RELATIONSHIP BETWEEN AIRBNB HOUSING FEATURES AND ITS OCCUPENCY RATE

Kanshuai Wang	**Fanghao Zhong**	**Haotian Guan**	**Bolin Yang**
Center for Data Science	Center for Data Science	Center for Dsta Science	Center for Data Science
New York University	New York University	New York University	New York University
New York, NY 10012	New York, NY 10012	New York, NY 10012	New York, NY 10012
kw2606@nyu.edu	fz477@nyu.edu	hg1580@nyu.edu	by641@nyu.edu

March 12, 2020

ABSTRACT

As home-stay style accommodation gradually becoming more and more popular, people start getting used to looking for Airbnb housing while traveling and visiting. Sometimes, it is hard to find ideal Airbnb housing in desired areas due to the imbalance between supplying and demanding of Airbnb housing. In order to attract more house owners to post their housing online, we conduct several models to predict whether the housing would have a relatively higher occupancy rate given other housing features, and suggest what kind of features should be valued by house owners in order to increase the occupancy rate. Our model contains Logistic Regression, Decision Tree, K-Nearest Neighbors, Fully Connected Neural Network, Random Forest and Extreme Gradient Boosting. Through tuning the hyper-parameters for each model, we are able to improve the predictive ability of our model.

1 Business Understanding

According to the report from Douglas Quinby, the growth of private accommodation market has been having a better performance than the growth of total travel market from 2013, and the private accommodation market has already reached 36.6 billion dollars worldwide by 2018 [6].

With this prevailing trend of home-stay, people would like to post their houses online and rent them to travelers or business people for a short period. Nowadays a lot of companies, providing the platform for the homeowners and travelers, have been well known. The majority researches claim that the private accommodation market still have enormous potential, which means, a bunch of houses could be listed on the home-stay platforms to enlarge the private accommodation market, especially in some popular tourist destinations.

Among all the online home-stay platforms, Airbnb is the most illustrious one. Susquehanna International Group

reported that Airbnb, globally, accounted for about 15% [1] of room nights in the private accommodations market. Taking Airbnb as an example, sometimes, it is still difficult to find a satisfying house on Airbnb in the desired area, such as New York City. One reason why some house owners dislike putting their houses on the platforms is that they cannot directly learn how much money they can earn with their listed house.

Thus, in order to attract more proprietors to list their house on its platform, our team plans to set a classification model for Airbnb, given all other house features, to predict the expected occupancy rate of a house and also find which features potentially contribute more to the occupancy rate in our model. The using scenario for Airbnb is that it can utilize our model to provide the range of occupancy rate to its new posters in its system. This would help Airbnb to improve the competence compared with other platforms. Meanwhile, the homeowners can use this information to make sure the approximate profit they can get if they post their houses online. Another using scenario is that, according to the importance of features in the model we build, some proprietors, whose listed houses have a relatively low occupancy rate, can know how to improve their properties to attract more renters. To build our occupancy rate classification model, our group will experiment on six models: Decision Tree, Logistic Regression (LR), Random Forest, K-Nearest Neighbors (KNN), Neural Network and XGBoost.

2 Data Understanding

2.1 Data Source

For the potential listing rooms, we want to forecast how well it prepared and whether the room will be accepted by the market, so we want to get data of Airbnb housing including their past listing rooms' amenities, locations, prices, etc associated with their reviews/occupancy rates. We get our data from `http://www.insideairbnb.com`, which is an independent, non-commercial, open source data tool to explore how Airbnb is used in different cities.

The data is published monthly and formed in three parts: listings, reviews, and calendar. In listings, a data instance includes information about listing (url, id, last scraped, etc), host (response time, id, location, etc), property (type, amenities, square feet, etc), neighborhood (location, transit, market, etc), price (nightly price, cleaning fee, security deposit, etc), review score (accuracy, cleanliness, communication, etc), policy (cancellation, minimum night, instant booking, etc), and availability (past 30 days, 60 days, 90 days, etc). In reviews, a data instance includes listing id, review date, and review content. In calendar, a data instance includes listing id, date, and availability on that date.

We select 2018 data in New York City, NY as our training data and we use new listings in 2019 which is not available in 2018 to evaluate our models. We assume that the host does not change listings within a year, so we pick the last available month's listings in the year. In 2018, we reduce total listings from 81472 to 58228, and we apply the same filters in 2019. Detailed listing reduction procedure will be described in Section 3.1.

2.2 Selection Bias and Reliability

Data leakage is easily neglected while checking data reliability. We use new listings in 2019 which is not available in 2018 to evaluate our model, so we do not encounter data leakage problem in our settings.

