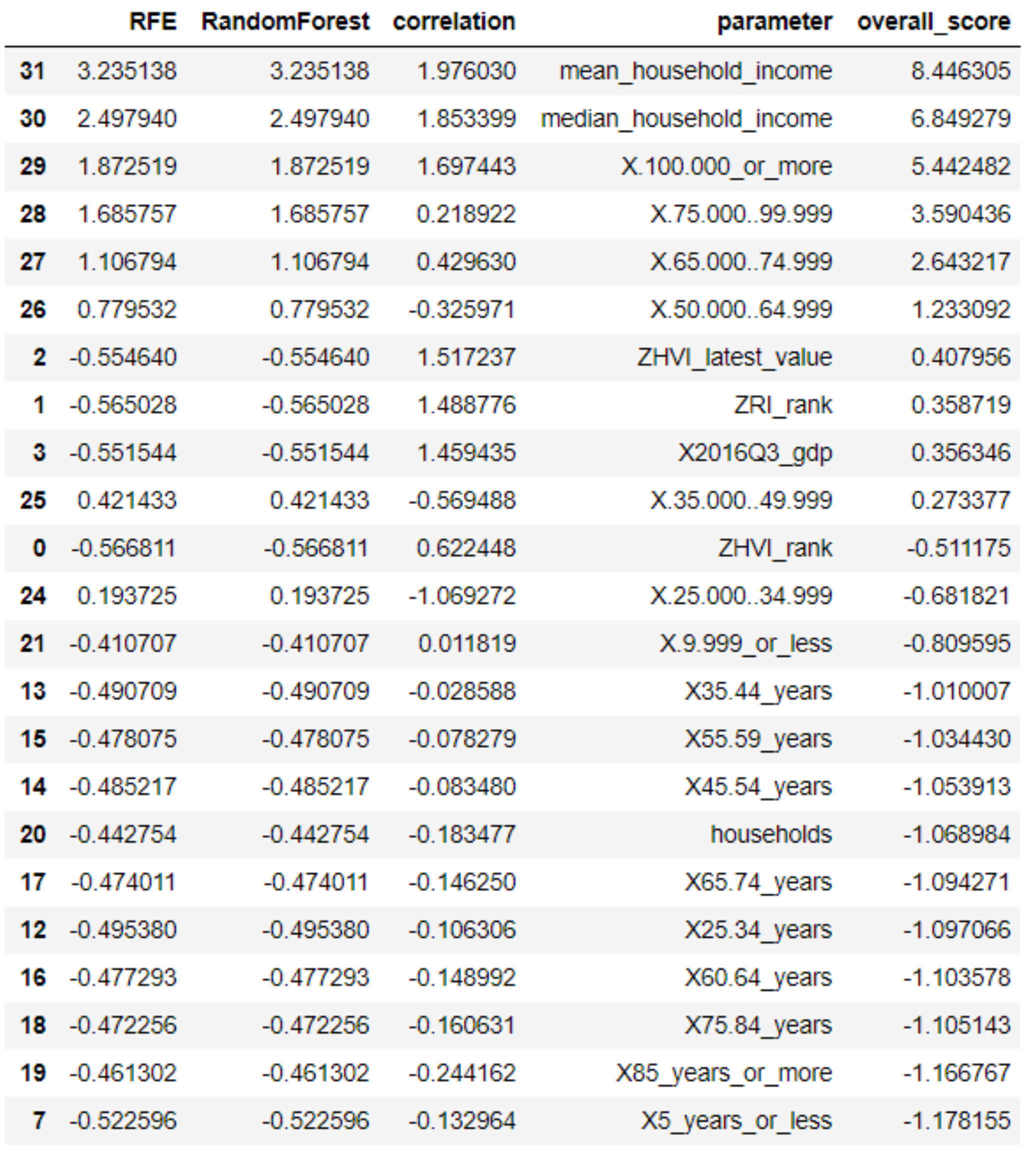
We choose RFE because recursive feature elimination is based on the idea to repeatedly construct a model (linear regression here) and choose either the best or worst performing feature (for example based on coefficients), setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. Features are then ranked according to when they were eliminated. As such, it is a greedy optimization for finding the best performing subset of features.



