



# Report 3:


# Tree-Based Models

Team 5: Yunhao Bai, Zhiyi Chen, Michelle Guan, Huiqiong Wu

BUS212a - Analyzing Big Data II

Prof. Arnold Kamis

Spring 2022



<b>1. Introduction</b>	<b>1</b>
<b>2. Dataset</b>	<b>1</b>
<b>3. Classification Tree</b>	<b>3</b>
<b>4. Regression Tree</b>	<b>6</b>
<b>5. Ensemble Methods</b>	<b>10</b>
<b>6. Summary</b>	<b>16</b>

## 1. Introduction

As housing is a basic human necessity, many people are concerned about apartment rental prices. Those that seek housing are interested in what they can afford with their budgets. Those that can provide housing are interested in what the market is willing to offer. Housing listings can be found all over the internet, and appropriate pricing of rental properties can be unclear without a basis of comparison. To better understand the condition of the rental housing market, this project aims to explore the relationship between pricing and features of a property, specifically apartment units. The variable of interest in this venture is apartment rental prices across the United States. This study will analyze the practical aspects that lead to the rental pricing among different types of apartments. By performing this study, suggestions can be produced for people with different expectations based on an apartment unit's location, size, offered amenities, etc. Previous reports in this project utilized multiple regression, logistic regression, and k-nearest neighbors methods, with the inclusion of polynomial terms and clusters. This report will focus on tree-based models to investigate the relationship between apartment price and property features.

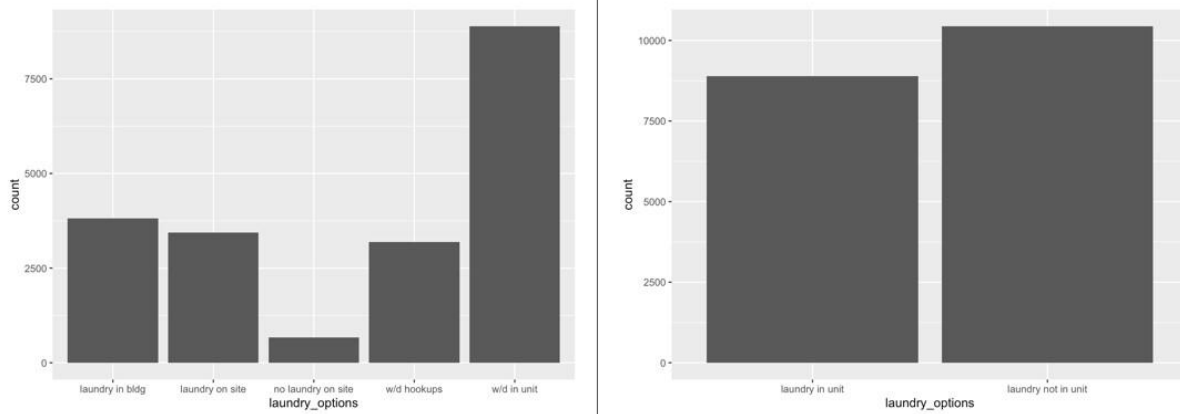
## 2. Dataset

The dataset used in this study is an original dataset on Kaggle that was collected from rental housing listings on Craigslist in 2020. As a popular website that caters to communities across the United States, the data from Craigslist can be useful in understanding the rental housing market nationwide. Additionally, the data is collected from early 2020, which avoids the impact of the Covid-19 pandemic.

Since the raw data contains 18 variables with over 384,000 observations, data wrangling and down-sizing is necessary. Outliers and missing values are dropped to avoid misleading information, and duplicates are also removed to ensure randomization during the data preparation process. The focus of the housing type has been limited to only "apartments".

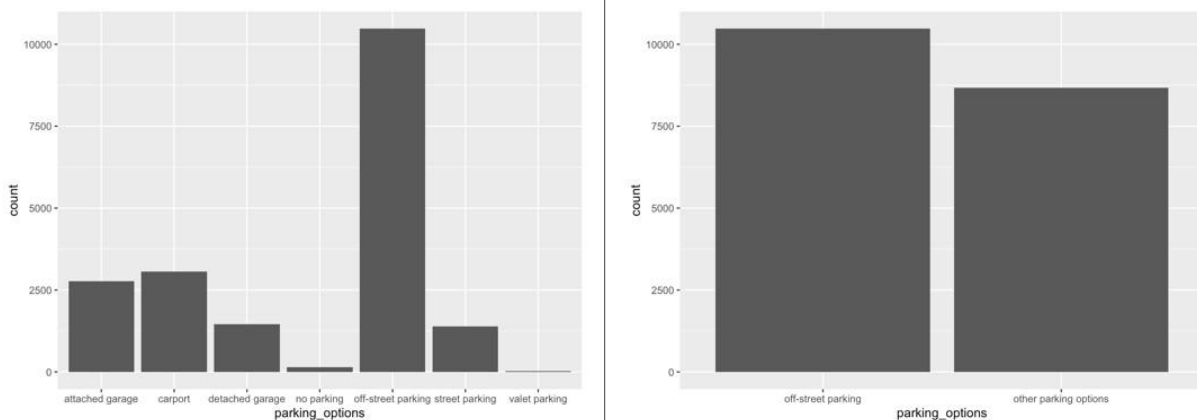
The cleaned new dataset is random sampled and down-sized to 20,000 observations for the representativeness of the data. Data splitting will be done to ensure the validity of the data. Descriptive statistics on the dataset can be found in previous reports as well.

In order to better utilize predictor variables in tree models, two variables (*laundry\_options* and *parking\_options*) were recategorized in order to avoid imbalance problems.



