Final Project

Ian Char
Ji Hoon Kim
APPM 5590

February 26, 2018

1 Introduction

While a constant rise in homeownership pervaded American society in the 1900s, the recent years have shown a reversal as people have been turning towards the rental market for their housing needs. In 2014, 36% of Americans resided in rental properties - the largest since 1994 [1]. This upward trend in demand in recent years has garnered the interest of those investing in the rental market such as the investment company, Two Sigma. As a result, gauging rental property interest in the market has become not only an important problem, but a valuable and economical one as well. This brings us to the problem presented by Two Sigma: given a set of rental listings, can one estimate the expected interest for the property by classifying them into low, medium or high interest levels?

To tackle this problem, we first describe the methods that we used to analyze the data, process the data, and perform feature engineering in order to make it more accessible to the models. The two fundamental models that are used in this paper are Random Forest and Ordinal Logistic Regression. As a further consideration, the dataset derives from actual rental listings in New York City without which any treatment or control is prescribed; thus, our study is purely experimental.

Ultimately, we are interested in solving the Two Sigma problem as to whether we can correctly classify the rental listings into the correct categories. Furthermore, we hope to answer which model performed best, which variables seem to be the most important in our analysis, and whether we can perform feature engineering in order to improve our predictions. To see the code used for this paper visit `https://github.com/IanChar/StatsModelling`.

2 Background and Overview of Data

2.1 Description

The data that will be used for this was provided by RentHop for a previous Kaggle competition. RentHop is a website that allows users to search for apartments in specific areas. The training dataset provided is composed of 49,352 different apartments, all exclusively located in New York City. Each measurement in the dataset is comprised of the interest level of RentHop users for the given apartment (either low, medium, or high), the price of the apartment (US dollars per month), the number of bedrooms, the number of bathrooms, the latitude and longitude, links to the photographs of the apartments, a description of the apartment, a list of features that the apartment has, and the date that the apartment listing was posted.

Listing 1: Summary of Dataset

```
   bathrooms          bedrooms          price          num_photos
1       :39379   1       :15734   Min.    :     43   Min.    : 0.000
2       : 7649   2       :14602   1st Qu.:   2500   1st Qu.: 4.000
3       :  743   0       : 9462   Median :   3150   Median : 5.000
1.5     :  645   3       : 7270   Mean    :   3830   Mean    : 5.604
0       :  312   4       : 1929   3rd Qu.:   4100   3rd Qu.: 7.000
2.5     :  277   5       :  245   Max.    :4490000   Max.    :68.000
(Other):  285   (Other):   48
 num_features      desc_length     interest_level
Min.    : 0.000   Min.    :   0   0:34234
1st Qu.: 2.000   1st Qu.: 340   1:11220
Median : 5.000   Median : 564   2: 3836
Mean    : 5.429   Mean    : 602
3rd Qu.: 8.000   3rd Qu.: 809
Max.    :39.000   Max.    :4466
```

As one can see, this dataset is rich in the type of information it supplies. Not only is there numerical and ordinal data, but there are also pictures and text as well. In order to make this information easier to

analyze, we will consider only the number of pictures provided, the number of features listed, and the length of the description for each apartment. A summary of the data (without location and date) can be seen in Listing 1. For interest level, 0 represents low interest, 1 represents medium interest, and 2 represents high interest.

2.2 Cleaning the Data

There are several abnormalities with the dataset that will first need to be addressed before any further analysis. For one thing, it seems that there are some apartments that are clearly not located in New York city. These 62 data points were removed from the dataset since we wish to develop a model for apartments in New York exclusively.

Furthermore, there appear to be several outliers in the dataset when it comes to pricing. For instance, there are two apartments listed for $43 and $45 per month respectively. This is an outrageous price for an apartment in New York City and is almost certainly either a fake listing or is due to an error in entering data. As such, these two points are removed from the dataset. Additionally, there are four apartments that are over one million dollars a month. Not only this, but the next most expensive apartment is only $135,000 a month. However, we do not remove these points since they give valid information about the problem.

2.3 Exploration of Data

We now do some initial exploration of the data in order to get some intuition for what features should be important in a model. A natural first thing to look at is the relationship between the price of the apartment and the interest level. From Figure 1, one can see that the apartments with medium interest have consistently lower prices than the ones with low interest. Likewise, the apartments with high interest have consistently lower prices than both the apartments with lower and medium interest. Not only does the mean in price increase as interest rises, but the amount of spread in the data also decreases. Thus, for higher interest levels we can more confidently predict that the price of the apartment will be lower.

Figure 1: Box plots of price plotted by interest level. Note that outlier points extend outside of bounds of plot, especially for an interest level of 0.

However, we also have reason to believe the relationship between price and interest level is affected by the number of bedrooms in an apartment. For example, an apartment for $3,000 might be a bad deal if there is only one bedroom but a good deal if there are three bedrooms because one could expect that price to be split between three different people. In the four plots in Figure 2, it seems like the previous trend holds for the most part. Another interesting thing to notice is that there seems to be a wider range of prices as the number of bedrooms in the apartment increases.

