Become a Top Airbnb Earner

CMPS 140 Project Final Report

Tangni Wang
University of California Santa Cruz
twang63@ucsc.edu

Tung Hoi Man
University of California Santa Cruz
tuman@ucsc.edu

Yunxiang Fu
University of California Santa Cruz
yfu7@ucsc.edu

ABSTRACT

This project is referenced from a project idea and a corresponding dataset on Kaggle with the same name. Airbnb, as a privately held global company that provides an online service to connect host and guests via websites and mobiles apps to form a better travel experience. The fact that not all guests are using this service with the same purpose. There are poor and rich, some use it for short term traveling and others use it for long term stay, some prefer locations and some prefer house quality.

Airbnb hosts' earning could vary quite significantly in the same area. We want to know what factors help low performers to receive better reviews from others and how to make more profit. Within the idea of helping Airbnb host to develop their service and become a future top earner, we used the dataset from Seattle Airbnb dataset to predict the important features for Airbnb premium users in order to become a top earner. In this document we will first describe the reason we decided to work on the project "how to become a top Airbnb earner" and our motivation, purpose and goal. Then we will discuss data, methodology, features we used, evaluation on our results and our conclusion.

1. INTRODUCTION

1.1 Goal and Impacts:

Due to most of concerns on how to be a top earner, we are using the datasets of the Airbnb Seattle case to compare different performers' data. We want to have those ones with more bookings and high rating in a comparison with the ones with less bookings and low rating in terms of what are the major features and what are the things that matter more than the other. This way, the hosts would be able to know the users' needs so they can safe money and spend it on some features that matter more than the other. Our main goal is to make recommendations for the low earner hosts and make their listings more profitable and gain more revenue than before.

1.2 Related Work

Since 2008, guests and hosts have used Airbnb for an easy and more fordable trip. As a part of the Airbnb inside initiative, the dataset we use that describes the listing activity of homestays in Seattle, WA, is from Kaggle "Seattle Airbnb Open Data" dataset. The methodologies we use for our baseline model Dummy Regression with median value and stretch model includes Linear Regression, Linear Regression with cross validation, Decision Tree, Lasso Regression, Bayesian Ridge Regression, Naïve Bayes, and Random Forest Regression, are referenced from sklearn.
Many comparisons have been done between high earners and low earners. We are inspired by those work. We want to see if there are other interesting factors or combination of factors that would affect booking rate. The features that we will be analyzing is based on the top ranked correlation of score_review. By having the monthly_reviews and the review_rating, we are multiplying them and divide by 100 to know approximately which owners have more completed orders or better reviews than others. By looking at the score reviews of top earner we will be able to identify what are the factors to be a top performer and what make the low performers. (reference 2)

2. DATA

2.1 Relevant Data

The data we use comes from Seattle Airbnb Open Data (reference 2). There are three datasets from this open source which includes calendar.csv, listing.csv and reviews.csv. Calendar has attributes: listing_id, the price and availability for that day. The review has listing_id, date, reviewer_id, reviewer_name and comments. Listing have the full descriptions of each post. Since it had a lot of attributes, we will extracted some useful attributes to build out model.

Calendar.csv: 1048576 rows
Listing.csv: 3819 rows, 92 columns
Reviews.csv: 84850 rows

We used only Listing.csv for our project. The row number means how many hosts we are looking at, the column number means how many features we start with. The ***original dataset*** provided **92** features but most of them are not relevant features such as id, name, picture_url. We dropped those columns with .drop() and end up with **17 *relevant features*** that we will use.

2.2 Data Preprocessing

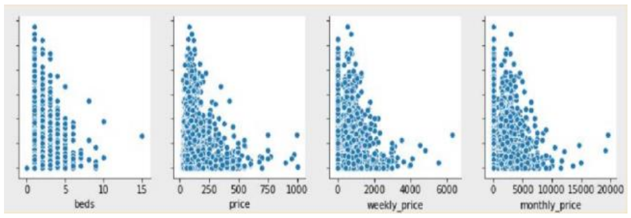
In order to our program to recognize all data, we also did some cleaning with these features.