Random forest's impurity based ranking is aggressive in the sense that there is a sharp drop-off of scores after the first few top ones. This can be seen from the example where the third ranked feature has already 4x smaller score than the top feature (whereas for the other ranking methods, the drop-off is clearly not that aggressive).



After getting the individual scores from three methods, in order to ensure the consistency measure, we standardize the scores and use the average as the final score for our feature ranking and later features selections



	RFE	RandomForest	correlation	parameter	overall_score
31	3.235138	3.235138	1.976030	mean_household_income	8.446305
30	2.497940	2.497940	1.853399	median_household_income	6.849279
29	1.872519	1.872519	1.697443	X.100.000_or_more	5.442482
28	1.685757	1.685757	0.218922	X.75.000..99.999	3.590436
27	1.106794	1.106794	0.429630	X.65.000..74.999	2.643217
26	0.779532	0.779532	-0.325971	X.50.000..64.999	1.233092
2	-0.554640	-0.554640	1.517237	ZHVI_latest_value	0.407956
1	-0.565028	-0.565028	1.488776	ZRI_rank	0.358719
3	-0.551544	-0.551544	1.459435	X2016Q3_gdp	0.356346
25	0.421433	0.421433	-0.569488	X.35.000..49.999	0.273377
0	-0.566811	-0.566811	0.622448	ZHVI_rank	-0.511175
24	0.193725	0.193725	-1.069272	X.25.000..34.999	-0.681821
21	-0.410707	-0.410707	0.011819	X.9.999_or_less	-0.809595
13	-0.490709	-0.490709	-0.028588	X35.44_years	-1.010007
15	-0.478075	-0.478075	-0.078279	X55.59_years	-1.034430
14	-0.485217	-0.485217	-0.083480	X45.54_years	-1.053913
20	-0.442754	-0.442754	-0.183477	households	-1.068984
17	-0.474011	-0.474011	-0.146250	X65.74_years	-1.094271
12	-0.495380	-0.495380	-0.106306	X25.34_years	-1.097066
16	-0.477293	-0.477293	-0.148992	X60.64_years	-1.103578
18	-0.472256	-0.472256	-0.160631	X75.84_years	-1.105143
19	-0.461302	-0.461302	-0.244162	X85_years_or_more	-1.166767
7	-0.522596	-0.522596	-0.132964	X5_years_or_less	-1.178155

4.3 Linear Regression Model

We first use linear regression models to fit our data, where independent x variables are economics, demographics and real-estate data for each zip code and y variable is the mean listed one-night price for each zip code. For this linear regression model, we assume that the relationship between x and y is linear.

4.3.1 Ordinary Least Squares

We first fit an ordinary least squares model between x and y using the function lm in R. Then we only keep those variables in x that have significant estimated coefficients, which are ZHVI_rank, ZHVI_latest_value, 2016Q3_gdp, 2016Q3_per_capita, 75.84_years and 9.999_or_less. The estimated coefficients are:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	111.67280	23.438768	4.764448	2.788489e-06
ZHVI_rank	21.18607	4.741958	4.467789	1.071063e-05
ZHVI_latest_value	20.00510	4.450139	4.495387	9.478213e-06
X2016Q3_gdp	82.35365	86.864615	0.948069	3.437541e-01
X2016Q3_per_capita	99.04076	85.508889	1.158251	2.475581e-01
X75.84_years	15.50532	5.522675	2.807573	5.273809e-03
X.9.999_or_less	-18.28986	5.136186	-3.560981	4.210225e-04

The root mean squared error for training set is 74.76781 and that for testing set equals 95.68859.

4.3.2 Ridge Regression

Ridge regression provides a way for coefficient shrinkage by adding an squared L2 norm regularization term in the minimization objective function. We use ridge regression because of the high correlation among our x variables. We use the function glmnet in the package glmnet in R. We train the parameter lambda using cross validation and input the best lambda into our prediction model. The root mean squared error for training set is 74.21548 and that for testing set equals 94.70719.