Geographic location is an important selection criteria in Airbnb rooms or hotels. In our data, the distribution of total number of listings by boroughs (Figure 1) are similar

to airbnb supply distribution by borough in 2015 (Figure 2) published by Airbnb, so our data is reliable and does not suffer selection bias.

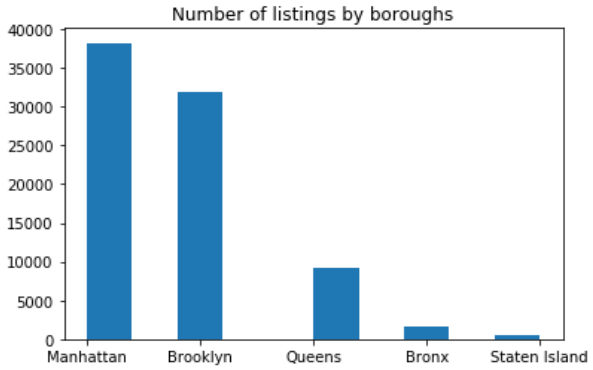


Figure 1: Number of total listings by boroughs

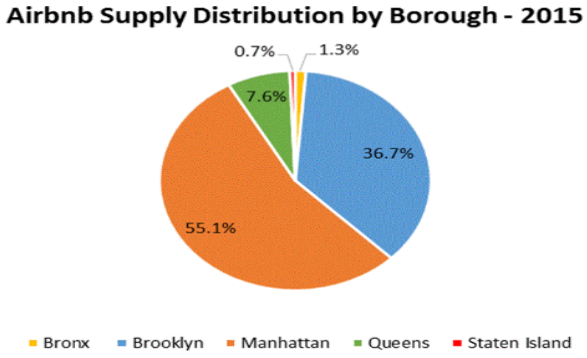


Figure 2: Airbnb Supply Distribution by Borough - 2015 [7]

3 Data Preparation

3.1 Data Cleaning and Feature Engineering

After merging the data and using the last month data for each Airbnb housing listing in 2018, we get 81472 data points with 97 unique features including id associated with each listing. Using domain knowledge selection, we select 34 related variables. Among which, there is a variable called "amenity" containing different features of the housing that have been verified by Airbnb. Using Bag of Words method on this variable, we are able to add 135 additional variables. Among all numerical features we have, we drop the missing data in the feature if the missing rate is lower than 1%, if it is above 1%, we use the mean of the feature value to fill the missing value. And then standardize all the numerical features. For all categorical data, we use one hot encoding method to indicate whether the data point contain such feature. After feature engineering, we have 58228 data points with 120 features.

3.2 Feature Selection

After the data cleaning step, we realize that using the Bag of Words will leave us a sparse data, and there are too many features in the data, which is not good for model prediction. Thus, we need to drop all variables except the features that will best explain the variation in the data. Here comes the PCA(Principal Component Analysis). After applying PCA on our standardized data, we realize that if we only use 120 features out of 220 features, we could explain 80% of the variance in data, which is surprisingly good as it drops nearly half of the features. Figure 3 shows that cumulative explained variance as the number of principle component goes up. As a result, we uses a base model Decision Tree Classifier to find the feature importance for each feature, and then select the top 120 features that has relatively high feature importance.