We see a spatial plot of interest levels across New York City in Figure 3. Interestingly enough, although we can say somewhat that the interest in apartments on Manhattan are low and the interest in apartments

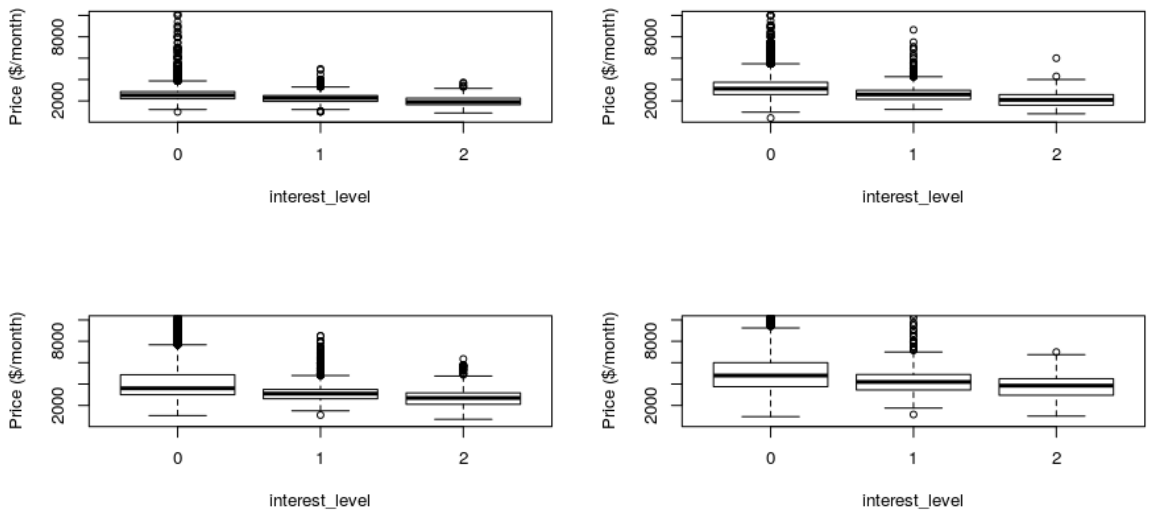

Figure 2: Box plots of price plotted by interest level. The plots show apartments with no bedrooms (upper left), one bedroom (upper right), two bedrooms (lower left), and three bedrooms (lower right). Again, outlier points may extend outside of the range of the plot.

across Brooklyn and the Queens seems to be much higher, the interest level seems to still be rather randomly scattered. This may be because majority of the listings in Manhattan are very high in price and therefore, the average user on RentHop is most likely not looking to rent in the area. From this, we generate a notion of "neighborhoods", or areas of that share similar features, area and prices. This will be described in a later section

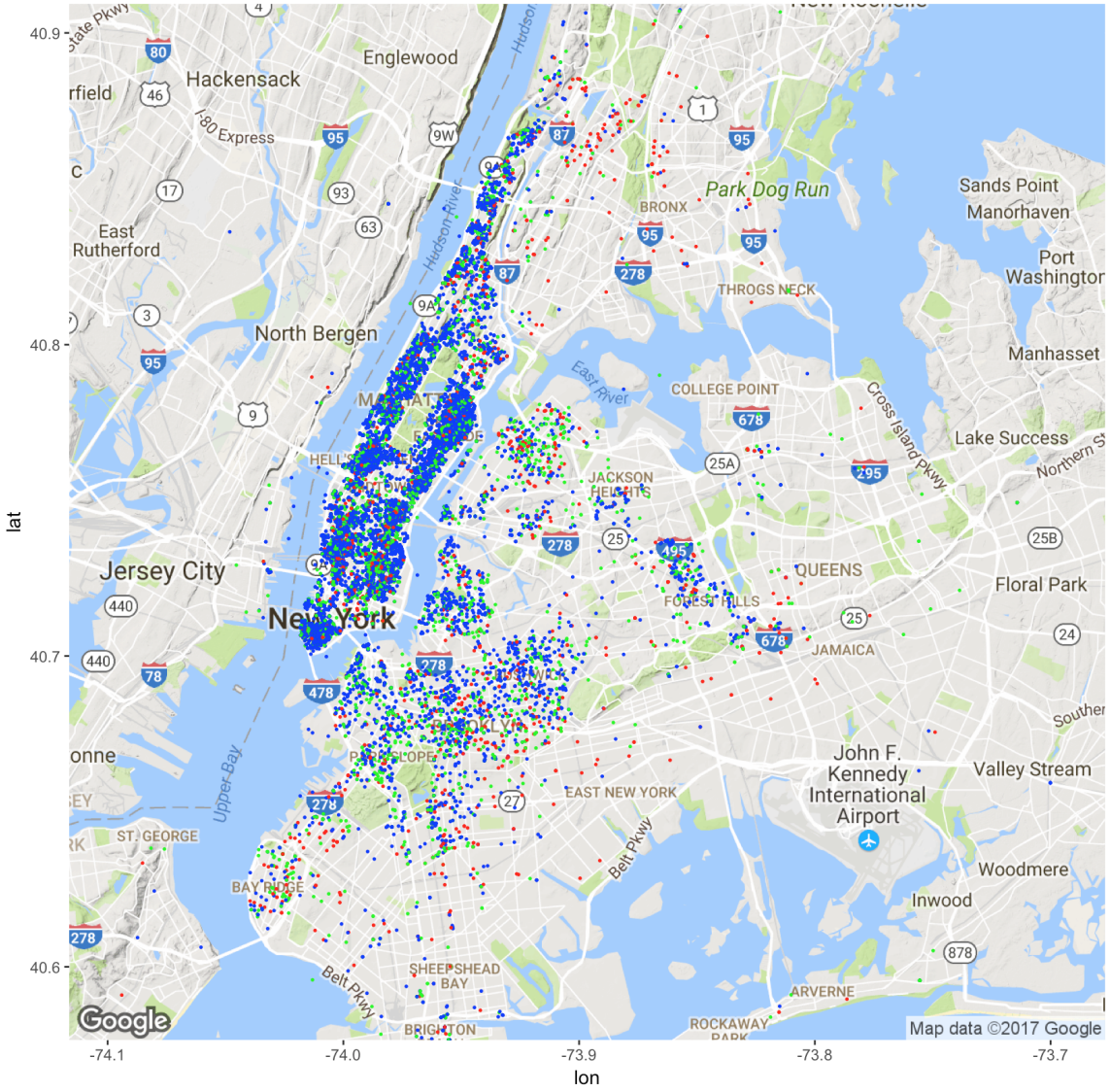

Figure 3: Spatial plot of interest levels. Blue, green and red colors correspond to low, medium and high interest levels respectively

Alternatively, one can observe the temporal aspects of the data in Figure 4. Note that while there is not much fluctuation when it comes to day and month, the hour of the day seems to contain some amount of information. Not only is there a massive spike at the beginning of the day, but the proportion of apartments observed in the later part of the day seems to increase as interest increases.