4.3.3 Lasso Regression

Lasso regression provides a way for variable selection which forces many estimated coefficients to zeros by adding an L1 norm regularization term in the minimization objective function. We use it again because of the collinearity problem. We again use the function glmnet in the package glmnet in R. The root mean squared error for training set is 76.15058 and that for testing set equals 94.42657.

Among these three linear regression models, the ridge regression gives the best prediction results for the training set.

4.4 Support Vector Machine Model

Support vector machine captures the nonlinear relationship between x and y by using the kernel trick mapping the variables into a much higher dimension space. It solves a convex optimization problem, which guarantees a result and any local result serves any the global result.

We use the function svm in the package e1071 in R to train our support vector regression model. It automatically figures whether our response variable is continuous or discrete. We get a root mean squared error of 75.0117 for the training set and 96.31665 for the testing set.

4.5 Gradient Descent Boosting

4.5.1 Methodology

Overfitting is a really common phenomena when fitting a regression. Gradient Descent boosting(GBD) is not only a greedy algorithm and can overfit a training dataset quickly, it could also benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. Due to the number of features in our dataset, we decide to choose Gradient Descent Boosting to reduce our overfitting.

4.5.2 Process

First, we would like to see feature importance by fitting a GBT first. As we could see on figure 1, ZRI_rank, ZHVI_rank, ZHVI_lastest_value, X2016Q3_gdp and X.15.000.24.999 are the most significant feature in our whole dataset, which means both gdp, fixed income and real estate value have important impact on Airbnb normal price.
Second, screening the top five features in our dataset, and then rerun our program, as we could see in figure 2 and figure3, the performance MSE is significantly better than before and there is not too much overfitting in our case.