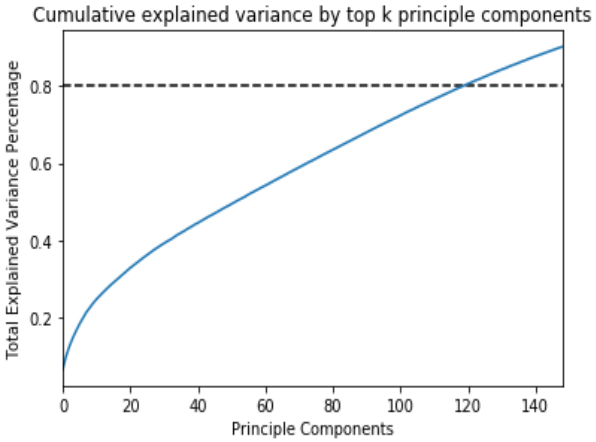


Figure 3: Cumulative explained variance

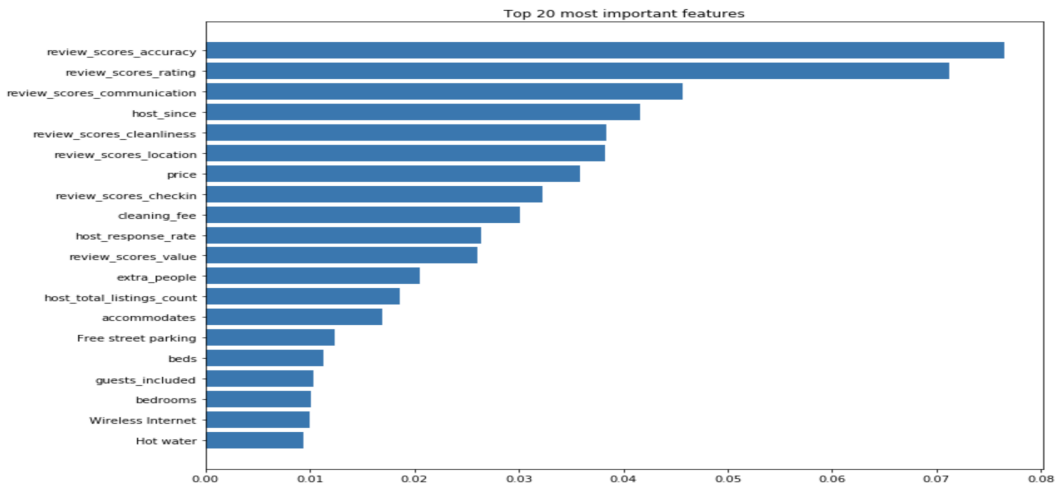


Figure 4: Cumulative explained variance

3.3 Target Variable

Our goal is to predict the classes of occupancy rate for new Airbnb listings based on the features they have. Since we cannot get such variable directly from our data set, we plan to generate it from other features in our data. For obtaining this variable, we have to make several seasonal assumptions. First, we assume the true number of nights customers stay in the housing should be proportional to the number of reviews customers leave for the Airbnb listing in any fixed period[5]. Second, because of no public statements were made about average stays, we assume that, if no minimum night stay is required, three nights are booked by customers for each review. Also, we use the number of available days in a year to normalize the number of reviews in the same period for each Airbnb listing. If no available days in this year, we remove such listing data because this situation is equivalent as not posting their houses on the platform. After computing the occupancy rate, we capped the number at 70%. The reason of doing so is to remove the outlier bias in this variable, such as housings with extremely high review rates, changed minimum nights, and relatively small numbers of available days. We then realize that the exact occupancy rate number is not useful for the new Airbnb listings. It would be more meaningful if we can provide the new listing owners the information whether or not their properties would have a lot of customers. Thus, we divide and bin the occupancy rate number into three classes. If the occupancy rate is 0, we label it as 0, which means, the occupancy rate is extremely low. If the occupancy rate is between 0 and 0.7, we label it as 1, meaning the occupancy rate is moderate. And when the occupancy rate is 0.7 (capped), we label it as 2, as the occupancy rate would be excellent. After labeling the data, we plot a graph about the distribution of the labels (Figure 5)). It seems relatively balanced, with slightly more data towards the label 2 occupancy rate.

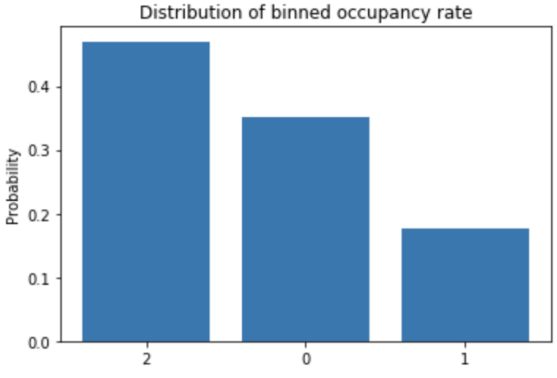


Figure 5: Distribution of binned occupancy rate

3.4 Train, Validation and Test set

Our training and validation sets come from the 2018 Airbnb data, with a split ratio of 75:25. For both sets, we ensure that all of the data have unique id in order to prevent the data leakage issue. For the test set, we gather the data from the 2019 Airbnb data, with only the new Airbnb listing owners' data. Since our goal is to predict occupancy rate classes for people who would like to post their houses online, we want to test whether our model could reach a high AUC score on the new posters. Both 2018 and 2019 data set follow the procedure we describe in Section 3.1.

4 Modeling and Evaluation

4.1 Evaluation Metric

The selection of evaluation metric is closely related to the deployment of models in practice. Not only accuracy of models must be considered, but also real-life meaning should be evaluated. Hence, most commonly used evaluation metrics for classification problem is Area Under the ROC Curve(AUC). AUC clearly explains how much model is capable of distinguishing among classes. A classifier with higher AUC indicates that it can have more true positives with the same number of false positives, and therefore more accurate the model is. Since we have three unique class labels in our target variable. In order to calculate AUC score, we need to apply the One-vs-All label binarizer to our target values.