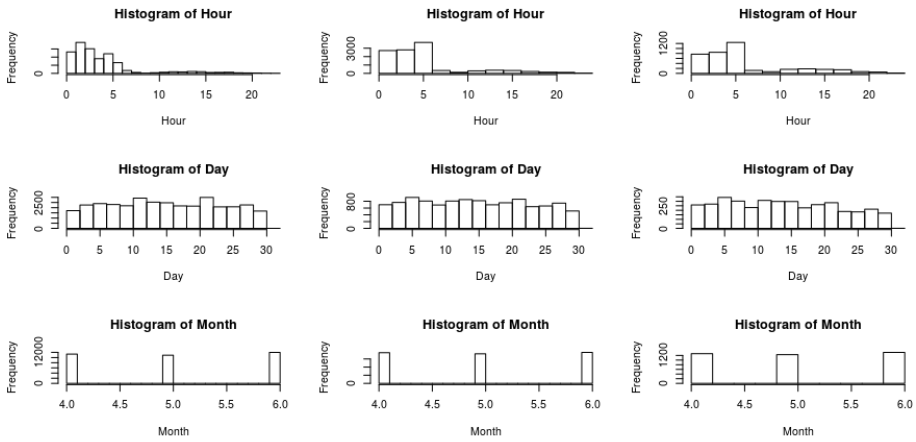

Figure 4: Histograms of frequency for hour, day, and month separated by interest level. Here the apartments represented in the leftmost plots have low interest, the middle plots pertain to medium interest, and the rightmost plots pertain to high interest.

Figures 5 and 6 are word clouds constructed from the descriptions and the features of apartments listings that were of "high" interest respectively.

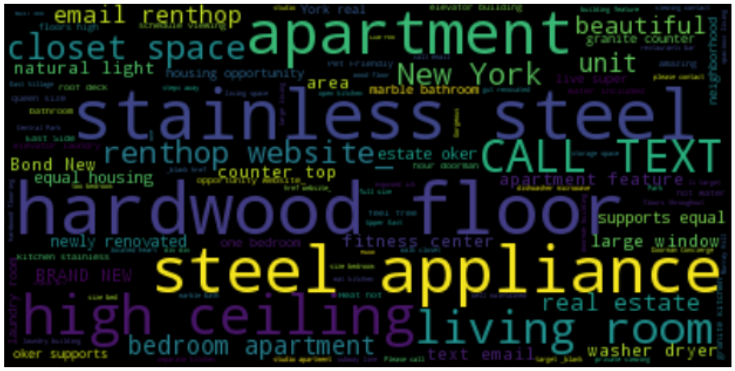

Figure 5: Word clouds of descriptions of apartment listings with high interest

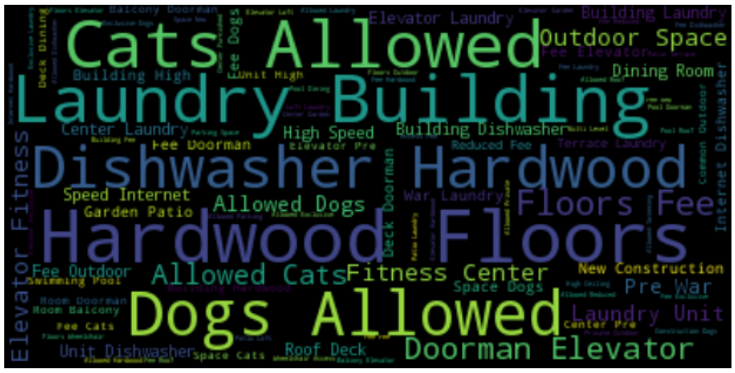

Figure 6: Word cloud of the features of apartment listings with high interest

Although these features are not included in the model, it is very interesting to see the descriptions and features that people found very important when selecting a rental listing. Note that there is also overlap between certain descriptions and features such as "Hardwood Floors" which could be an indicator to high interest if a rental listing has those features. Although our model only takes into account the number of features and the length of the description, further analysis could be done by incorporating these top features noted in the word cloud into the model.

3 Random Forest

After our exploration of the data, we begin delving into actually using a statistical model in order to glean information and then successfully classify apartments into their respective categories.

3.1 Motivation and Overview

Before we begin the analysis of the data, we must first understand what a random forest even is. And to understand random forest, we must begin with an overview of decision trees.

A decision tree is a flowchart-like tree where the root node represents the whole population and each node represents a test on a predictor variable. Then the population is filtered left or right depending on the outcome of the node on a predictor. As an example, if there is a predictor p, the test on the node is $p >= 30$ and the left and right branch outcomes are true and false respectively; if we have a data point with $p = 50$, then we move the data point down the left branch. This process is repeated for each data point until a leaf, or terminating node, is reached. Each leaf possesses a decision or classification; at this point, the data point takes on this classification. In our model, the classifications are "0" (low-interest), "1" (medium-interest), and "2"(high-interest).

As an example, we look at one decision-tree for our dataset in Figure 7. Consider the first data point. It has one bed and bath, $2400 a month, 40.7108 N, 73.9539 W, 12 photos, 7 features, 553 character description, and posted on May 16th at 5 AM. It is seen to have "medium" interest. Let's see how this data point goes through the tree.