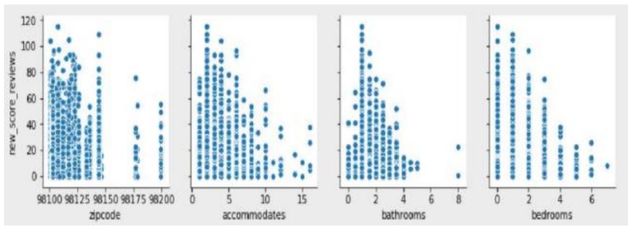
1. Function called ***correction()*** that takes away all "$" and "," with price, weekly_price, security_deposit, extra_people, cleaning_fee.
2. Another function called ***correction2()*** that takes away "%" and convert percentages to decimals.
3. Converted ***boolean*** "true" and "false" to integer "1" and "0" with host_is_superhost, host_identity_verified, instant_bookable.
4. Function ***changeTime()*** that changes host_response_time columns from string into numerical:
   a. "within an hour" to 1.
   b. "within a few hours" to 4.
   c. "within a day" to 24.
   d. "a few days or more" to 48.
5. Function ***convertPolicy()*** that converts cancellation_policy columns from string into numerical:
   a. "strict" to 1.
   b. "moderate" to 3.
   c. "flexible" to 5.
6. Fill missing value with reasonable values like 0 or 1.

After the data cleaning we had a very nice and clean data to analyze.

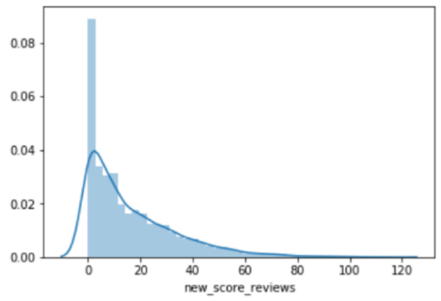
2.3 Distributions

To have a better evaluation of our model performance, we used function ***sns.pairplot(seattle_data)*** to have the pairs plot built on two basic figures, the histogram and the scatter plot. This histogram on the diagonal allows us to see the distribution of a single variable while the scatter plots on the upper and lower triangles show the relation (or lack thereof) between two variables. Here is part of the histogram in the figure below:





We also have a histogram that divides the variable into bins, counts the datapoints in each bin, and shows the bins on the x-axis and the counts on the y-axis to the figure:



We then took a ***label correlation coefficient***, or simply the correlation of our dataset, which has an index that ranges from ***-1 to 1***. When the value is near zero, there is no linear relationship. As the correlation gets closer to 1 or -1, the relationship is stronger. A value of exactly 1 or -1 indicates a perfect linear relationship between two variables.

Our result shows the features that have a relevantly:
Positive: bedrooms, price, cleaning_fee, accommodates, beds, bathrooms, weekly_price, monthly_prive, security_deposit, extra_people, minimum_nights.
Negative: hsot_is_superhost, instant_bookable, host_response_time, host_identity_verified, zipcode, cancellation_policy.

Stronger correlation: host_is_superhost, instant_bookable, host_response_time, bedrooms, price, cleaning fee.
Weaker correlation: accommodates, beds, bathrooms, host_identity_verified.
Almost no correlation: zipcode, weekly_price, monthly_price, security_deposit, extra_people, cancellation_policy, minimum_nights.

We have divided the dataset into Training set and Testing set:
Training set: 80%
Testing set: 20%

3. METHODOLOGY

We first made our baseline model (details in 3.2). We use it to compare to our stretch methods (details in 3.3). Then, we tried five different regression models and we are going to pick the following two as our main methods because they use very different mechanism to predict new values. They also turn out to have quite different result so they are worth to go into detail.

Linear Regression:
We use it to predict the target variable (y) based on independent variable (X). See 3.1 for more details. So, we can finds out a

linear relationship between features (X) and target variable (y). Also, we can know which feature is more important by looking at the coefficient of each feature.

