Title Block

BUDT758T-0506, Spring 2021
Group 04
Group members: Yuchen Chen, Tzu-Chieh Chen, Jiaqi Huang, Yu Zhang
May 02, 2021

Executive Summary

Airbnb business operators may want to test the customer's evaluation of airbnb based on different variables and different models, so as to obtain the accuracy of the model and adjust the operation of airbnb. In this regard, we have cleaned up and sorted out the data to ensure that some abnormal data will not have a negative effect on the accuracy of the model. We selected some variables that are more critical and conducive to model investigation, such as price and accommodates etc. In order to make the model more accurate, we merged a piece of external data and selected two variables that are not repeated. We made 6 different models to compare one another, attempting to select which model can achieve higher accuracy with high TPR and FPR that is no more than 10%. The six models were tree, knn, ridge and lasso, as well as random forest and xgboost. For these six models, we set different parameters to determine which parameter can make the model achieve a higher prediction accuracy. With a plenty of attempts, we found that out of six models, ridge, lasso and random forest models usually gave us relative higher accuracy, so we built data frames with cutoff, TPR and FPR for these three models, which can help us to select the highest TPR with FPR<0.1 simultaneously. The random forest model was the best. We also evaluated the performance of six models with ROC and AUC, and the results proved that the random forest model we built previously was the best model in terms of ROC and AUC outputs. As a matter of fact, we have to admit that choosing an optimal cutoff is very challenging as FPR of our several submissions exceeded 0.1 threshold. Accordingly, we decided to choose a more conservative cutoff with FPR closer to 0.09 for our final prediction, although our final TPR is not the perfect one, at least the final prediction meets the requirement.

Exploratory data analysis/feature engineering

For feature engineering, we focused on cleaning the data. There are so many messy columns.

First, we dropped the columns that do not need to be used for both train data and test data, such as “X1”, “zipcode”. For numeric columns, we replaced NAs with the mean of the data in that column, such as “accommodates”, “availability_30” etc. And for categorical columns, we grouped the features and replaced NAs with “Others” or “Unknown”. Some variables are quite troublesome. For example, the case of some variables is not uniform, or punctuation is added, but actually the features of many variables are originally the same thing, the results are separated. Thus, we used case_when() to unify these chaotic features into one feature.

There are some tricky columns like amenities and host_verification, which have a list of various values, so we decided to choose some of the important values from the list for our prediction, such as TV, wifi, air conditioning in amenities and email, phone in host_verification. We considered that these amenities are all attributes for which people choose the airbnbs may consider, and we also created new columns for each informative attribute and used “TRUE” or “FALSE” to decide whether a row contains this attribute or not. Furthermore, we found there are some values, such as “NY” and “ny”, in the market and state columns; these values seem to be different, yet they are all the same things, so we used summary function to find them out and used case_when() to group some of them together in order to reduce different categories. For the text columns, we tried to convert these four text columns into a bag of words representation using RTextTools function create_matrix() and used the removeSparseTerms parameter as well to control the number of words that ended up in our matrix. Next, we converted the resulting sparse matrix into a data frame and concatenated it with our feature matrix. We changed “removeSparseTerms” to make sure that each matrix

brought the number of columns between 20 and 30. Moreover, we found the most frequent words for some variables, such as description, summary and name, were highly similar, so we only choose one of them to remove duplicate words in our dataset. In addition, we combined these four newly created matrices to the original dataframe and again removed duplicate values.

Lastly, we created dummy variables for training data and validation data as well as added some interactive dummy variables. For interactive variables, we sorted out the variables that can most possibly influence the target variable. Some of the top variables are price, cleaning_fee, extra_people, maximum_nights, minimum_nights, security_deposit, accommodations, host_response_rate, beds and guests_included. We chose some of them and matched those we thought they're correlated and made them as our interactive variables.

The external Airbnb data we found from Kaggle (https://www.kaggle.com/kritikseth/us-airbnb-open-data). We extracted two variables "Number_of_reviews" and "Reviews_per_month" from the Airbnb data, cleaning the data and combining them into the original cleaned training and validation data.