4.2 Baseline: Decision Tree

In order to make sure that each model we use have some predictability, we create a baseline model with decision tree. We also apply decision tree to obtain feature importance and conduct dimension reduction according to principle component analysis (PCA), which is mentioned in the section of data cleaning. Since it is a baseline model, we just use the default setting of decision tree to compare with other models. The default setting of decision tree generates accuracy of 0.6386 and AUC of 0.7008.

4.3 Logistic Regression

Logistic regression is a commonly used model in classification problem. It is proven to be robust to out-of-sample multi-class prediction, and it is also proven to be reliable to small samples. In logistic regression, we will tune the penalty between 'L1' and 'L2' penalties and regularization C. With L1 penalty, we will retrieve sparse solution with heavy regularization, but with L2 penalty, it computes more efficiently than L1 penalty, and we will retrieve solutions that have small norm with heavy regularization.

4.3.1 Tuning Parameters

Penalty and Regularization: In Logistic regression, we do not use greedy search method to tune hyper-parameters. With default settings of logistic regression in sklearn package, we use the combination of penalty ['L1','L2'] and regularization [0.0001, 0.001, 0.01, 0.1, 1, 10, 100] to train the model using same training data and calculate AUC score using same test data. The results show in Figure 6. Obviously, L2 penalty performs better than L1 penalty with our dataset, and with thumb-of-rule, simple model is better, we set the regularization term to be 0.01.

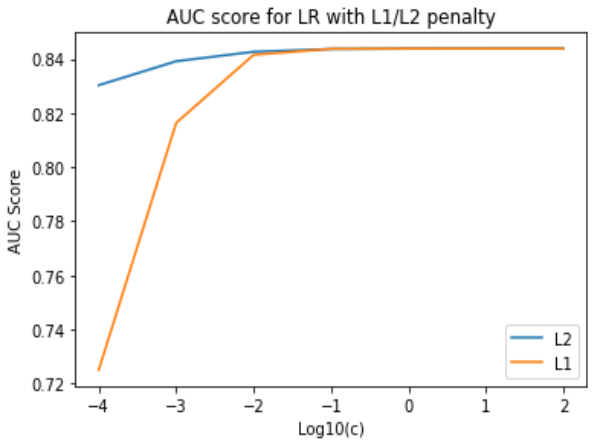


Figure 6: Logistic Regression

4.4 K-Nearest Neighbors(KNN)

K-Nearest Neighbors is a supervised machine learning algorithm which can solve both the classification problems and also regressions. However, commonly, it is used widely as a classification algorithm, and it is proven incredibly effective in some cases. We utilize the KNN classification as one of our algorithm to estimate our target.

4.4.1 Tuning Parameters

K Neighbors: Applying KNN to classify different categories of occupancy rates, we receive the accuracy of 0.6595 with the default parameter which is 5 for k-neighbors value, and the correspond AUC value is 0.7901. K nearest neighbors classifies a sample based on the class of its k nearest neighbors, so the only parameters we would change is the number of k so that we can find the best AUC in this algorithm. After trying a bunch of k numbers, we get the AUC trend in figure7. AUC would increase sharply when we increase the k numbers from 1 to 20; then, the growth of slope slows down. When the k number is bigger than 80, the AUC value would fluctuate in a tiny range. For keeping the complexity of the algorithm from a high level, and at the meantime, guaranteeing the AUC is at its peak, we choose 80 as our k number. The AUC at that point is 0.8314, the accuracy is 0.6974.

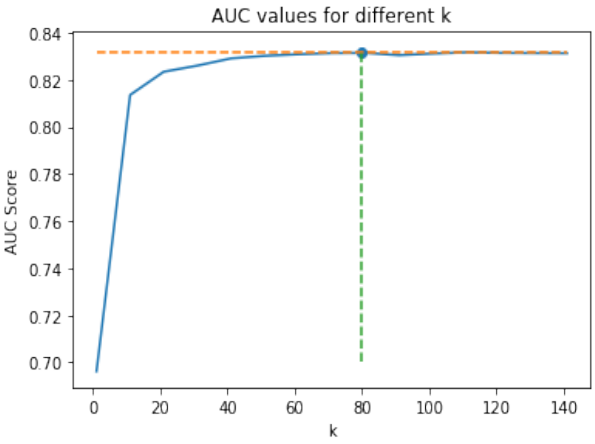


Figure 7: K-Nearest Neighbors

4.5 Random Forest

Random forest algorithm is essentially a collection of decision trees, which has been proven to have lower variance than decision tree method by averaging multiple deep decision trees and training on different parts of data. Also, machine learning algorithms need to be transferred to new data. Random forest can efficiently overcome the overfitting problem which frequently met in real life, because random forest randomly samples training observations and randomly subsets features for node splitting [4].