*Figure 2.1 Laundry Options before/after recategorizing*

Based on Figure 2.1 above, there are 5 categories under *laundry\_options*. “w/d in unit” (where there is a washer and dryer in the apartment unit) has the highest number of data, which will cause imbalance problems for the tree model and make it difficult for the model to generate nodes. In order to avoid potential problems, a combined version of *laundry\_options* has been generated. The previously “w/d in unit” option remains intact, and is now “laundry in unit”. All other subsets except “no laundry on site” have been combined into one category called “laundry not in unit”, for a total of two categories of laundry options available. The two subsets are much more balanced. The “no laundry on site” option contains a small amount of data (about 3% of the data), so there are no integrity issues in its omission. By excluding apartments with no laundry on site, laundry options are better categorized for analysis with tree models.



*Figure 2.2 Parking Options before/after recategorizing*

A similar procedure has been performed on *parking\_options*. The two subsets “valet parking” and “no parking” have been dropped due to lack of representation (less than 2% of total data). After dropping the two subsets, “off-street parking” becomes one subset and the rest combined becomes “other parking options”.

### 3. Classification Tree

#### 3.1 Target Variable Description

Since classification trees require a binary variable as the target, the variable *price* has been segmented into 2 categories: high price (1) and low price (0) based on the median of *price* to create a balanced dataset. By creating the new binary variable *price\_range*, the classification tree is able to classify new incoming data into different price ranges based on the conditions that the tree generates.

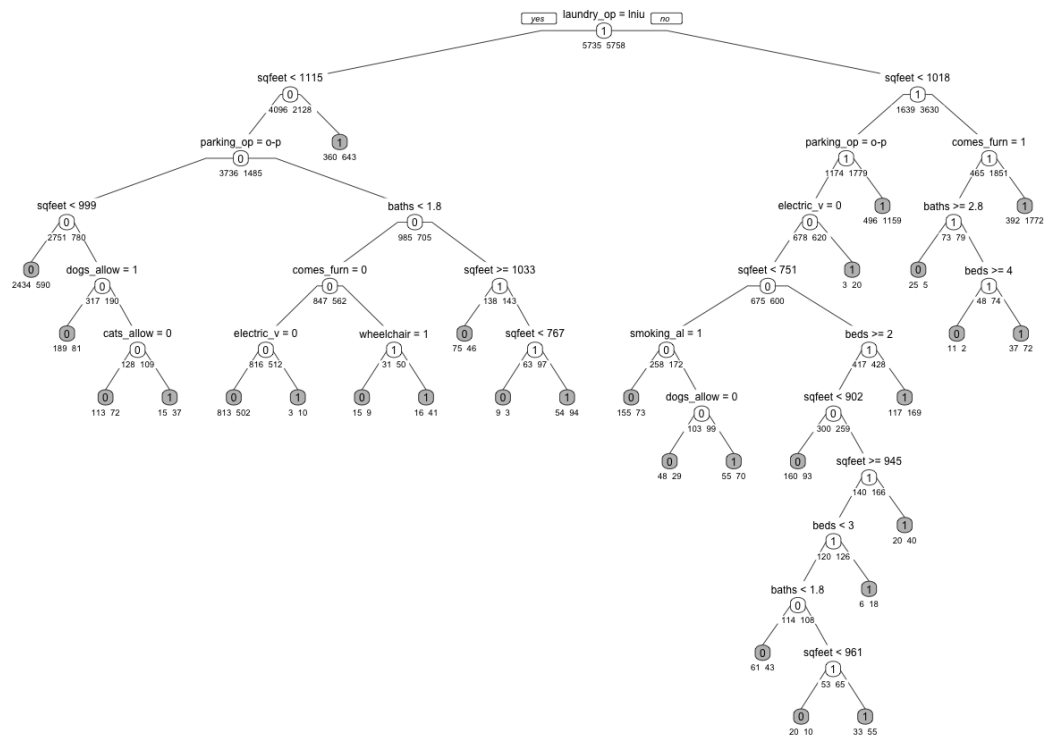
#### 3.2 Data Split and Cross Validation

Overfitting is a major problem for tree models, however it can be mitigated by performing data splitting and a cross validation analysis. The training data is 60 percent of the total data, and the validating data is 40 percent of the total. The data is randomly split into two groups, so the representativeness of the data is preserved. The model is trained with a cross validation method on the training set, then makes predictions with the validation set.

#### 3.3 Hyperparameters

In order to find the best hyperparameters, a grid search has been performed to help generate the best classification tree model. The complexity parameter (CP) has been set into a grid search loop that starts at 0.001 and ends at 0.1 with a 0.01 step increase on CP value each time through the loop. The minimum splits value has also been incorporated into the same grid search that starts from 1 and ends at 10 with a 1.0 step increase on the minimum splits value each time through the loop. The completed grid search will record the CP and minimum splits combination that results in the highest F1 value. Based on the grid search, the best hyperparameter combination found was: CP = 0.001 and Minimum Splits = 1, producing an F1 value of 0.7077.