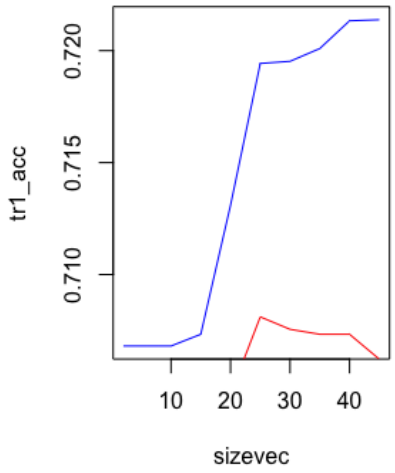
Finally, we got 100,000 observations and 126 variables in train data, 12,199 observations and 125 variables in validation data. After creating dummy variables, we got 100,000 observations and 219 variables for training dummy data, 12,199 observations and 219 variables for validation dummy data.

Model evaluation and selection methodology

1. Tree

For the tree model, we created a vector of pruned tree sizes depending on how many nodes the tree has. Then we plotted the accuracy of each tree size and found the best performed tree. (the plot indicates that our tree did a bad job on validation data).

```
#define a vector of pruned tree sizes
sizevec <- c(2,4,6,8,10,15,20,25,30,35,40,45)
numsizes <- length(sizevec)

#define vectors to store the training and validation accuracy
tr1_acc <- rep(0, numsizes)
va1_acc <- rep(0, numsizes)

#loop over each size in sizevec
for (i in c(1:numsizes)){

  #retrieve the size indexed by i
  treesize <- sizevec[i]

  #prune the tree to be treesize
  pruned_tree=prune.tree(full_tree, best = treesize)
```

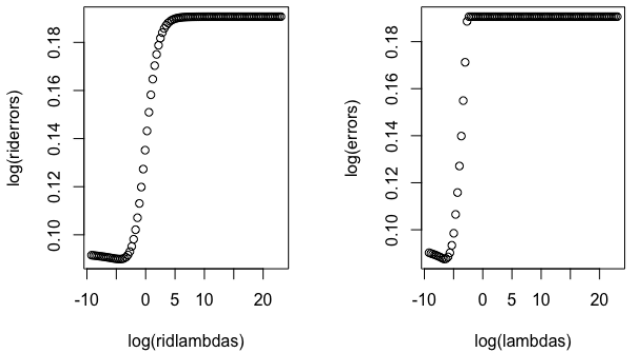


2. Lasso & ridge

For the lasso and ridge model, our first goal is to find the best lamba. We used the “glmnet package” and ran cv.glmnet with nfolds = 5 to secure the best lambda. Furthermore, we used this lamba to make predictions in the validation data. Our accuracy on lasso and the ridge model are similar.

Accuracy for the lasso model : 0.725

Accuracy for the ridge model : 0.723



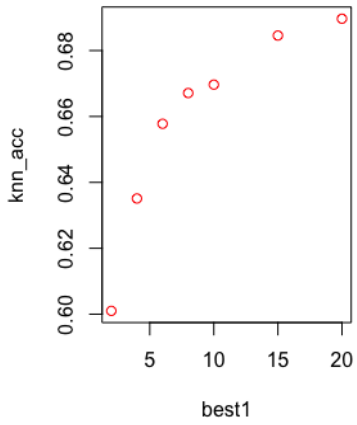
3. Random forest

For Random Forest, firstly, we did the model tuning to find the best mtry and ntree. From the result, we saw that the highest accuracy occurs when mtry=5 and ntree=1500. Thus, we decided to use mtry=5 and ntree=1500 to train our model, and used test data to predict our model. And the accuracy we got is about 72.5%.

```
[1] "Num trees =  500 , mtry =  3 , acc =  0.714666666666667"
[1] "Num trees =  500 , mtry =  4 , acc =  0.721666666666667"
[1] "Num trees =  500 , mtry =  5 , acc =  0.723666666666667"
[1] "Num trees =  1000 , mtry =  3 , acc =  0.715666666666667"
[1] "Num trees =  1000 , mtry =  4 , acc =  0.720333333333333"
[1] "Num trees =  1000 , mtry =  5 , acc =  0.722666666666667"
[1] "Num trees =  1500 , mtry =  3 , acc =  0.714666666666667"
[1] "Num trees =  1500 , mtry =  4 , acc =  0.72"
[1] "Num trees =  1500 , mtry =  5 , acc =  0.725333333333333"
```