4.5.3 Results

On training set: For mean price of Airbnb in a certain zip code RMSE: 57.9372

For mean number of Airbnb in a certain zip code RMSE: 101.0777

On testing set: For mean price of Airbnb in a certain zip code RMSE: 61.72

For mean number of Airbnb in a certain zip code RMSE: 105.0777

4.5.4 Plots and tables

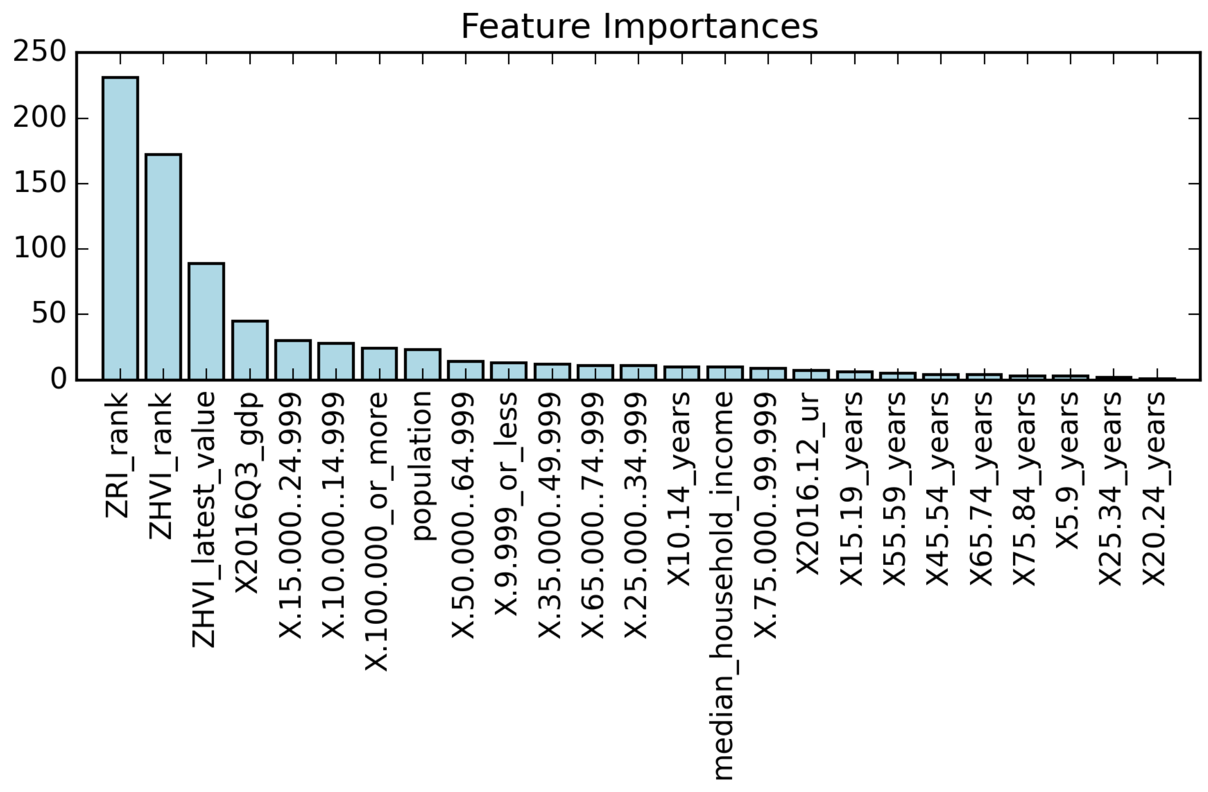


figure 1

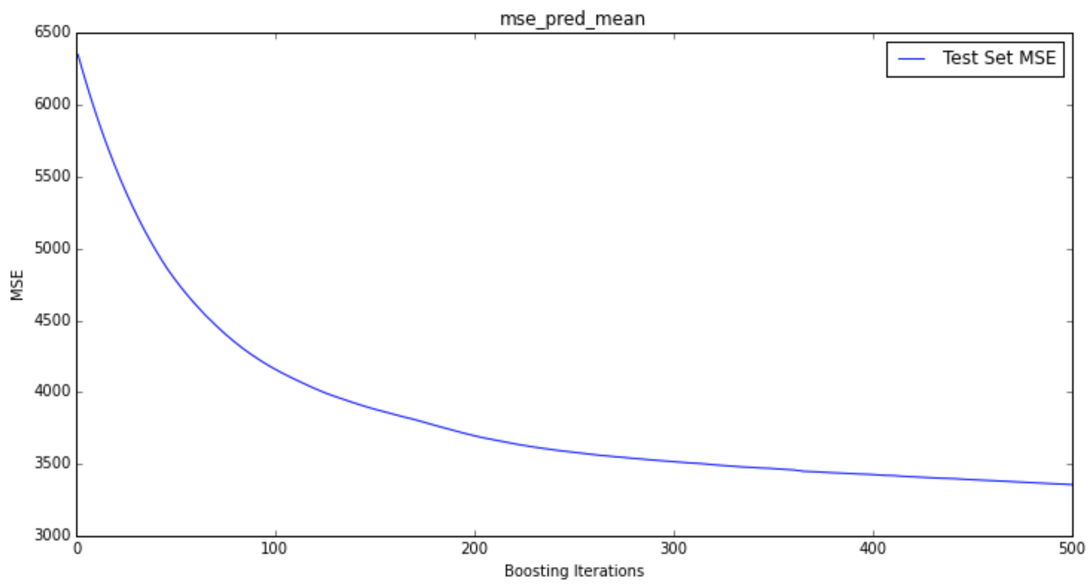


figure 2

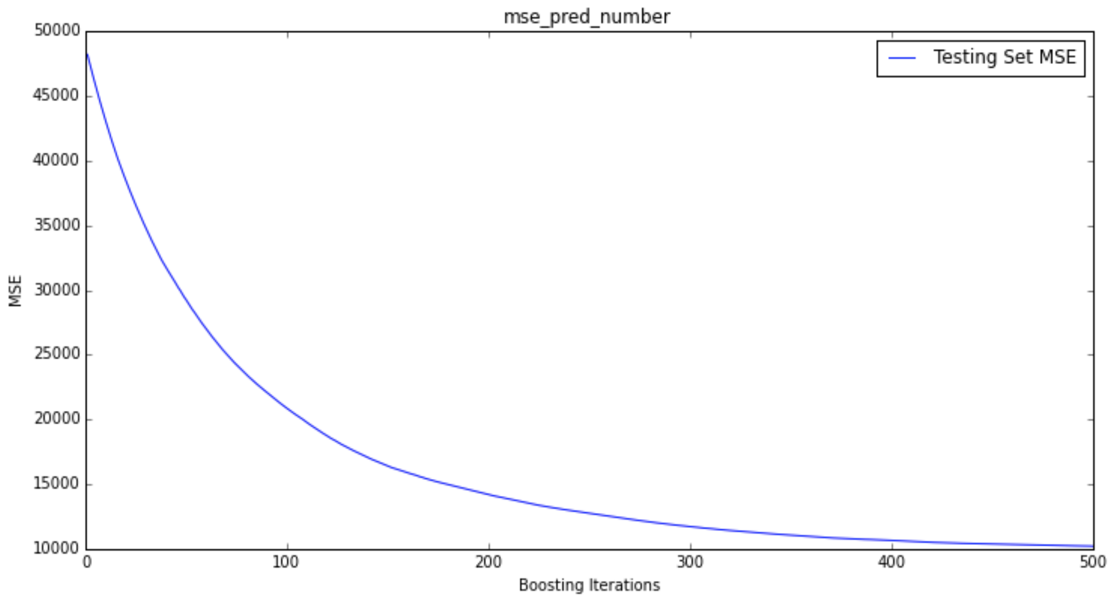


figure 3

4.6 Random Forest

4.6.1 Methodology

Random Forest is an ensemble learning algorithm using many decision tree models. It is a very robust learning algorithms with following advantages.

a) Random Forests can handle the missing values and maintains accuracy for missing data
b) Won't overfitting the model
c) Handle large data set with higher dimensionality.