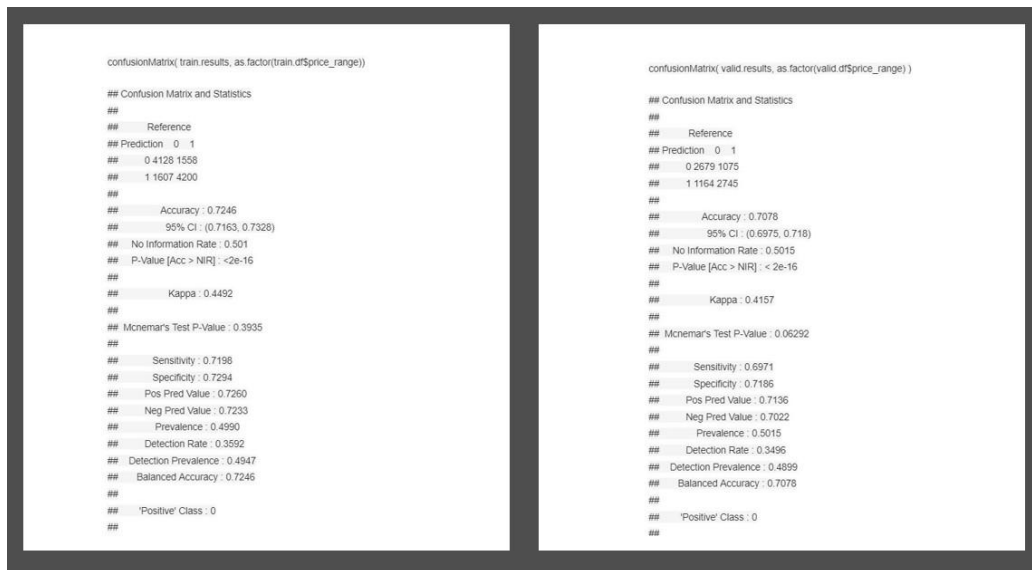
#### 3.4 Best Classification Tree Model



*Figure 3.1 Best Classification Tree Model*

Using the hyperparameters chosen from the grid search, a classification tree model was trained. In this classification tree, the root node starts with high priced apartments with laundry in-unit from. A total of 11 variables have been used to generate the classification tree, which are the same variables used to generate the classification regression model. There are 12 terminal nodes on the right side of the initial split at the root node, and 16 terminal nodes on the left, which proves the balance of the tree model. The terminal nodes are shaded gray and marked with 0 or 1 corresponding to low price range (0) or high price range (1). Each internal (non-terminal) node is labeled with a condition. For any node, the left branch indicates that the condition specified at the node is true, and the right branch indicates the node condition is false. All branches leading to a terminal node reflect the conditions that an observation must fulfill to achieve that classification. The tree model first splits at *laundry\_options*, and the first terminal node on the left shows that for a given apartment, if “laundry\_op = in unit” is true and “sqfeet < 1115” is not true, then class = 1 (high price range). On the right side of the root node, the first terminal node shows that if “laundry\_op = in unit” is false, “sqfeet < 1018” is true, and “parking\_op” is off street parking, then class = 1 (high price range). This method of interpretation applies for all terminal nodes.

### 3.5 Model Accuracy



*Figure 3.2 Confusion Matrix for Best Classification Tree Model*

Based on the confusion matrix, the prediction accuracy for the validation set is 0.7078, with a sensitivity of 0.6971 and specificity of 0.7186. The sensitivity measures how correctly a classifier can predict members in the data, which means the classification tree can correctly predict 69.71% of the class. The specificity measures how correctly a classifier can rule out the wrong members, which shows that the model can detect 71.86% of the false values. Overall accuracy of the model is 70.78%, which is reasonable, but there is still potential to improve in the future. The accuracy of the validation set is very similar to the training set. The sensitivity and specificity of the both training set and the validation set are also very similar. The classification tree model reasonably fits both the training set and the validation set, indicating the model does not have overfitting problems.

```

[1] "The F1 score of the test set"
[1] 0.7076861

```

*Figure 3.3 F1 Score for Best Classification Tree Model*

The F1 score of the best classification model is 0.7077.

### 3.5 Models Comparison

Train set	Test set
Confusion Matrix and Statistics	Confusion Matrix and Statistics
<pre> Reference Prediction  0  1 0  4614 1857 1  1448 3461 </pre>	<pre> Reference Prediction  0  1 0  3031 1278 1   947 2347 </pre>
<pre> Accuracy : 0.7096 95% CI : (0.7011, 0.7179) No Information Rate : 0.5327 P-Value [Acc &gt; NIR] : &lt; 0.000000000000000022  Kappa : 0.4139  McNemar's Test P-Value : 0.0000000000001275  Sensitivity : 0.7611 Specificity : 0.6508 Pos Pred Value : 0.7130 Neg Pred Value : 0.7050 Prevalence : 0.5327 Detection Rate : 0.4054 Detection Prevalence : 0.5686 Balanced Accuracy : 0.7060  'Positive' Class : 0 </pre>	<pre> Accuracy : 0.7074 95% CI : (0.697, 0.7176) No Information Rate : 0.5232 P-Value [Acc &gt; NIR] : &lt; 0.000000000000000022  Kappa : 0.4111  McNemar's Test P-Value : 0.0000000000002634  Sensitivity : 0.7619 Specificity : 0.6474 Pos Pred Value : 0.7034 Neg Pred Value : 0.7125 Prevalence : 0.5232 Detection Rate : 0.3987 Detection Prevalence : 0.5667 Balanced Accuracy : 0.7047  'Positive' Class : 0 </pre>

*Figure 3.4 Confusion Matrix for Logistic Regression with Cluster Variables*

The best classification model from Report 2b is logistic regression with cluster variables which produced a F1 score of 0.7. Compared to the classification tree's F1 score 0.707, the F1 scores for both models are very similar. Overall, the performance of the classification tree had slightly better results based on the confusion matrix, with accuracy 0.04% higher than the accuracy of logistic regression with cluster variables. The sensitivity of the classification tree is 6.48% lower than logistic regression, and specificity is 6.91% greater, which means the classification tree is better at excluding wrong members.

The classification tree uses the variables *beds*, *baths*, *cats\_allowed*, *dogs\_allowed*, *smoking\_allowed*, *wheelchair\_access*, *electric\_vehicle\_charge*, *comes\_furnished*, *laundry\_options*, *parking\_options*, and *sqfeet*. The logistic model used all the same variables and additionally included a polynomial term of *sqfeet*. Because of the similarities in variables used for each model, it is confirmed that these explanatory variables are effective in predicting the price range of an apartment. With the straightforward interpretability of the tree model, it can further be observed that if *sqfeet* is less than 1115, then it's likely that the price range will be low. If there are more than 2 beds and more than 3 baths then the price range will be more likely to be high.