4.5.1 Tuning Parameters

Criterion: With default setting in random forest, we train the model separately using Gini and Entropy (information gain) with same training data and apply to the same test data. We achieve a higher AUC score in Gini setting. Thus, we set the splitting loss criterion to be Gini index. **:**

min sample split and min sample leaf: The min sample split represents the minimum number of samples required to split an internal node. The higher min sample split is, the

more constrained the model is because it considers more samples at each node. Thus, this parameter can help prevent the occurrence of over-fitting. Min sample leaf is the minimum number of samples required to be at each leaf, which is similar to min sample split. Hence, we tune min sample split and min sample leaf together to find the best combination of them. We first try to use five equal-length intervals between 5% of the training data and 35% of the training data as min sample split, and five equal-length intervals between 2.5% of the training data and 15% of the training data as min sample leaf. The figure below shows the AUC of random forest models using different combination of min sample split and min sample leaf. It is clear that AUC is higher around 5% to 15% of training data as min sample split and 2.5% and 7.5% of training data as min sample leaf. We therefore enlarge the specific interval between 1% and 0% of training data and between 1% and 5% for min sample split and min sample leaf respectively. Figure 2 illustrates AUC using the new interval of min sample split and min sample leaf. We can see that the highest AUC occurs when we use 2% of training sample and 1% of training sample as min sample split and min sample leaf respectively, which means min sample split is 873 and min sample leaf is 437. Random forest with Gini as criterion, 873 as min sample split, and 437 as min sample leaf returns accuracy of 0.7072 and AUC of 0.8637.

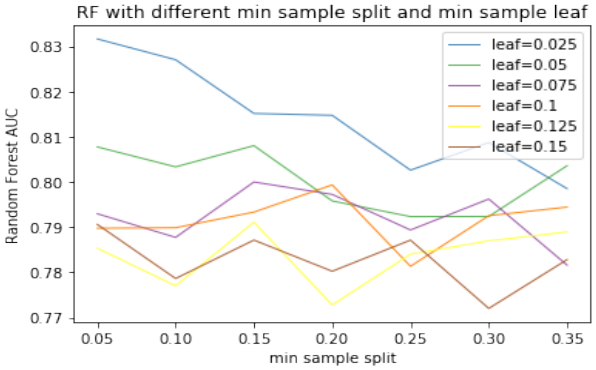


Figure 8: Initial Parameter Interval for Random Forest

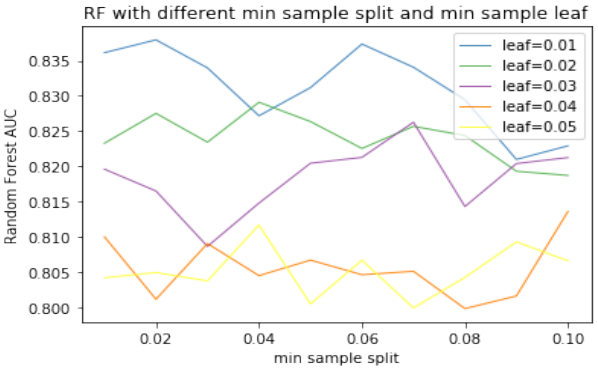


Figure 9: Adjusted Parameter Interval for Random Forest

4.6 Neural Network

Neural network (NN) is a popular, effective, and common way for large-scale classification problems. And the most simple neural net (NN) architecture uses one hidden layers with 100 nodes in the hidden layer [2]. In our project, we will apply a classical two fully connected (FC) hidden layers neural network (NN), and the hidden size will be determined in the tuning process. The understanding of the structure of two hidden layers network is that the first hidden layer acts as the embedding layer and the second layer is regarded as the predicting layer. Each hidden layer will be followed by linear transformation function (ReLU, Logistic, Tanh, Identity), and the linear transformation function will also be determined in the tuning process. The model will use default solver Adam, default L2 penalty 0.0001, and default learning rate 0.001. The reason we do not turn our learning rate is that we assume the adaptive learning rate will automatically find a good learning rate in optimisation process. After the hidden layers, it follows a softmax activation that is used for predicting the classes.

4.6.1 Tuning Parameters

Hidden layer nodes: Hidden layer node is an important feature in neural network architecture. Too small hidden layer nodes cannot sufficiently group all information from

inputs, and too large hidden layer nodes compute inefficiently and may result poor performance. There are a couple thumb-of-rule to pick hidden layer nodes [3]:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

Thus, the hidden layer nodes are first tuned among [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140]. The AUC score vs Hidden Layer Nodes shows in Figure 10. By greedy search, around 20 hidden layer nodes look good, then the hidden layer nodes are tuned among range(5, 20) (Figure 11). By comparing AUC scores and following principle of simple model is better, we set the hidden layer nodes to be 14.