The coefficients of 17 features:

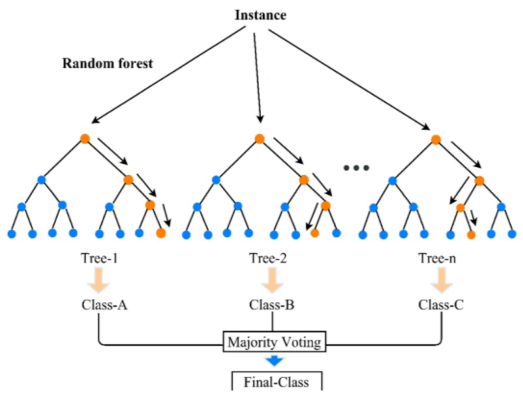
Features	**Coefficients**
zipcode	-1.60501238e-02
accommodates	9.59663259e-01
bathrooms	7.59530190e-02
bedrooms	-1.99980604e+00
beds	-1.03277111e-01
price	-1.20640722e-02
weekly_price	6.09900969e-04
monthly_price	2.14805937e-04
security_deposit	-2.10722857e-03
cleaning_fee	-7.07223764e-02
extra_people	4.66326070e-02
minimum_nights	-1.61309119e-02
host_response_time	-1.16148260e-01
host_is_superhost	1.04866759e+01
host_identity_verified	2.36693767e+00
instant_bookable	9.50549861e+00
cancellation_policy	-7.49124008e-01

We can see features like ***bedrooms, host_is_superhost, host_identity_verified, instant_bookable and cancellation_policy*** have a relatively larger coefficients than the others. These features have a stronger linear relationship to the target variable (y = new_score_reviews).

Note: The correlation coefficient is an index that ranges from -1 to 1. When the value is near zero, there is no linear relationship. As the correlation gets closer to plus or minus one, the relationship is stronger.

Improve model with cross validation (cv):
After that we also tried to improve model with cross validation (cv): cv can ensure data split randomly when we split it to train/test set. Here, we performed 9-fold cross validation (cv) for linear regression. The score is improved by 20% (details on section 4). However, mean absolute error (mae) and mean square error (mse) didn't improve at all because the core mechanism for linear model to predict new value is still the same. The only difference is the input data are split more randomly than before.

Random Forest Regression:
This method uses multiple decision trees in determining the outputs instead of using one decision tree. The higher the number of trees in the forest gives the high accuracy results. It uses majority voting.



Prediction mechanism:

1. First take the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target).
2. Calculate the votes for each predicted target.
3. Consider the high voted (majority voting) predicted target as the final prediction from the random forest algorithm.

There are couple reasons for using random forest model:
It is very easy to measure the relative importance of each feature on the prediction.It is very good at handling tabular data with numerical features, or categorical features with fewer than hundreds of categories. Compare to linear models, random forests can capture non-linear interaction between the features and the target.

Feature importances:

Features	**Importances**
zipcode	0.09530789
accommodates	0.03210362
bathrooms	0.02251223
bedrooms	0.02483156

Features	**Importances**
beds	0.01582465
price	0.12527684
weekly_price	0.0590654
monthly_price	0.05294428
security_deposit	0.03838474
cleaning_fee	0.10170245
extra_people	0.05346347
minimum_nights	0.07522294
host_response_time	0.14587976
host_is_superhost	0.06682052
host_identity_verified	0.0158356
instant_bookable	0.03819375
cancellation_policy	0.0366303

We can see features like ***zipcode, price, cleaning_fee, host_response_time, host_is_superhost*** are more important than the other features.

The resource code and original explanation of the above methods can be refer to the scikit-learn documentation page (reference 4).