## 4. Regression Tree

### 4.1 Target Variable Description

The target variable *price* has been used to perform the regression tree analysis. The numeric variable *price* is the most direct benchmark to measure the rental housing

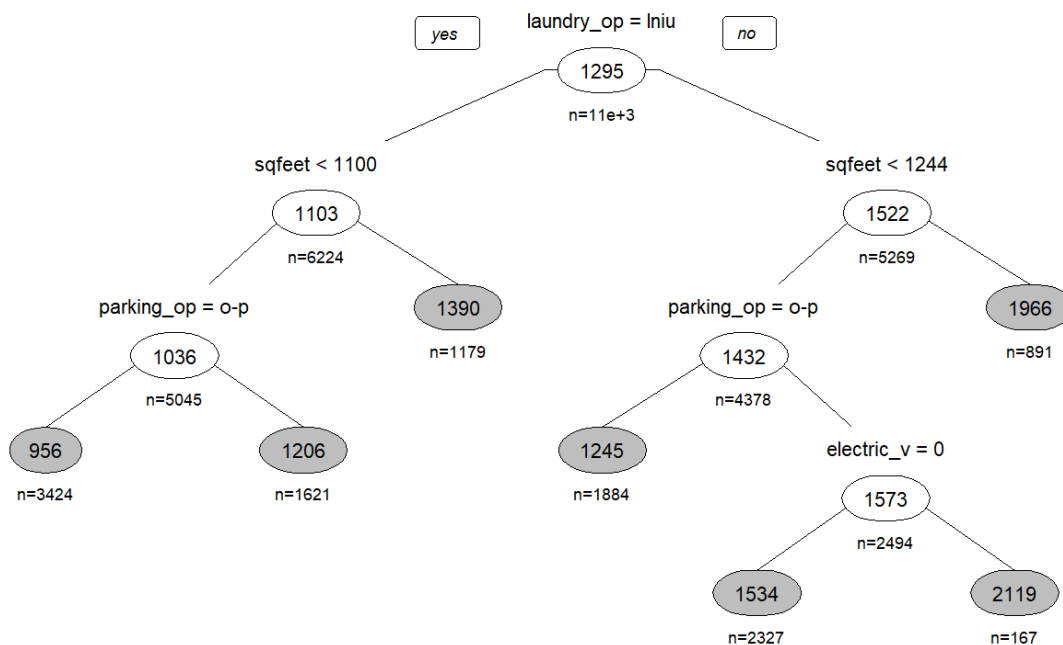
market and to observe how other variables impact apartment prices. To preserve the interpretability of the regression tree, the target variable *price* will not be normalized as it was in the previous reports, which may be seen as higher variance in the data.

## 4.2 Cross Validation

For the regression tree, the same cross validation that was performed on the classification tree has been used again in order to avoid overfitting. Again, 60% of the data has been divided into training sets and the remaining 40% of the data is the validation set.

## 4.3 Dimension Reduction

Since there are 12 variables in the dataset (11 predictor variables and 1 target variable), the regression tree will lose accuracy due to this high dimensionality. As variables are introduced, regression tree models can fail due to the fact that the model tries to use all the variables, which can result in models that are not useful. In order to reduce the dimensionality, an initial regression tree was created using all 12 variables and performed on the entire dataset to check which variables are useful in tree generation.



*Figure 4.1 Initial Regression Tree using all variables*  
Based on this initial regression tree, only 4 predictor variables are used to



generate the model, meaning the rest of the predictor variables that do not appear in the tree can be removed due to insignificance.

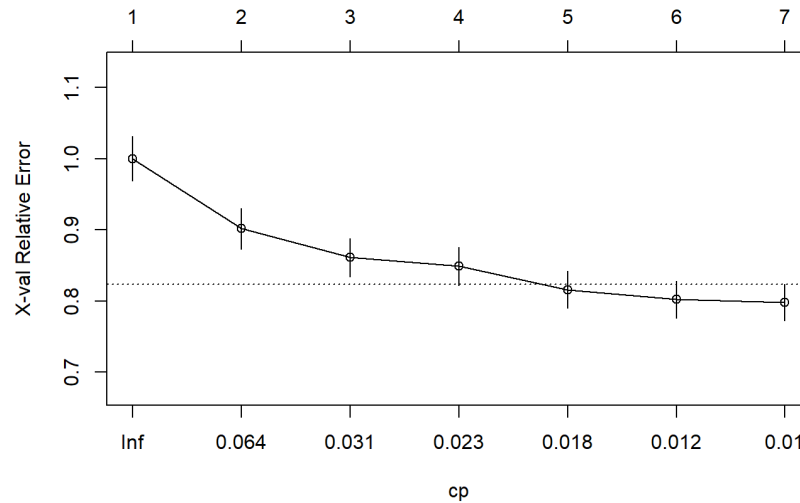


Figure 4.2 CP for Initial Regression Tree

	CP	nsplit	rel error	xerror	xstd
1	0.09873723	0	1.0000000	1.0001196	0.03114744
2	0.04155672	1	0.9012628	0.9016311	0.02884005
3	0.02351933	2	0.8597060	0.8613502	0.02691106
4	0.02280711	3	0.8361867	0.8487764	0.02669933
5	0.01350244	4	0.8133796	0.8157010	0.02595870
6	0.01048244	5	0.7998772	0.8020376	0.02570823
7	0.01000000	6	0.7893947	0.7977915	0.02567647

Figure 4.3 Xerror for Initial Regression Tree

Figure 4.2 presents the CP values corresponding to their cross validation error and size of the tree. Based on the results, at CP value 0.01 and tree size 7 with 6 splits, the cross validation error (xerror) is the smallest for the initial regression tree with a value of about 0.798.