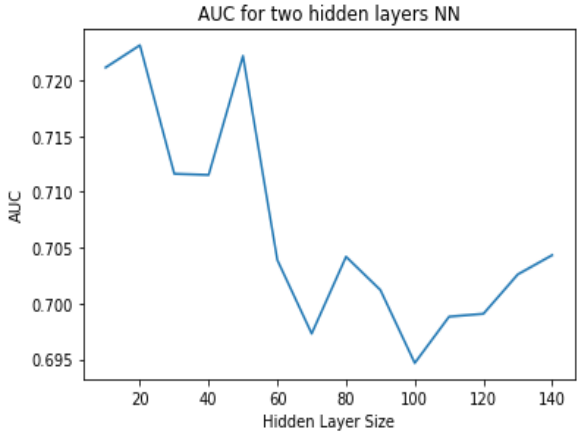


Figure 10: AUC vs Hidden Layer Nodes (1)

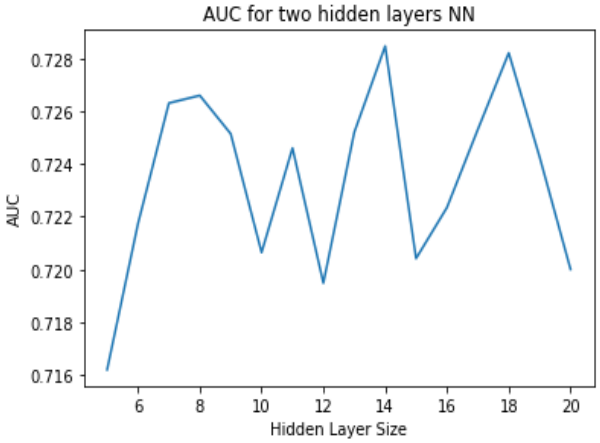


Figure 11: AUC vs Hidden Layer Nodes (2)

Activation Function: After setting hidden layer nodes, we want to find best activation function among Logistic, ReLU, Identity, and Tanh. Applying same procedure as tuning hidden layer nodes, we find Logistic function performs best among all activation functions (Figure 12). Thus, we set the activation function to be Logistic.

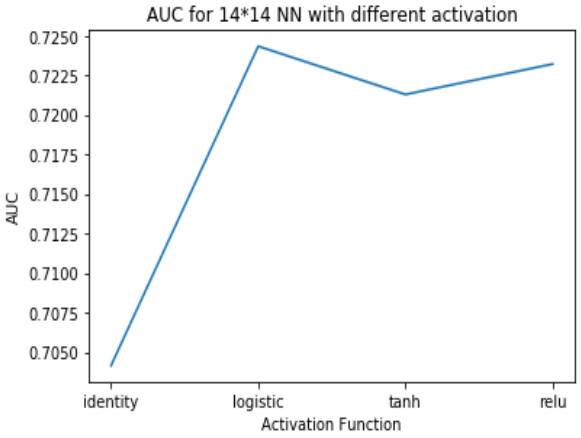


Figure 12: AUC vs Activation Functions

4.7 Extreme Gradient Boost

Boosting is a method to convert weak learner(i.e. decision tree method) into strong learner by fitting a modified version of the original dataset in each step. Comparing to the previous step, the new step focuses on the part of data that

is difficult to classify by assigning a higher weight to it. so that we can improve upon the prediction of previous step. Gradient boosting identifies the shortcoming of previous step using gradient in the loss function. We decide to apply Extreme Gradient Boost to improve our baseline decision tree model. Since we have three class labels, we choose "multi:softmax" as our model objectives.

4.7.1 Tuning Parameters

min child weight: Minimum sum of instance weight (hessian) needed in a child. The larger "min child weight" is, the more conservative the algorithm will be. Since our test sample are the samples that might not even seen before, we want to make sure that the algorithm should be robust, thus we have tried value range from 0 to 10.

gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. Same reasoning as it in min child weight, we use small values [0.5, 1, 1.5, 2, 5]

max depth: Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. We try out small values like [3, 4, 5, 6, 7], preventing the model from being too deep, thus decreasing the generalization error

By grid searching the variables from min child weight, gamma and max depth, we found that the best parameter for Extreme Gradient Boost is min child weight =5, gamma=5, and max depth =6, with accuracy=0.7468 and AUC=0.8710.

4.8 Model selection and Evaluation

We use the default parameters in each model to test on whether using PCA to reduce about half of the features will reduce accuracy on our model prediction. We put the accuracy score for both data with all features and data with only 120 features for all models in Table 1. We can see that there is small variation in the accuracy without a major shift. Thus, we believe that PCA in Section 3.2 does help to reduce the data dimension without losing much information.