The point filters left if true and right if false. The first node splits on price, anything greater than $2524 a month goes left, and if not, goes right. We move right towards the next node since the price for the given apartment is $2400 a month. Continuing that path through the tree, we move sequentially right, left, right, right, left, and then finally right simply by following the rules of the tree. Through this, we land on the 6th leaf from the left. According to this decision tree, we classify the point as "medium" interest as the leaf node has the value 1 or "medium." Interestingly, the actual class of the datapoint is "medium" as well meaning that we correctly classified this point through this tree.

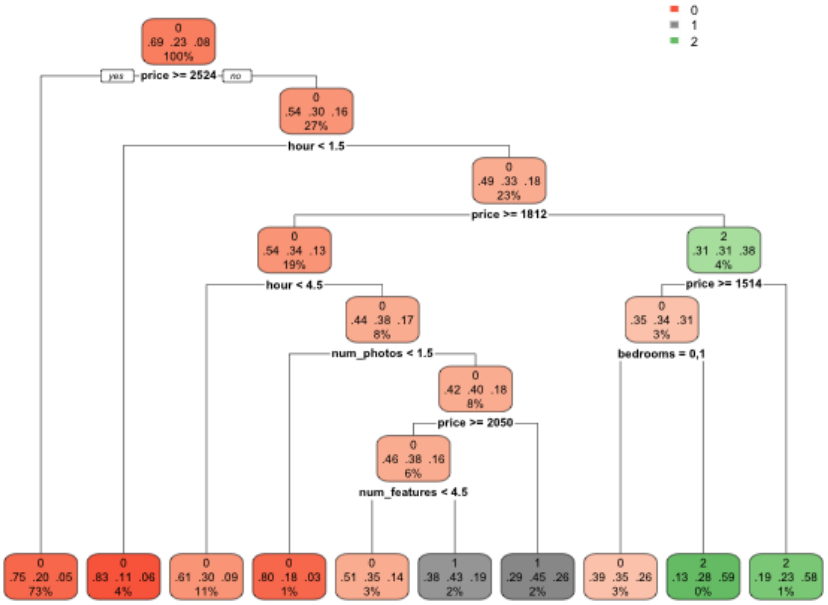

Figure 7: Sample decision tree created by the `rpart` package

While this answers the question of how decision trees work, we must discuss how the trees are created to make these decisions. There are many algorithms to make these decision trees; however, we will be focusing on the algorithm that was used to create the decision tree in R using the `rpart` package. The algorithm at a high level is simple:

1. Find the best feature/predictor of the dataset for predicting the classification of that object
2. Find a value/split on the predictor to split the dataset into two new subsets
3. Keep performing the previous two steps until all the predictors have been exhausted

We call this algorithm CART (Classification and Regression Trees).

The method we use to determine the best/most significant feature and the best value to split the two datasets is beyond the scope of the project. As a general overview, it uses variable selection by computing the entropy/information gain for each variable in the design matrix. It then sorts the variables by ascending entropy/information gain and then uses the variable with the most entropy. The intuition is to select the variable that gives you the most information first in order to make decisions quickly and accurately. Again, the literature for entropy/information gain is beyond the scope of the project and is available elsewhere. Then the value used to select the split is through maximizing the split between the classes (filtering the most data points into unique bins). Note that this approach to constructively build the tree is greedy, like forward selection. This means that each decision tree is not necessarily the optimal tree. Thus, we move onto random forests which help to tackle this problem.

A random forest is, as the name implies, a collection of decision trees to provide an ensemble for our data set. The number of trees is determined by the user. The random forest takes in a subset of the original

population through a sample. Then a decision tree is grown from this sample through the process described above. This process is repeated until the desired number of trees is reached. Then, each data point is filtered through all the trees and each classification through each tree is totaled. The majority vote is used to classify the data point. For example, if we perform random forest with 1000 trees and a data point is classified as "0" 610 times, "1" 212 times, and "2" 178 times, then the data point is given the classification of "0" since it was the classification "voted" on by the trees.

3.2 Advantages and Disadvantages

There are several advantages to the random forest classifier

1. Simple to understand and is a white box test
2. Works with both numerical and categorical data
3. Not a linear model and therefore has more flexibility
4. Can handle thousands of predictors with ease
5. Works with large amounts of data once the trees are grown

There are few disadvantages to using random forest; however there are a couple notable cases

1. Growing a tree optimally is NP-hard and therefore, requires a greedy algorithm approach to grow. This makes it nearly impossible to ensure a globally optimal decision tree as the number of predictors grows.
2. As a large number of predictors are put into the model, there is a tendency for the decision tree to overfit. Random forest mitigates this over-fitting using the democratic decision process but still suffers from it.
3. Growing the decision tree takes a significant amount of time if improperly tuned

3.3 Justification of Assumptions

There are a couple of assumptions that Random Forest makes:

1. Assumes that all of the splits done at each node is the best split possible
2. Sampling of the data is representative of the population
3. Observations are independent

We assume that the splits done at each node is the best since, although the algorithm is greedy, an exhaustive search for the tree is computationally impossible (NP-hard). Therefore, our assumption is only to compensate for this fact. We can also assume that sampling is representative since the sampling is done at random through a uniform distribution and no related population is selected in particular for sampling (whether it is to select the training set for our random forest or the procedure done by random forest to create the decision trees). Furthermore, we can assume that the observations are independent, i.e. they are not from a designed experiment nor a selective population nor from a matched case-control study. This is because this study is purely experimental and the data was randomly fetched from the RentHop website.

3.4 Clustering

In order to differentiate our random forest model from a naive random forest solution (simply throwing all of the covariates in our model), we cluster our dataset based off longitude and latitude in order to gain some notion of neighborhoods and add it as an additional variable.

For this model, we use the CLARA clustering algorithms. CLARA (CLustering LARge Applications) is a method of clustering large datasets that takes a sample from the dataset, computes the k-medoids (unlike

k-means) for this sample, and clusters all the data based off of these medoids. This method is similar to k-means clustering and the clustering algorithm PAM. What CLARA does, though, is compute the PAM algorithm (computing medoids) across many subsets and then applies the observation of the dataset to the nearest medoid, thus clustering en masse.