#### 4.4 Hyperparameters

After the dimension reduction process, only 5 variables including the target variable *price* will be used in the optimized regression tree model. In order to further reduce the cross validation error, a grid search has been performed to find optimal hyperparameters.

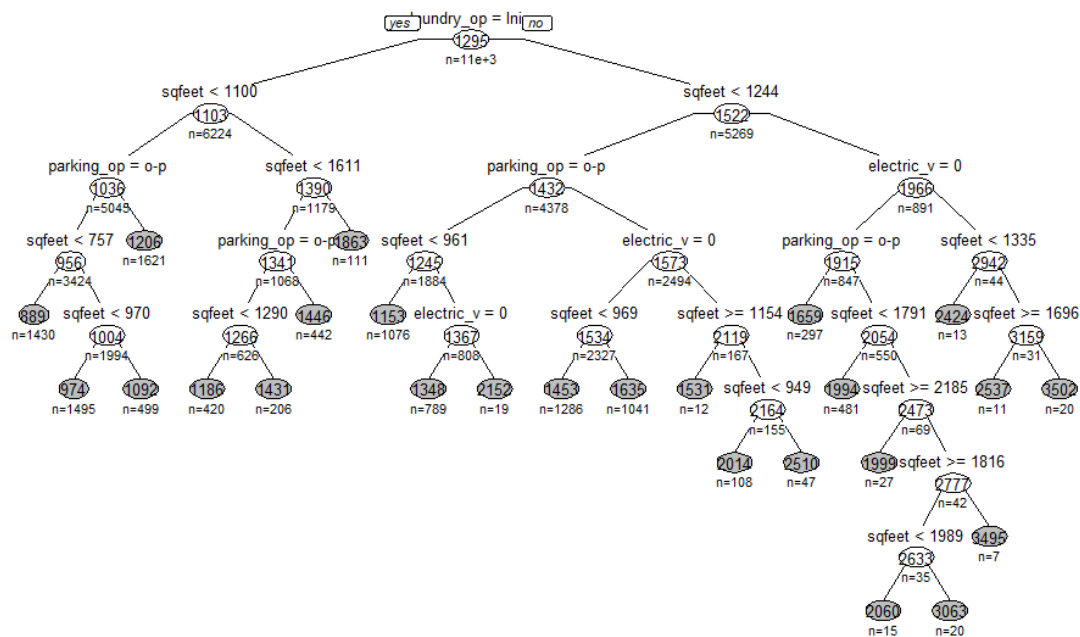
The complexity parameter is used to control the size of the decision tree and to select the optimal tree size. The minimum splits (minsplit) control the minimum number of nodes that a regression tree must have. The grid search for CP values started from 0.001 to 0.1 with a 0.01 step increase for each loop, and minimum splits values from 2 to 15 with a 1 step increase each loop were also included.

minsplit <dbl>	cps <dbl>	cp <dbl>	error <dbl>
6	0.001	0.001637289	0.7639229
13	0.001	0.001955848	0.7677097
12	0.001	0.001274840	0.7678800
4	0.001	0.001274840	0.7691030
9	0.001	0.002145174	0.7691755

*Figure 4.4 Grid Search For hyperparameters*

Based on the results from grid search, after removing insignificant variables and readjusting CP and minimum split values, the cross validation error was further reduced to 0.7639 with an optimized CP value of 0.001 and a minimum split value of 6.

#### 4.5 Best Regression Tree Model



*Figure 4.5 Best Regression Tree Model*

Using the best hyperparameters found in the grid search, a regression tree was generated on the training set. Based on the regression tree model, there are three major branches of the tree under the nodes “sqfeet < 1100”, “parking\_op = o-p”, and “electric\_v = 0”. Each branch has 8 terminal nodes except the branch under “electric\_v = 0” with 9 terminal nodes, which proves balance within the tree model. The tree starts

at the root node “laundry\_op = in unit” - if the apartment has laundry in the unit, then it will move to the second node “sqfeet < 1100”, and will be evaluated based on square footage at this node. If the apartment is smaller than 1100 square feet, then it will proceed to the third node “parking\_op = o-p”. At this node, if the apartment’s parking option is not off-street, then it will move to the terminal node 1206, which means that an apartment that has laundry in unit, *sqfeet* < 1100 and a parking option that is not off-street parking will have a predicted rental price of \$1206. Overall, large apartments with *sqfeet* > 1335 are more likely to have high rental prices.

#### 4.6 Model Accuracy

The final RMSE of the regression tree model is 568.4006 which suggests that, on average, the predicted rental housing prices are about \$568.4 off from the actual rental price. The adjusted R-squared for the regression tree model is 1.000653. This higher adjusted R-squared value is to be expected, as the data was not normalized.

#### 4.7 Model Comparison

The regression tree used 4 variables: *laundry\_options*, *sqfeet*, *parking\_options*, and *electric\_vehicle\_charge*. The regression model uses the same variables, with the addition of *wheelchair\_access*, *dogs\_allowed*, *comes\_furnished*, *beds*, *cats\_allowed*, *baths*, *smoking\_allowed*, and *state*. The difference in variables used can be attributed to the dimension reduction done earlier.

Compared to the best regression model from Report 2, which had an adjusted R-squared of 0.5773 and an RMSE of 0.2717, the regression tree model has both a higher adjusted R-squared and RMSE due to the difference in normalization of the data. This is largely due to an attempt to preserve the interpretability of the regression tree model. If the variables in the tree model were logged, the RMSE would decrease, though the readability of the regression tree would be low, negating the advantage of using a tree model. As it stands, the regression tree performs better than the regression model, though the error is much higher.