Model/Accuracy	All variables accuracy	120 variables accuracy
Logistic Regression	0.7059	0.7043
Decision Tree	0.6396	0.6386
KNN	0.6541	0.6591
NN(2 layers)	0.6537	0.6534
Random Forest	0.7017	0.6992
XGBoosting	0.7138	0.7131

Table 1: Model accuracy

We had a Decision Tree baseline model with AUC=0.70. After fine-tuning the hyper parameters for all other 5 models we have, we compute the AUC for each model. Table 2 shows the comparison between the AUC score before and after fine tuning. We can see that there is at least 0.1 increase in the score, with highest AUC score from XGB, 0.8719. Based its highest AUC score, we decide to use this fine-tuned XGB model to estimate the occupancy rate class for our new 2019 data.

Model	AUC before tune	AUC
Logistic Regression	0.8437	0.8426
KNN	0.7901	0.8314
NN(2 layers)	0.7085	0.7288
Random Forest	0.8547	0.8637
XGBoosting	0.8625	**0.8719**

Table 2: Model AUC

Using all the data in 2018 as the training data for the fine-tuned XGB model, we evaluate it on the 2019 data and get an accuracy of 0.7339 and AUC of 0.7826, which is slightly lower than the 0.8719 of the validation AUC from 2018 due to generalization error. Based on this AUC score, we can say that our model is robust and able to predict occupancy rate class for new Airbnb housing owners. One potential reason why the AUC score drops is that since we are only able to obtain 9 months data in 2019, which

excluding the housing data in the winter season. Which might leads to incorrect data distribution, and thus less likely to obtain correct results. The top 6 important features in determining the occupancy rate class is shown in figure 13. We can see from the figure that in order to increase occupancy rate, housing owner should increase their quality in the general setting of the housing, checking service, proper location, and including hot water, free street parking and wireless internet as housing condition.

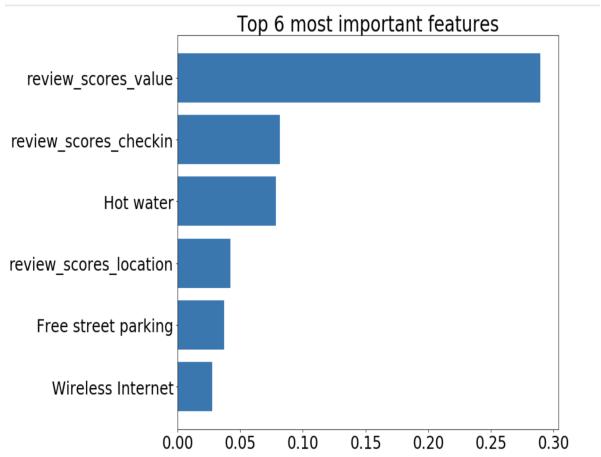


Figure 13: Top 6 feature importance

5 Conclusion and Deployment

For setting our model adequately, we made several assumptions and calculate the occupancy rate as our target feature. However, these assumptions might not be fully reflected in the real life situation. If any people or firms would like to use our model, they should be aware of such issue. we separate the occupancy rate to three categories and select our features which can best explain the variance of our target feature with PCA. Next, several different algorithms have been applied to make our classification model, which include Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbors (KNN), Neural Network (NN) and XGBoost (XGB). Comparing with others, XGBoost has the best performance and obviously beat the baseline algorithm.

Even though our model cannot predict the specific occupancy rates of listed houses on Airbnb, the estimation of the range of occupancy rates works well enough when we test it with the validation data (New listings in NYC in 2019). The estimation of the occupancy rate range is economically worthwhile, for it can directly show to the homeowners the economic values and attractiveness of their properties. Therefore, our model catches the goals we mentioned at the beginning of this paper. Airbnb can apply our model to estimate the classification of the occupancy rate of posters' house and attract new hosts to use its platform. Also, the importance of features to predict our target is retrieved by our model.

Also, there is a risk or bias we do not solve in this model. Most of the top important features we retrieved in our models are about reviews, which would give hosts incentive to pursue higher review scores by purchasing fake review scores in market, which would be unfair to hosts who do not purchase the fake review scores. For example, hosts may pay people to book their rooms on Airbnb and leave them a high review score. With a great review score, hosts are easier to rent their rooms to real renters, then renters may be cheated to stay at a terrible room but having a great reputation on Airbnb.