We cluster on the variables longitude and latitude as seen in Figure 8 into 10 different clusters across the region of interest.

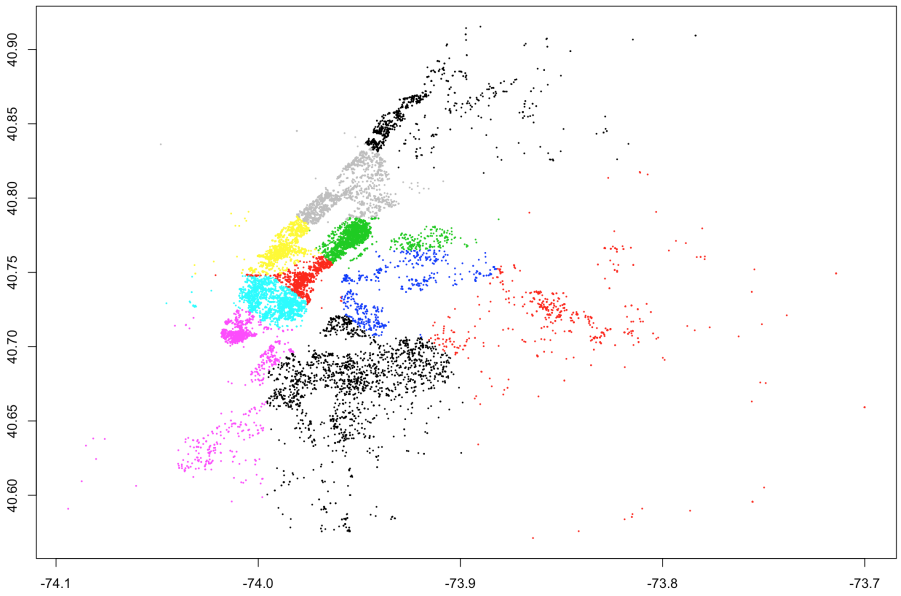

Figure 8: Clustering the rentals based on longitude and latitude

If we cluster on all of the different variables, we obtain the plot in Figure 9

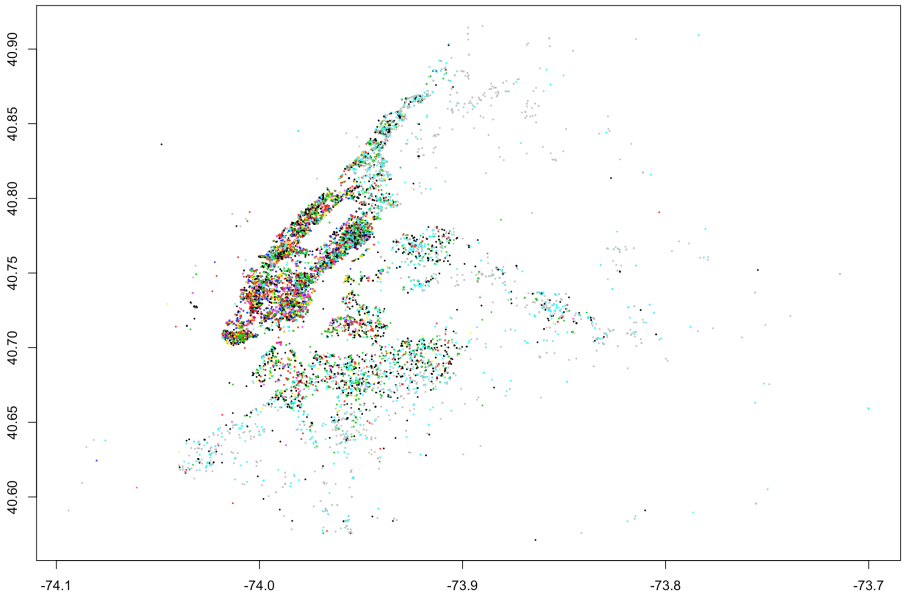

Figure 9: Clustering the rentals based on all of the other variables

We will be analyzing the two different clusters in the Results section later on. For now, we add this clustering value as a new variable into our model.

3.5 Selection of Variables

The wealth of variables in our model comes as a double-edged sword. We could potentially glean more information from our data from an increase in the number of variables; however, this also means that we could overfit the model, especially since random forest is prone to this. Therefore, we must implement some notion of variable selection. The variables we have available, as a reminder, are bedrooms, bathrooms, price, latitude, longitude, number of photos, number of features, length of description, month, day, and hour.

Interestingly, random forest has a method of measure variable importance through the use of Gini importance. This is done by computing the mean Gini gain produced by the j-th variable. This is seen by the `importance` parameter in random forest. We will not be covering the equation used to compute this Gini importance in this paper, but much literature exists on how to compute this. How R computes the variable importance is by growing a certain number of trees and then using this mean decrease Gini value. Another method that R uses is through the mean decrease in accuracy. Simply by the same process, the mean decrease in accuracy by the j-th variable is computed. For this paper, we will be using the mean decrease in accuracy variable importance. We compute the variable importance plot as seen in Figure 10 by running the random forest algorithm on the dataset once and by including all of the predictors excluding the clustering variable.