### 5. Ensemble Methods

Ensemble learning is a machine learning technique that aims to combine multiple models, also known as “weak learners”, to get better results. Assumptions are made that the right combination of weak learners will improve prediction accuracy. Therefore, an ensemble tree method is to combine several decision trees to produce better predictive performance than using any single decision tree method. Three algorithms are used to perform ensemble decision trees: Bagging, Random Forest, and

Boosting. The ensemble methods seek to predict the target binary variable *price\_range*.

## 5.1 Bagged Trees

Bagging is a technique used to reduce the variance of a decision tree. The idea is to create several subsets from randomly chosen training data with replacement, and each subset will train their decision tree, resulting in an ensemble of different models.

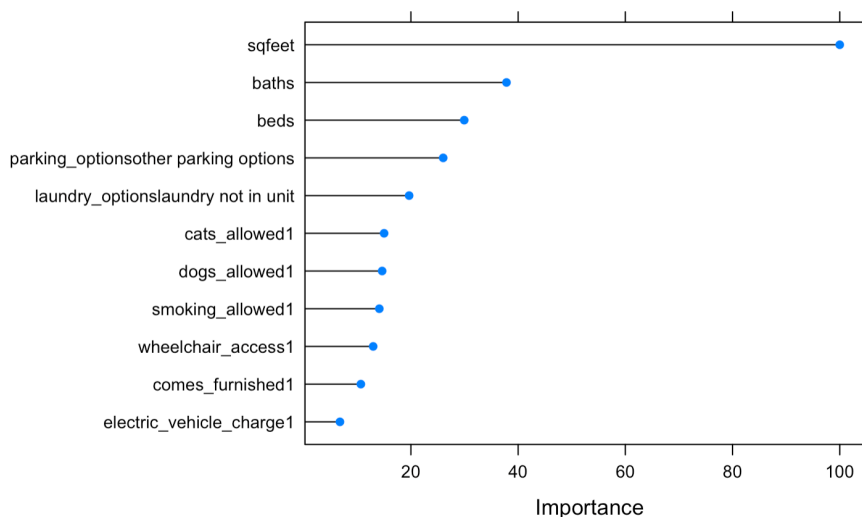
### 5.1.1 Hyperparameters

```
"The F1 score of the bagging tree before prune:"
0.7067032
"The F1 score of the bagging tree after prune:"
0.6796205
```

*Figure 5.1 The Comparison of F1 Score between bagged model before prune and bagged model after prune*

The pruning method is utilized to find the best bagged model. However, the F1 score of the after-prune bagged model is lower than the F1 score of the before-prune bagged model. Therefore, the best bagged model is the before-prune model.

### 5.1.2 Interpretation



*Figure 5.2 The Importance of Variables in Bagged Tree*

The plot of variable importance calculates variable importance for the bagged model produced by train set. The bagged model used all 11 predictor variables in the dataset. From figure 5.2, the variable *sqfeet* has the greatest importance on the bagged model, reaching approximately 100. The variables *baths*, *beds*, *laundry\_options*, and *parking\_options* are next in importance in the bagged model with a range of 20 to 40.

### 5.1.3 Model Accuracy and Comparison

```

##      Reference
## Prediction  0   1
##      0 2738 1142
##      1 1105 2678
##
##      Accuracy : 0.7068
##      95% CI : (0.6964, 0.717)
##      No Information Rate : 0.5015
##      P-Value [Acc > NIR] : <2e-16
##
##      Kappa : 0.4135
##
##      McNemar's Test P-Value : 0.4476
##
##      Sensitivity : 0.7125
##      Specificity : 0.7010
##      Pos Pred Value : 0.7057
##      Neg Pred Value : 0.7079
##      Prevalence : 0.5015
##      Detection Rate : 0.3573
##      Detection Prevalence : 0.5063
##      Balanced Accuracy : 0.7068
##
##      'Positive' Class : 0
##

```

*Figure 5.3 The Confusion Matrix for Bagged Tree*

The bagged trees method produced a model with 70.68% accuracy, which is very similar to the classification tree with 70.78% accuracy in the validation set. The sensitivity of the bagged model is 0.7125, which is slightly higher than the classification tree. The specificity of the bagged model is 0.7010, which is slightly lower than the classification tree. The F1 score of the bagged model is 0.7067, which is approximately 0.001 lower than the classification tree. Based on the comparison of the best bagged tree and the classification tree, their performances are very similar.

## 5.2 Random Forest

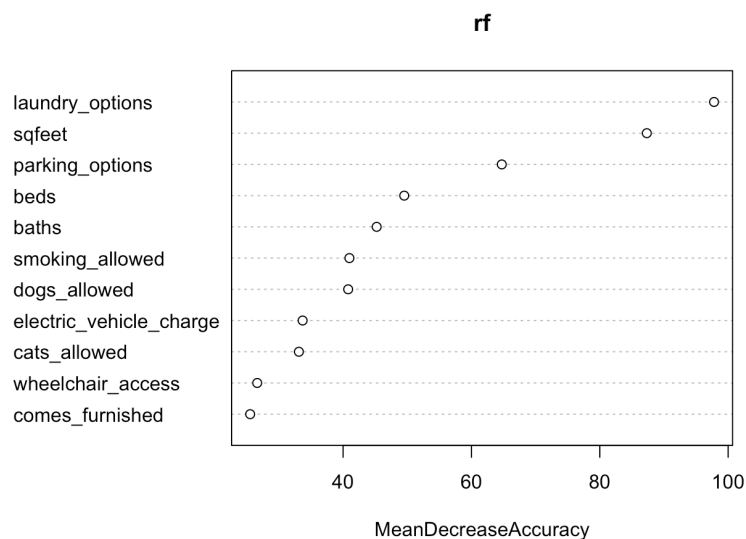
Random Forest is an extension of bagging, as it randomly selects subsets from training data and randomly selects features to grow trees. The results will contain many random decision trees.