3.1 FEATURES

A new feature:

Since we are analyzing how to become a top Airbnb earner, it would be the best to have the amount of bookings and the price of each booking in order to get how much each Airbnb host are earning per month. However, unfortunately we do not have related features in the provided datasets such as the bookings. Instead, we decide to use *review_per_month* and *review_score_rating* as our variables to get the result of the host that gets the best reviews. As an online service that connects host and visitors, Airbnb reviews and ratings are extremely important because that is the first thing that pops out when searching hosts, this is also one of the only way for guest to know about the host. Imagine some host with terrible rating and reviews will not likely to have many bookings. Even if the host has very low price, he/she will still be less likely to get more bookings as the hosts that have the similar price but better rating. As a conclusion, we came up with a new feature ***new_score_review*** as our target variable with the equation:

*new_score_reviews = (reviews_per_month * review_scores_rating) / 10*

Top and low performers threshold:

Now with our new feature ***new_score_reviews***, we are assuming the top 90% of hosts in new_score_reciews are the top earners and the low 25% are the low earners. We defined the top and low performers threshold with **quantile(0.9)** and **quantile(0.25)** to set top90flag and upto25flag :

Top performers: new_score_reviews >= 90% quartile (44.11)

Low performers: new_score_reviews <= 25% quartile (6.48)

For top 90% boundary we got score proximately 44.11 and below 25% boundary we got score proximately 6.48, which means any host scores greater than or equal to 44.11 is considering a top performer, and any host scores less than or equal to 6.48 would consider as a low performer. Next, we will apply our model to compare attributes, decide which attributes has causal effect to new_score_reviews.

17 relevant features:

We extracted 17 features which we think are relevant to the performance of airbnb hosts.

zipcode, accommodates, bathrooms, bedrooms, beds, price, weekly_price, monthly_price, security_deposit, cleaning_fee, extra_people, minimum_nights, host_response_time, host_is_superhost, host_identity_verified, instant_bookable, cancellation_policy.

We have 92 features from the dataset but we only pick these 17 features that are relevant to want we want to predict (new_score_reviews). The other features like listing_id, picture_url are irrelevant so we just dropped them.

3.2 BASELINES

We are using Dummy Regressor from sklearn library to make our baseline model (reference 3). There are several rules we can choose. Mean always predicts the mean of the training targets. Median always predicts the median of the training targets. Quantile always predicts a user provided quantile of the training targets. Constant always predicts a constant value that is provided by the user. For our project, we choose median to get the median of the training targets. This regressor used here to make prediction by simple rule as a simple baseline to compare with other real regressors we are going to implement.

3.3 STRETCH METHODS

Linear Regression:

We import linear_model from sklearn library. We build the linear regression model and use its fit method with our training set as parameter (X_train, y_train). Then, we use the predict method with our test set as parameter (X_test) to predict new value of y. We use pyplot from matplotlib library and seaborn library to plot graph. The graph (figure 1) shows the distribution of y_test vs y_pred_lm.

We also use mean_squared_error, r2_score, mean_absolute_error, median_absolute_error, mean_squared_log_error, explained_variance_score from sklearn.metrics library to measure how good is this model for prediction. Last, the method lin_model.coef_ provide us the coefficient of each feature.

Linear Regression with cross validation (cv):
We perform 9-fold cross validation (cv) for linear regression. The graph (figure 2) shows the distribution of y_test vs y_pred_lm_cv.
Cross-validated scores: [0.21546241 0.28762751 0.28189387 0.24390108 0.25506256 0.26884101 -2.61113245 0.27962862 0.39240827]
You can see the last fold improved the score of the original model—from 0.322 to 0.392. (See methodology section for details.)

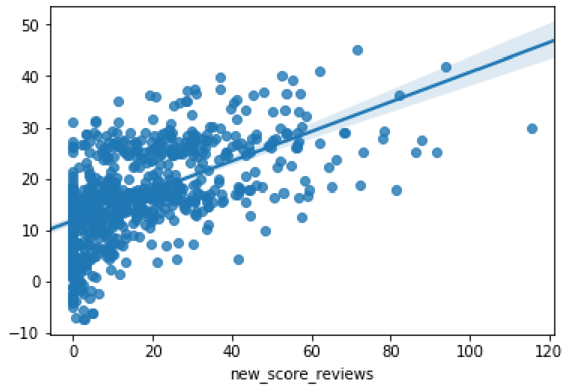


Figure 1. x=y_test, y=y_pred_lm

Lasso Regression and Bayesian Ridge Regression:
The steps for implementing these two models are very similar to *Linear Regression.* We just need to import different model from sklearn library. The procedure are pretty much standard like building the model, fit with training set, predict with test set, plot graphs, obtain scores and errors.
(Figure show on the top)