4. Knn

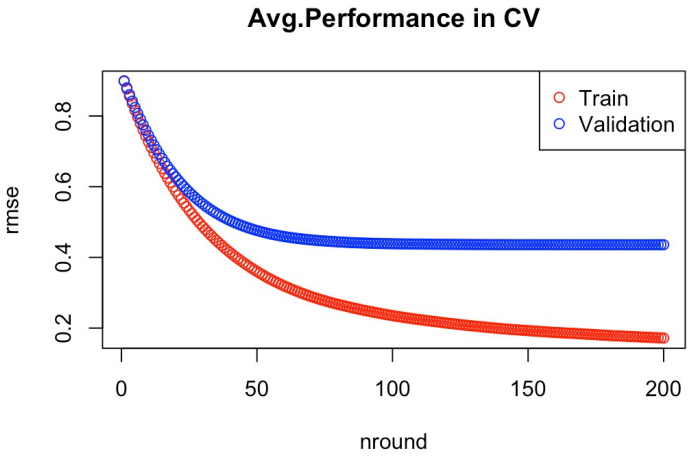
For KNN, we separated data into train and test, x and y, and we tried different values for k (k= 2, 4, 6, 8, 10, 20) in the Knn model, and it returned the accuracy for each k. The fitting curve we got is shown below. We found that when k=20, the accuracy is the highest. Thus, we used train.X, valid.X and train.y1 with k=20 into our best knn model. Lastly, we decided to use 0.5 as our cutoff for classification. The accuracy we got is about 69%.



5. Xgboost

For Xgboost, we initially converted our train and valid data to DMatrix for Xgboost. Then, we assigned a list of parameters to variable xgb.params, setting colsample_bytree to 1, subsample to 1, the booster we used was “gbtree”, max_depth to 12, eta to 0.03, evaluation metric to “rmse” and gamma to 0. After finishing setting parameters, we tried to use the cross-validation method for xgboost with 5-fold cross validation and 200 iteration to search the best round to apply it. Next, we plotted the valid and train rmse to check whether there is any overfit in this setting or not, and we found the best round in this model is 188, so we applied 188 to our xgboost model. The final accuracy of xgboost is 0.718.

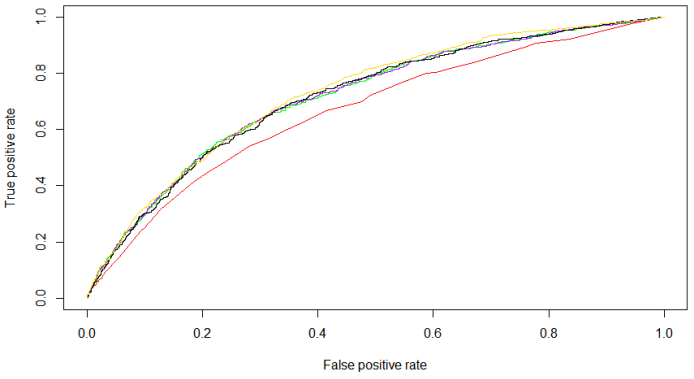
```
Multiple eval metrics are present. Will use test_rmse for early stopping.
Will train until test_rmse hasn't improved in 30 rounds.

[21]    train-rmse:0.575899+0.000834    test-rmse:0.618062+0.004128
[41]    train-rmse:0.407990+0.001714    test-rmse:0.500874+0.002401
[61]    train-rmse:0.316076+0.001936    test-rmse:0.457830+0.001038
[81]    train-rmse:0.265105+0.002808    test-rmse:0.443424+0.001139
[101]   train-rmse:0.233921+0.002132    test-rmse:0.438474+0.001574
[121]   train-rmse:0.213407+0.002518    test-rmse:0.436856+0.001907
[141]   train-rmse:0.197823+0.002240    test-rmse:0.436154+0.002140
[161]   train-rmse:0.187289+0.002531    test-rmse:0.435893+0.002231
[181]   train-rmse:0.178652+0.002486    test-rmse:0.435841+0.002394
[200]   train-rmse:0.171899+0.002819    test-rmse:0.435914+0.002319
```