### 5.2.1 Hyperparameters

A grid search is performed to find the best performing model in Random Forest. Two parameters are chosen to find the hyperparameters: the number of trees to grow (ntree) and the number of variables randomly sampled at each split (mtry). The minimum number of trees chosen is 100, and the maximum is 500, because the number of trees should be large so that the randomly selected samples are representative. The minimum number of variables at each split is 1, and the maximum is 10. The grid search will choose the best parameters based on the highest F1 value. With grid search, the hyperparameters are chosen: ntree = 200 and mtry = 5 when F1 is approximately

0.7298, which improves from the original classification tree model, 0.7077, by approximately 3.12%.

### 5.2.2 Interpretation



*Figure 5.4 Random Forest Important Variables*

Based on the best random forest model with the chosen hyperparameters, the variable importance plot tells how much the variable is being used by the model to produce accurate predictions, or how important the variable is to the model. This plot measures the Mean Decrease Accuracy, which is measured by permuting the values of the predictors in the out-of-bag samples and then calculating the corresponding accuracy decrease. The variables *laundry\_options*, *sqfeet*, and *parking\_options* are the most important variables, as removing any of these will lose over 60% of the model accuracy, and the variables *beds*, *baths*, *smoking\_allowed*, and *dogs\_allowed* are important (40-60% decrease each), whereas variables *electric\_vehicle\_charge*, *cats\_allowed*, *wheelchair\_access*, and *comes\_furnished* are not as important to the model (under 40%).

### 5.2.3 Model Accuracy and Comparison

Confusion Matrix and Statistics				Confusion Matrix and Statistics			
		Reference				Reference	
Prediction	0	1		Prediction	0	1	
0	2869	1116		0	2679	1075	
1	974	2704		1	1164	2745	
Accuracy : 0.7273				Accuracy : 0.7078			
95% CI : (0.7171, 0.7372)				95% CI : (0.6975, 0.718)			
No Information Rate : 0.5015				No Information Rate : 0.5015			
P-Value [Acc > NIR] : < 2.2e-16				P-Value [Acc > NIR] : < 2e-16			
Kappa : 0.4545				Kappa : 0.4157			
McNemar's Test P-Value : 0.002041				McNemar's Test P-Value : 0.06292			
Sensitivity : 0.7466				Sensitivity : 0.6971			
Specificity : 0.7079				Specificity : 0.7186			
Pos Pred Value : 0.7199				Pos Pred Value : 0.7136			
Neg Pred Value : 0.7352				Neg Pred Value : 0.7022			
Prevalence : 0.5015				Prevalence : 0.5015			
Detection Rate : 0.3744				Detection Rate : 0.3496			
Detection Prevalence : 0.5200				Detection Prevalence : 0.4899			
Balanced Accuracy : 0.7272				Balanced Accuracy : 0.7078			
'Positive' Class : 0				'Positive' Class : 0			

*Figure 5.5 Confusion Matrix for Random Forest and Classification Tree*

Figure 5.5 shows the confusion matrix for the best model using both random forest and classification tree methods. On the right side, the best random forest model shows a 72% accuracy, and, on the left side, the best classification tree model shows a 70% accuracy on the validation set, which improves the model by 2.75%. The sensitivity of the random forest model is 0.7466, which is 7.1% greater than the classification tree model, meaning it is more likely to detect false negatives and is better at predicting true positives correctly. However, the specificity of the random forest model is 0.7079, 1.48% smaller than the classification tree model, meaning that the random forest model will result in more false positive values, and the classification tree model is slightly better at predicting the true negative values.

### 5.3 Boosted Trees

Boosting is a method to create a collection of predictors, with the learners learning sequentially from previous learners that fit the data. This, in turn, reduces the errors.

#### 5.3.1 Hyperparameters

A grid search is performed to find the best performing boosted tree model, and the parameter used is the number of trees to be included in the model (mfinal). The minimum number of mfinal is set to be 1, as the number cannot be set to 0, and the maximum number of mfinal is set to be 100, because the default value is 100. The method is aimed to find better performing parameters not exceeding the default base model. Using the grid search, the hyperparameter is chosen: mfinal = 31, with the



highest F1 value, 0.7111, which improves from the classification tree model by 0.48%.

### 5.3.2 Interpretation

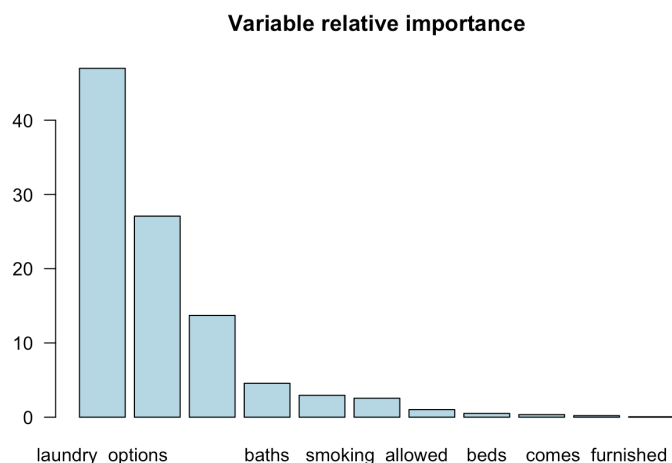


Figure 5.6 Boosted Tree Important Variables

Based on the best boosted tree, the most important variables are *laundry\_options*, *sqfeet*, and *parking\_options*, and the least important variables are *beds*, *comes\_furnished*, *wheelchair\_access*, and *cats\_allowed*. Compared with the important variables selected by the random forest model, the most important variables and least important variables are similar, except for the variable *beds*. In the boosting method, the variable *beds* is not important in model prediction.