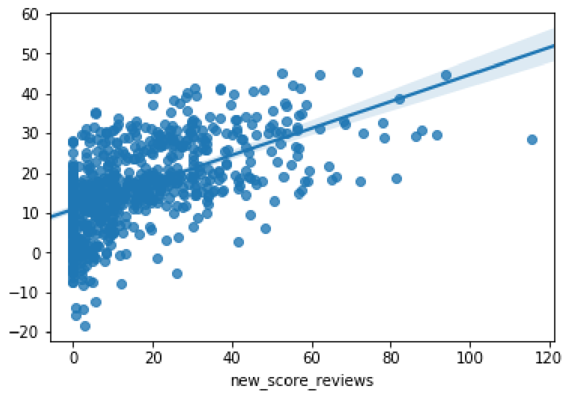


Figure 2. x=y_test, y=y_pred_lm_cv

Random Forest Regression:
Similar implementation procedure as previous models. However, this model has a special method call feature_importances_ which can show us the importance of each feature. (See methodology section for details.)

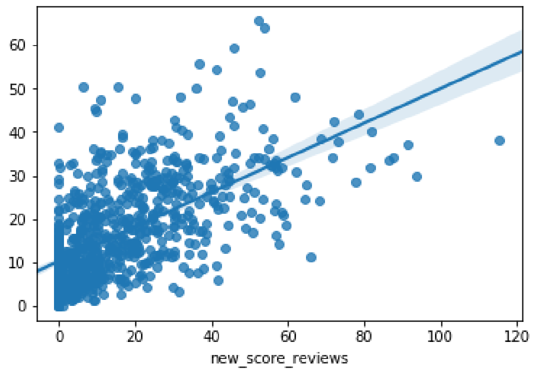


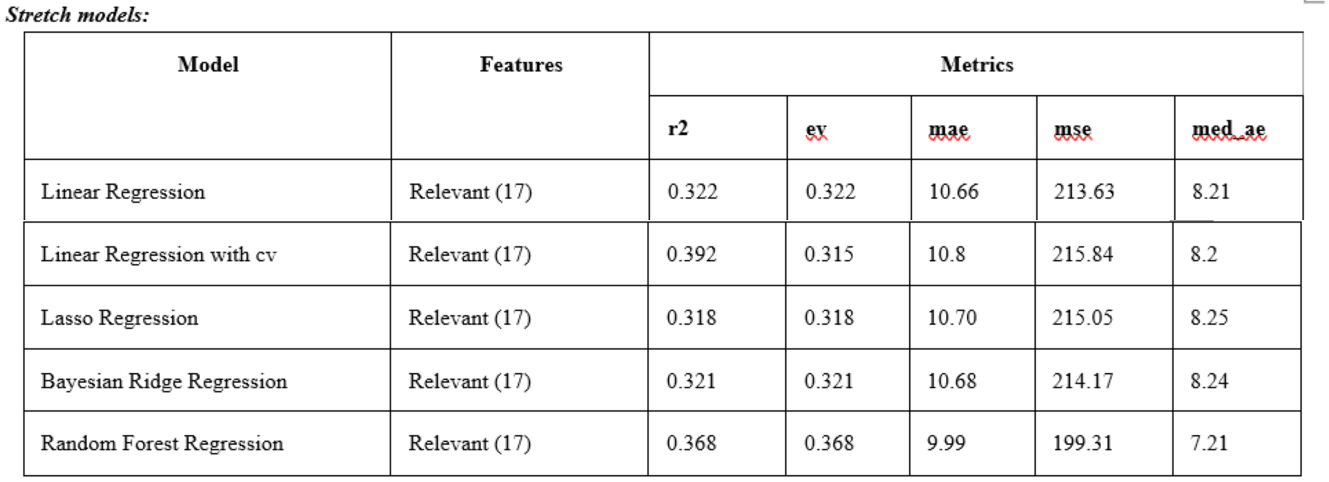
Figure 3. x=y_test, y=y_pred_rfr
Note: all the implementation is on jupyter notebook on our project repository (reference 1).