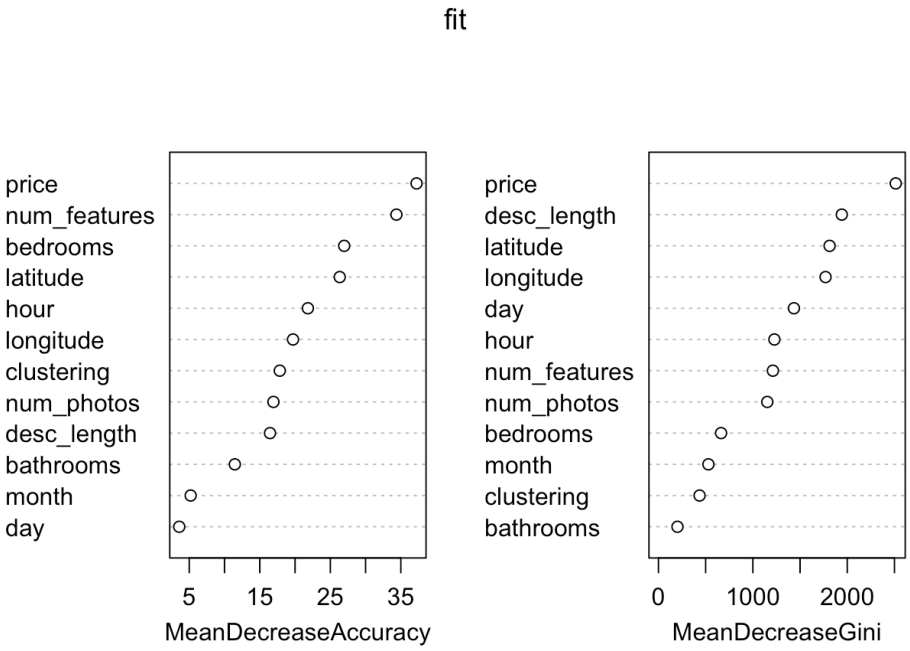

Figure 10: Variable importance as measured by random forest. The right plot is computed by the mean decrease in Gini value. The left plot is computed by mean decrease in accuracy.

Understandably, we see that price takes the first place as the most important variable in determining interest levels followed closely by the number of features of the rental listing. Interestingly enough, the number of bathrooms is ranked very low on the importance value; however, this may be because bedrooms is highly collinear with this variable (a listing with a high number of bedrooms will also likely have a high number of bathrooms and vice versa) and bedrooms is noted as rather important in terms of interest. The least important variables are month and day by the mean decrease accuracy metric.

We tune the model by removing variables that are considered the least important. Variables are removed until the Log Loss score stops decreasing. Through this, we simply remove month and day.

3.6 Metric for Evaluation of Model

At this point, all the data points have a classification so we must have a metric to define how "good" our model and its classification is. We define two metrics: the true positive rate (TPR) given by the confusion matrix and the log loss metric. The TPR is the total number of correct classifications over the total number of classifications performed. Therefore, the higher this rate, the better the model.In particular, we wish this number to be higher than random guessing. In our case, the best type of random guessing would be guessing all listings have low interest (which would yield 69% correct classifications as by Figure **??**).

The other metric we will be using in this paper is multi-class log loss metric. This metric is computed through a formula that quantifies accuracy by penalizing false classification. Each data point is run through the model and the model outputs probabilities for each class. For instance, a data point could have probabilities 85%, 10%, and 5% for classes 0, 1, and 2 respectively. This gives an overview of not only whether the correct class is picked, but also how accurate the probabilities are for the other classes. The equation is given by

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathbb{1}_{i,j}\ log(p_{ij})$$

where N is the number of data points in the dataset and M is the number of possible classes (3 in our case). $\mathbb{1}_{i,j}$ is an indicator where if $i = j$, then the value is 1 and 0 otherwise, i.e. this indicator function turns on if j is the correct class for the i-th data point in the dataset. Finally, p_{ij} is the probability for class j for the i-th data point.

In order to obtain a more accurate model, we attempt to minimize this log loss metric.

3.7 Model Evaluation

By this point, we have cleaned the data, performed feature engineering, explored the dataset, selected parameters, added clustering to our model, and discussed model evaluation. We now use random forest to predict the classifications of the rental listings and tune the random forest model using the metric described in the previous section. For consistency, we use 1600 trees for all of our models.

3.7.1 Naive Model

Here the naive model that we used to determine the important variable in the Selection of Variables section is used. This model includes all of the covariates available to us. Computing random forest, we obtain the following confusion matrix:

Table 1: Confusion matrix for naive model. The sum across rows is the total amount the model predicted for that entry.

	0	**1**	**2**
0	10642	2499	527
1	687	1058	512
2	56	167	282

The true positive rate is 0.7292757. We measure the log loss metric to be 0.8999.

3.7.2 Adding the Clustering variable

For this model, we cluster our dataset with 10 clusters and add a new variable to our dataset called cluster and run random forest on all of the covariates in addition with this cluster variable. We obtain the following confusion matrix where the sum across rows is the total amount the model predicted for that entry:

Table 2: Model with the clustering variable

	0	**1**	**2**
0	10599	2516	551
1	712	1050	496
2	58	178	270

The true positive rate is 0.7254413. We measure the log loss metric to be 0.62136. Although we did not increase the true positive rate, we see that through the log loss metric, we increase the accuracy of our predictions through the probabilities. Also it is interesting that creating more clusters seems to decrease the log loss metric. We perform the same clustering technique except with 50 clusters. The results are as follows:

Table 3: Random forest with 50 clusters

	0	**1**	**2**
0	10727	2563	510
1	659	986	416
2	68	211	290

TPR: 0.7305539 Log-Loss: 0.61688

This reduces our Log-Loss value just by a little bit but not significantly

3.7.3 Variable Selected model

We now take out the month and day variables to prevent overfitting of the model. Furthermore, we continue to use 50 clusters for the clustering variable. The following are the results:

Table 4: Random forest with 50 clusters

	0	**1**	**2**
0	10635	2482	498
1	745	1038	499
2	71	166	296

TPR: 0.7284845 Log-Loss: 0.61489

Again, we manage to increase the log loss metric ever so slightly by reducing the amount of overfitting in the model.