### 5.3.3 Model Accuracy and Comparison

Confusion Matrix and Statistics	Confusion Matrix and Statistics
Reference Prediction 0 1 0 2795 1167 1 1048 2653  Accuracy : 0.7109 95% CI : (0.7007, 0.7211) No Information Rate : 0.5015 P-Value [Acc > NIR] : < 2e-16  Kappa : 0.4218  McNemar's Test P-Value : 0.01217  Sensitivity : 0.7273 Specificity : 0.6945 Pos Pred Value : 0.7055 Neg Pred Value : 0.7168 Prevalence : 0.5015 Detection Rate : 0.3647 Detection Prevalence : 0.5170 Balanced Accuracy : 0.7109  'Positive' Class : 0	Reference Prediction 0 1 0 2679 1075 1 1164 2745  Accuracy : 0.7078 95% CI : (0.6975, 0.718) No Information Rate : 0.5015 P-Value [Acc > NIR] : < 2e-16  Kappa : 0.4157  McNemar's Test P-Value : 0.06292  Sensitivity : 0.6971 Specificity : 0.7186 Pos Pred Value : 0.7136 Neg Pred Value : 0.7022 Prevalence : 0.5015 Detection Rate : 0.3496 Detection Prevalence : 0.4899 Balanced Accuracy : 0.7078  'Positive' Class : 0

Figure 5.7 Confusion Matrix for Boosted Tree and Classification Tree

Figure 5.7 compares the confusion matrix between the boosted tree and



classification tree models. The accuracy of the boosted model increases from 70.78% to 71.09%, by approximately 0.438%. The sensitivity increases from 0.6971 to 0.7273, meaning that the boosting tree is better at predicting true positives correctly, whereas the specificity is 0.6945, which is 3.35% less than the classification tree model, meaning that the classification tree is better at predicting true negatives than the boosted tree model. Overall, the accuracy increases after adapting the boosting method to the model.

## 6. Summary

### 6.1 Comparison of Different Ensemble Models

After choosing the hyperparameters via grid search, random forest has the highest F1 value, 0.7298, among all of the ensemble models. This is reasonable because random forest takes all subsets into consideration by randomly choosing subsets. Random forest also has the highest sensitivity value, 0.7466, so it is better at predicting true positives correctly. Moreover, the random forest model also has the highest specificity value, 0.7079, which means that it is also better at predicting true negatives correctly, and it makes random forest the best model out of all of the ensemble models.

### 6.2 Comparison of Non-Ensemble Models to Ensemble Models

The prediction accuracy for the classification tree model is 70.78%, and after using an ensemble model (random forest), the accuracy increases by 3.12%, from 70.78% to 72.98%. With the use of the boosted tree method, the accuracy increases by 0.438%, from 70.78% to 71.09%. For the bagged tree method, the accuracy decreases by 0.12%.

The accuracy for the logistic regression model is 0.7074, as mentioned in section 3.5. After using the ensemble method, the random forest model has increased the accuracy by 3.16%, and the boosted tree model has increased the accuracy by 0.49%. However, the bagged tree model has decreased the accuracy by 0.084%.

Overall, ensemble methods help increase the prediction accuracy over non-ensemble methods. The random forest and boosted tree models perform better in terms of accuracy. The bagged tree model shows a slightly lower accuracy than the classification tree model and the logistic regression model. One potential reason why the accuracy doesn't improve could be because the bagged tree model helps reduce variance error, whereas the base model contains bias errors instead of variance errors, so the random forest model and boosted tree model would perform better, as they are designed to reduce bias errors.

### 6.3 Reflections

The process of modeling with trees required slightly different considerations than the previous classification and regression models. There were differences in how we handled the data because of the strengths in the methods. In the previous reports more detail in data wrangling was required - for example, it was important to perform normalization of the data. In tree models, normalizing the data does not matter as much, and we actually found it to impede interpretability in our regression tree. With normalized data, the tree gave very small values that we could not make sense of in the context of apartment price.


While the accuracy for tree models was higher than the previous models, interpreting the models was not always intuitive because of the many branches that a more complex tree could have. However, it is a bit simpler to see larger patterns and to distinguish more important variables in a tree.

In building the tree models, we were aware of what variables would be included in the tree models. Though multicollinearity does not have the same effects on the predicted results in a decision tree that would occur in a linear or logistic regression, we removed collinear variables such as *price* and *price\_range* before generating the trees. We also omitted the cluster variables from Report 2b after considering that it may be difficult to separate out the characteristics of a specific cluster in a real-world context.

Another consideration during the process of using tree methods is that trees are sensitive to data. If certain data are omitted or added, it could change the trees significantly. We understand that in combining laundry options as well as parking options, this produced a different tree than if the dataset was left alone, however, we felt that having balanced data was important for model accuracy.

For trees, we found that overfitting was not as obvious as in other methods where training and testing accuracy can directly be compared. It is possible that a tree can be very closely fitted to the training data but also perform well on test data. To keep an eye on potential overfitting, we made sure to avoid models that got too complex, and used grid searches to find the best parameters for our trees. Pruning trees can be done to reduce overfitting as well. In our bagged trees model, we saw that pruning reduced the accuracy and elected to preserve the better performance, though this may have indicated the presence of some overfitting.

Additionally, we found that using grid searches greatly improved our process of finding the best hyperparameters for different models. After we implemented a grid search program into our code, we could easily check combinations that would be difficult to try one at a time. The grid search also returned values in a ranked table that gave us a better understanding how the combinations of hyperparameters would affect the model. However, the process of grid search can take a long time to perform, and this



problem was especially apparent when we tried to deploy ensemble methods. The long processing time was made longer due to the many subsets that the ensemble models engaged.

Because all of our models were close in accuracy, it is difficult to tell which one is best out of them all. The most accurate model in this report was created using the random forest method, however, there was some interpretability lost because of the black box nature of the method. It will be interesting to see how all our models so far compare with one created by artificial intelligence.