6 Future Work

Furthermore, the user of this model can update it frequently by keeping adding data to fit the changes of the preference of Airbnb users. Meanwhile, as the accumulation of our data set of other places, our model can enlarge to a more comprehensive one which would fit to different types of areas, not only the New York City or similar areas. Addi-

tionally, if we can collect the data of a long-term period in the same place, we can apply the seasonal oscillation in our model and improve our model a lot. Also, from Airbnb's data we saw, there are lots of picture data included in each listing. In intuition, when customers look for rooms on Airbnb, the first thing they will see is the pictures of rooms. So if we have more time, we can apply Computer Vision (CV) techniques to retrieve additional features from pictures. After consummating our model, it is possible to extend this model to more applications, such as automatically analyzing what should be improved for a specific property and showing it to its owner or fitting this model to other platforms.

References

[1] Tyler Cain. *Perfecting Private Accommodations: Airbnb vs. Expedia*. URL: https://www.toptal.com/finance/market-research-analysts/airbnb-vs-expedia.

[2] Sklearn Documentation. *sklearn.neural_network.MLPClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[3] Jeff Heaton. *The Number of Hidden Layers*. URL: https://www.heatonresearch.com/2017/06/01/hidden-layers.html. (Published: 06.01.2017).

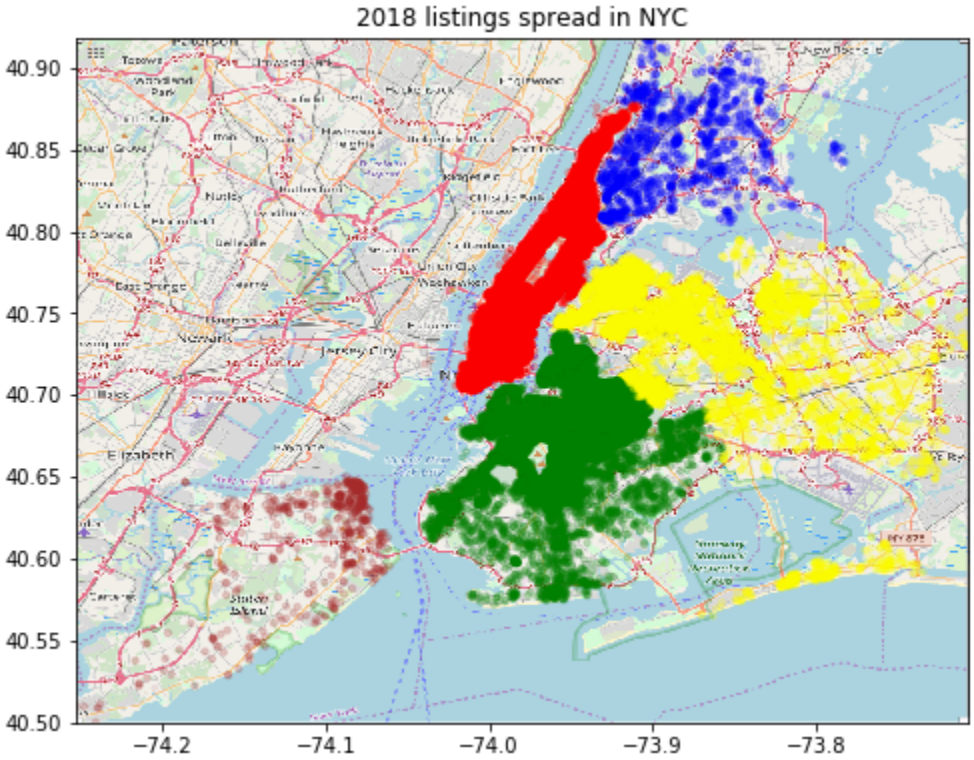
[4] Will Koehrsen. *An Implementation and Explanation of the Random Forest in Python*. URL: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76. (Published: 08.30.2018).

[5] Yi Luo. *What Airbnb Reviews can Tell us? An Advanced Latent Aspect Rating Analysis Approach*. URL: https://lib.dr.iastate.edu/etd/16403. (Published: 2018).

[6] Douglas Quinby. *U.S. Private Accommodation Market to Reach $36.6 by 2018*. URL: https://www.phocuswright.com/Travel-Research/Research-Updates/2017/US-Private-Accommodation-Market-to-Reach-36B-by-2018. (Published: 02.2017).

[7] Claudia Alvarado Stephen Hennis Jessica Haywood. *STR: Airbnb's impact on NYC's boroughs*. URL: http://www.hotelnewsnow.com/Articles/30455/STR-Airbnbs-impact-on-NYCs-boroughs. (Published: 02.24.2016).

Appendix A


2018 listings spread in NYC
40.90
40.85
40.80
40.75
40.70
40.65
40.60
40.55
40.50
−74.2
−74.1
−74.0
−73.9
−73.8