3.7.4 Many Trees Model

Finally, we take a look at the impact of many trees. Instead of 1600 trees, we increase this amount to 8000 trees. In order to make this computation quicker, we use the `doSNOW` and `doMC` packages in order to run multiple random forests on the processors after which the forests are aggregated and merged into one larger forest. The table shown in 5 is the confusion matrix.

Table 5: Random forest with 8000 trees

	0	**1**	**2**
0	10612	2502	551
1	726	1104	446
2	59	164	266

We have the following values for our model: TPR: 0.7292757 Log Loss: 0.61559

Interestingly enough, at a certain point of number of trees, we hit a limit and no longer improve our Log Loss score. Obviously, if we grow a forest of only 2 trees, our log loss score will be much higher than what we obtained. However, what this shows is that there is some asymptotic value after which it does not matter how many trees are grown to solve the problem.

4 Ordinal Logistic Regression

4.1 Motivation and Overview

In this section of the paper, we now consider using an ordinal logistic regression model to predict the interest level of users on the website. To justify the use of this model, consider Y_i^* which is the true interest for the i^{th} apartment listed on the site. In particular, Y_i^* can take values on the continuous interval $[0, 100]$ where 0 represents absolutely no interest in the apartment and 100 represents the most interest an apartment could possibly have. While it would be ideal to take measurements of Y_i^*, this task would be extremely difficult if not impossible.

To get around this problem, we divide this continuous interval of interest up into three segments: one of lower interest, one of medium interest, and one of high interest. We can then approximate the interest level with Y_i, the segment of the interval that Y_i^* falls into. This set up suggests that an ordinal logistic model should be used since we are presented with an ordered list of categories and would like to know the probabilities of falling into the different possible categories given some observations.

In order to make an ordinal logistic model for this, we define π_0, π_1, and π_2 to be the probability that a given apartment has low, medium, or high interest respectively. With this, the model becomes the following where p is the number of predictors:

$$log\frac{\pi_0}{\pi_1 + \pi_1} = \beta_{0,0} - (\beta_1 x_1 + \ldots + \beta_p x_p)$$
$$log\frac{\pi_0 + \pi_1}{\pi_2} = \beta_{0,1} - (\beta_1 x_1 + \ldots + \beta_p x_p)$$

In other words, we are finding the log odds of there being low interest and at least medium interest. An important thing to note is that coefficients β_1 through β_p are the same for each model. Having this implicitly assumes that the predictors have the same effect on the different odds. This assumption, referred to as the *proportional odds assumption*, will be checked for later on.

4.2 Initial Remarks

To begin, we first must consider what subset of the data we want to consider for possible covariates in our model. Because just looking purely at the longitude and latitude for a given apartment may be deceiving, we will disregard these values now. However, the location of the apartment will be considered in a later section of the paper.

The time stamp of when the apartment was posted poses a problem for this model. Ideally each field of the time stamp (i.e. hour, date, etc.) would be considered as a factor; however, the number of levels of each factor would increase the complexity of the model dramatically. Furthermore, we have previously seen that the most important field of the time stamp is the hour the listing was posted. As such, we will only consider the hour the listing was posted and whether it was in the first, second, third, or fourth quarter of the day. Grouping the hours this way follows the trend that was previously seen, validating this decision.

Lastly, the decision is made to treat the number of bedrooms and bathrooms as factors. This decision makes sense since the majority of apartments will have at most four bedrooms or bathrooms. For apartments with more bedrooms or bathrooms, we create a "five or more" level in the factor.

With this established, the data is processed in order to make it more suitable for use. In particular, let X be the design matrix for the data in question. Building on this, let Z be a transformation of X where the column of 1s is removed and the columns for the numerical data is standardized.

With this matrix Z, one can check for the presence of multicollinearity. First, variation inflation factors (VIF) are computed for the numerical covariates. Note that VIF cannot be computed for the number of bedrooms, bathrooms, or the hour. This is because we are treating these covariates as factors and there is

no equivalent idea of adjusted R^2 when these are considered to be the responses. The values found in Table 6 suggest that there is no multicollinearity for the considered numerical covariates since their VIF is well below 10.

Covariate	**VIF**
Price	1.006340
Number of Features	1.294438
Number of Photos	1.080491
Description Length	1.276436

Table 6: Table of VIF for the numerical covariates.

That being said, it was found that the condition number of $Z^T Z$ is 31.61106. This value is above the rule of thumb of 30. Unfortunately since the data we are presented with is mixed between factors and numerical, we cannot use principal component analysis to deal with the multicollinearity. However, we should be wary of multicollinearity and its effects moving forward.

4.3 Model Selection

To start with, we consider the ordinal logistic model with all possible predictors included. Furthermore, we consider interaction terms between price and bedrooms, price and bathrooms, and price an number of features. These interaction terms are included because one would imagine that a user would perhaps be willing to pay more based on these things. Denote this model as M_{full}.

Looking at the p-values for the t-test of each of these coefficients, it seems that the interaction term between the number of features and price may not be necessary. We therefore do a likelihood ratio test between M_{full} and M_{red}, where M_{red} is the model excluding the aforementioned interaction term.

Listing 2: Likelihood Ratio Test for Price:Number of Features

```
Likelihood ratio test

Model 1: interest_level ~ price + num_photos + desc_length + num_features +
    bedrooms + bedrooms:price + bathrooms + hour + bathrooms:price +
    num_features:price
Model 2: interest_level ~ price + num_photos + desc_length + num_features +
    bedrooms + bedrooms:price + bathrooms + hour + bathrooms:price
  #Df  LogLik Df  Chisq Pr(>Chisq)
1  36 -8394.6
2  35 -8395.2 -1 1.1323     0.2873
```

The p-value for this test (shown in Listing 2) is such that we cannot reject the null hypothesis; thus we take M_{red} to be our new model. As further justification for this choice, one can see that the Akaike Information Criteria (AIC) for M_{full} is 17139.49, and the AIC for M_{red} is 17137.49 (lower scores are preferred). Re-examining the p-values for this new model, it seems that bathrooms does not seem to be important. Therefore, another likelihood ratio test is performed in order to judge the importance of the predictor.