4. EVALUATION

Note: r2: r2_score, ev: explained_variance_score, mae: mean_absolute_error,
mse: mean_squared_error, mae: median_absolute_error

Baseline model:

Model	Features	Metrics				
		r2	ev	mae	mse	med_ae
Dummy Regressor (median)	Relevant (17)	-0.12	0	13.02	353.18	9.94



Stretch models:

Model	Features	Metrics				
		r2	ev	mae	mse	med_ae
Linear Regression	Relevant (17)	0.322	0.322	10.66	213.63	8.21
Linear Regression with cv	Relevant (17)	0.392	0.315	10.8	215.84	8.2
Lasso Regression	Relevant (17)	0.318	0.318	10.70	215.05	8.25
Bayesian Ridge Regression	Relevant (17)	0.321	0.321	10.68	214.17	8.24
Random Forest Regression	Relevant (17)	0.368	0.368	9.99	199.31	7.21

In general, all the stretch models are better than the baseline model based on r2 score, explained variance score, mean absolute error, mean squared error, and median absolute error (See Appendix for details). Linear Regression has a relative better result than Linear Regression with cross validation, Lasso Regression, and Bayesian Ridge Regression. Linear Regression has the lowest mae and mse among these four models. Random Forest Regression give us the best result among all models. It has high r2 score and low mean absolute error which means prediction has less error for future data.

5. CONCLUSION

Recommendations for low performers:
Based on coefficient of each feature from linear regression model and feature importances in random forest regression. We conclude the following 9 features is crucial to host's earning:

zipcode, bedrooms, price, cleaning_fee, host_response_time, host_is_superhost, host_identity_verified, instant_bookable, cancellation_policy

Provide more bedrooms to rent. Set reasonable market price and keep cleaning_fee minimum. Be responsive since most of the top performer hosts always giving response within an hour. Be a superhost since it's the status and recognition from the Airbnb which implies good reputation. Activate the instant bookable features to allow instant booking. Make sure your account is verified by Airbnb to show reliability. Many low performer accounts have not verified by Airbnb. Last, make flexible cancellation policy.

Discussion:
1. Is the new feature (new_score_reviews) a good indicator for host's earning?
It is still questionable whether the new feature (new_score_reviews) is a good indicator as the target variable to predict host's earning. We just assume it is reliable and related host's performance.

2. Future work: Incorporate top and low performance threshold to build models.

6. MILESTONES

Date	Task	Progress
01/23/19	Submit a project proposal. (1 page)	Done (100%)
02/06/19	Collect the dataset, begin to implement the baseline models.	Done (100%)
02/15/19	Complete the baselines & submit a progress report (3 - 4 pages).	Done (100%)
02/20/19	Implement the stretch models.	Done (100%)
02/27/19	Complete the stretch models & begin the evaluation.	Done (100%)
03/06/19	Finish the evaluation & work on the poster and the final report.	Done (100%)
03/14/19	Poster presentation.	Done (100%)
03/24/19	Submit final project report (5 - 7 pages).	Done (100%)

7 REFERENCES

1. Project repo: https://github.com/tonyman316/airbnb
2. Kaggle idea: https://www.kaggle.com/yogi045/how-to-become-top-earner-in-airbnb/
3. Scikit-learn baseline: https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyRegressor.html
4. Scikit-learn regression: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

8 APPENDIX

Note: Mean absolute error (mae) is calculated as an average of absolute differences between the target values and the predictions. It penalizes huge errors that not as that badly as MSE does so it is not that sensitive to outliers as mean square error. Math equation:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Note: Coefficient of determination R2 is the fraction (percentage) of variation in the response variable Y that is explainable by the predictor variable X. It ranges between 0 (no predictability) to 1 (or 100%) which indicates complete predictability. A high R2 indicates being able to predict response variable with less error.