Model specification comparison

In order to choose the best model, we generated ROCR performance and retrieve TPR and FPR for each model.

```
> performance(pred_tree, measure = "auc")@y.values[[1]]
[1] 0.6674638
> performance(pred_lasso, measure = "auc")@y.values[[1]]
[1] 0.7171532
> performance(pred_ridge, measure = "auc")@y.values[[1]]
[1] 0.717012
> performance(pred_xgboost, measure = "auc")@y.values[[1]]
[1] 0.7166502
> performance(forestpred, measure = "auc")@y.values[[1]]
[1] 0.7288078
> performance(pred_knn, measure = "auc")@y.values[[1]]
[1] 0.526512
```



As we can see on the charts above, our random forest model has the best performance. The next step is to choose the optimal cutoff value that would generate a low FPR and high TPR in our prediction.

```
> cutoffs_findrf <- cutoffs_rf$fpr < 0.1 #finding the fpr that below 0.1
> cutoffs_findrf
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [16]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [31]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [46]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [76]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [91]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[106]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[136]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[151]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[166]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[181]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[196]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[211]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[226]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
    cut        fpr      tpr
231 0.4 0.09586466 0.315367
```

Finally, we found that the best cutoff for our model is 0.4, which has a 0.095 FPR and 0.315 TPR. However, the FPR might be floating around 0.003 in the real test data, so we decided to use a higher cutoff to reduce the FPR.

Conclusions/takeaways

What did your group do well?

For data cleaning, we screened out effective variables and processed complex data. In terms of model building, we selected six effective models, compared different parameters, and used the performance of different parameters in the model to select the model with the highest accuracy. During the project, the members of our group communicated effectively and cooperated very well, and we did our own tasks efficiently. During this entire contest period, we turned in our prediction in time for almost every evaluation and wasted none of any opportunities we had. Everyone tried their best to tune the model after every evaluation and found the strategy for the next evaluation, hoping to improve accuracy and make models stronger and more reliable to compete with other teams.

What were the main challenges of this project?

The main challenge is that although we did a lot to work on data cleaning, added different interactive

variables, applied different models to predict, the results of accuracy and TPR didn't improved much since the first submission. In addition, sometimes we chose the FPR less than 0.1 and thought it was supposed to pass the FPR threshold but at last it didn't. In our opinion, it's hard to find a cutoff that we can make sure FPR<0.1 and not too far away from 0.1 as well, and with a great TPR at the same time.

What would you have done differently?

We would start to build the models, especially those that would take a long time to run, even earlier. Since every attempt for each model would bring different results, we would like to try as many times as possible to obtain a high TPR under the circumstance that FPR is less than 0.1. We should have grasped every single opportunity we had to submit the result since seeing the results from leaderboard can help us to make further improvement and polishment for our models and do more concise data cleaning.

What advice would you have for a student beginning this same project next year?

I would suggest they should check project guidelines and rubric carefully in the first place. As we hadn't read the rubric until the last two weeks before due, we did something wrong at the very beginning when we were cleaning the data. We had deleted all the text columns, but then we found we need to get at least three unstructured text columns to secure grade A. Moreover, we need to include an external data source and six different models in our work. It's fortunately that we noticed these grading rubric before deadline and still had time to fix all the problems, but it really wasted our time to make us add back some deleted columns.

What would you do if you had more time to work on this project?

We would add the parameter size and number of try for models such as random forest in order to train more different situations. We would also like to try different model types other than the six types we used. We would change interactive variables in more different ways to get a higher TPR. And we should

spend more time on researching the background knowledge regarding the advanced model such as xgboost, knowing how to tune the best hyper-parameters for it and understand the fundamental concept about it, making our model more useful and more reliable.

Appendix

Group member's main contributions:

Yuchen Chen: Final report writing, Model tuning, Model selecting.

Tzu-Chieh Chen: Final report writing, Model tuning, Model selecting.

Jiaqi Huang: Final report writing, Data cleaning, Feature engineering.

Yu Zhang: Final report writing, Data cleaning, Feature engineering.