For our datasets, Random Forests stands out for there are many features and missing data as well. We can use Random Forest to handle these unpleasant situations.

4.6.2 Process

a) Search for the best parameters.

There are several parameters that are essential to random forest performance. For example, number of estimators, maximum depth of each tree, minimum numbers of sample split. So at the first step, we used GridSearchCV from sklearn in Python to determine the best parameters.

For the sake of convenience, we just plot the RandomForestRegressor scores as a function of the number of trees. We used 10 validation sets to test the model performance.

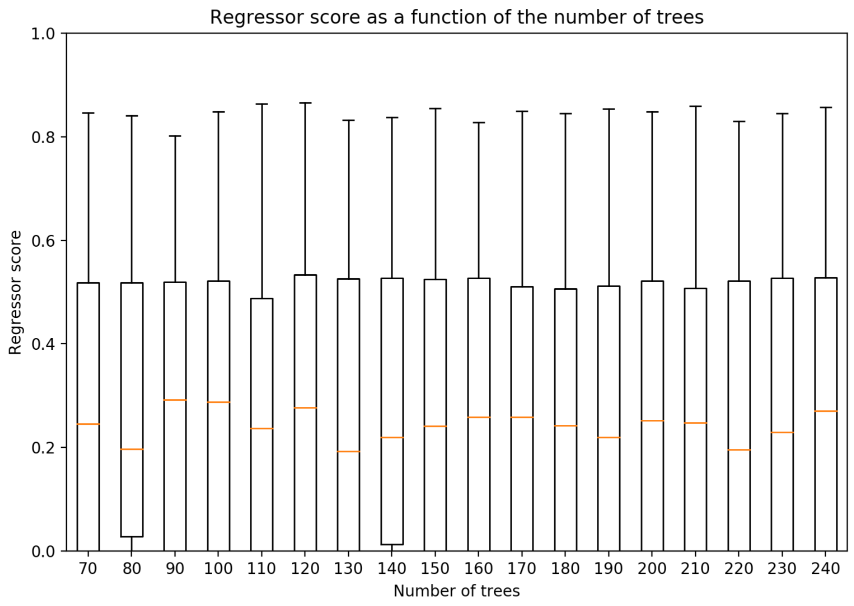


Figure 4. Regressor score as a function of the number of trees

We can draw the conclusion that n_estimator = 90 is the best parameter for the number of sub-trees. This is the same as we got from using GridSearchCV. Final parameters used for Random Forest are:

n_estimators	**90**

min_sample_leaf	**2**
Min_sample_split	**4**
Max_depth	**4**
oob_score	**True**

Figure 5. Parameters used for Random Forest

4.6.3 Results for training and testing set

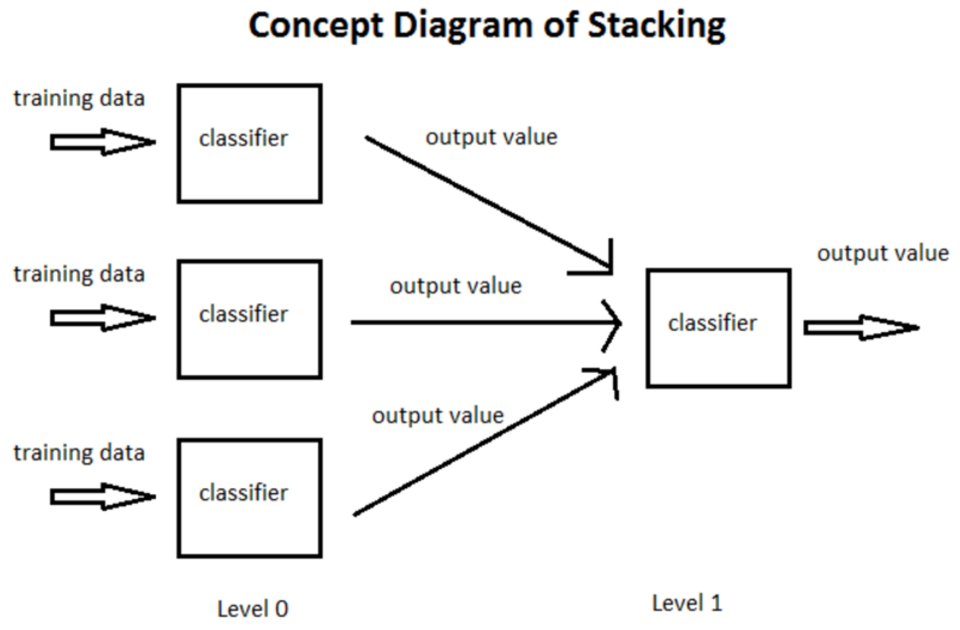
After we determine the parameters, we can easily train Random Forest Model on training sets. And then test the performance on the testing datasets. Root mean square error of Airbnb in a certain zip code are shown below:

Training Set	71.9
Testing Set	82.6

Figure 6. RMSE of Airbnb in a certain zip code

4.7 Stacking

Stacked generalization is a method of combining generalizers to make a new generalizer. The goal is to optimally integrate what each of the original generalizers has to say about the learning set.



After careful testing and tuning, Ridge regression, random forest and XGB boost stand out as the good prediction models for the problem. After stack the output values from three models to form the independent variable matrix, we apply a final layer linear regression on the matrix. This is to aim for preserving the best among three models while averaging out the errors which will lead much more accurate predictions in the end.
Below result table confirms the best prediction coming from two layer stacking model.

Performance	Ridge Regression	Random Forest	XGB Boost	Final Model: Ridge, Forest and XGB *ensemble by* Regression
Training RMSE	74.215	71.9	59	55
Training R^2	0.5421	0.58	0.602	0.64
Test RMSE	94	82.6	61	54
Test R^2	0.5237	0.59	0.593	0.63

5. Summary Statistics and Conclusions

We start with exploratory data analysis to detect non-normality and collinearity. Then we perform feature selections and features ranking through a thorough and systemic average method using RFE, random forest and correlation to find the appropriate features to use. After that, we carefully test and tune three machine learning and statistical models for the problem. Finally, we innovatively ensemble three models together with a linear regression to achieve even better prediction.

For any places in U.S, as granular as to zip code level, our tested model will able to provide a good recommendation to Airbnb regarding the average listed price level.