This time it seems that bathrooms is needed in the model (see Listing 3). That being said, the standard error for the bathroom coefficients is quite large and the 95% confidence interval includes 0. The same is true for the interaction term between price and number of bathrooms. This high standard error and the fact that the likelihood ratio test does not agree with the t-test may be do to the multicollinearity mentioned previously. This makes sense because one would expect that bathrooms would be closely correlated with the number of bedrooms and the other predictors in the dataset. Therefore, any term dealing with the number of bathrooms will be removed from the model. The details of this final model are shown in Table 7.

Listing 3: Likelihood Ratio Test for Bathrooms

```
Likelihood ratio test

Model 1: interest_level ~ price + num_photos + desc_length + num_features +
    bedrooms + bedrooms:price + bathrooms + hour + bathrooms:price
Model 2: interest_level ~ price + num_photos + desc_length + num_features +
    bedrooms + bathrooms:price + bedrooms:price + hour
  #Df  LogLik Df  Chisq Pr(>Chisq)
1  35 -8395.2
2  27 -8440.1 -8 89.849  4.991e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1  1
```

Coefficient	Value	Std. Error	p value	Odds Ratio	2.5%	97.5%
price	-19.22836624	0.78518378	1.934743e-132	4.458889e-09	-20.76729817	-17.68943431
num_photos	0.08044929	0.02408406	8.367164e-04	1.083774e+00	0.03324540	0.12765318
desc_length	0.16576739	0.02428495	8.735636e-12	1.180299e+00	0.11816977	0.21336501
num_features	0.28889022	0.02424616	9.900091e-33	1.334945e+00	0.24136863	0.33641182
bedrooms1	1.17962944	0.67483631	8.046003e-02	3.253168e+00	-0.14302543	2.50228431
bedrooms2	-3.03126114	0.61465848	8.155107e-07	4.825474e-02	-4.23596962	-1.82655265
bedrooms3	-1.26772530	0.42715313	2.998932e-03	2.814712e-01	-2.10493005	-0.43052056
bedrooms4	-0.54780725	0.23127531	1.785377e-02	5.782163e-01	-1.00109854	-0.09451597
bedrooms5+	-0.30554191	0.09694507	1.623223e-03	7.367240e-01	-0.49555077	-0.11553306
hour1	0.45207723	0.08298340	5.099567e-08	1.571573e+00	0.28943276	0.61472170
hour2	-0.33869361	0.07357045	4.151230e-06	7.127008e-01	-0.48288905	-0.19449817
hour3	0.09187140	0.06330738	1.467254e-01	1.096224e+00	-0.03220879	0.21595159
price:bedrooms1	38.18819474	2.51000393	2.838251e-52	3.845229e+16	33.26867744	43.10771204
price:bedrooms2	-8.57109798	2.26489438	1.541259e-04	1.895045e-04	-13.01020939	-4.13198657
price:bedrooms3	0.73668328	1.76462577	6.763326e-01	2.088995e+00	-2.72191967	4.19528623
price:bedrooms4	0.88368379	1.29231538	4.941022e-01	2.419797e+00	-1.64920781	3.41657540
price:bedrooms5+	1.69588454	0.96898788	8.009058e-02	5.451466e+00	-0.20329681	3.59506589
0—1	1.52297563	0.19178218	2.002766e-15	4.458889e-09	-	-
1—2	3.44238446	0.19444103	3.905424e-70	1.083774e+00	-	-

Table 7: Coefficient information for the final ordinal logistic model including the value of the coefficient, the standard error, the p value for the t-test for the coefficient, the odds ratio, and the bounds of the 95% confidence interval.

4.4 Examination of Model

Looking at Table 7, there are several interesting insights that our model gives about the data. For instance, there seems to be a strong relationship between price and interest level. Given a studio apartment (i.e. 0 bedrooms) and with all other predictors fixed, one can expect both odds ratios (low interest odds ratio and at least medium interest odds ratio) to increase by a multiplicative factor of about 4.45×10^9 per standard deviation of price increase (recall these terms are being subtracted off hence the multiplicative factor is greater than 1). In other words, it starts to become more and more probable that the interest level for a studio will be lower as price increases. Note that while this factor might lead one to think that price is the most important predictor, it is tough to actually tell since several expensive apartments make the standard deviation quite large.

One can see the importance of price by estimating the probabilities of seeing low, medium, or high interest levels for a new set of data. Figure 11 shows how the price of an apartment is a dominating predictor in

indicating whether the the apartment will have interest. This is shown by the fact that as price rises, the probability of there being low interest in an apartment seems to converge to 1. This happens despite the number of bedrooms the apartment has.

Figure 11: Plots of the predicted probabilities for a new dataset as a function of price. Here 0 (red) corresponds with low interest, 1 (green) corresponds with medium interest, and 2 (blue) corresponds with high interest. The plots are separated by the number of bedrooms in the apartment where 0 bedrooms is at the top and 5+ bedrooms is at the bottom.

Looking at the other coefficients for the predictors gives several other insights about the data. For instance, it seems that if one wants to post an apartment listing then posting in the second quarter of the day is most likely to lead to the highest interest. Furthermore, it seems that RentHop users are most interested in one bedroom apartments and have the least amount of interest for two bedroom apartments. Lastly, it seems that apartments with more photos, longer descriptions, and more features are more likely to have interest among users.

4.5 Validity of Model

Recall that one of the major assumption taken with this model was the proportional odds assumption. To check this, we do an informal test described by a UCLA tutorial [2]. This test considers two separate logistic models: one where the response is that there was medium interest or higher, and the other where the response is that there was high interest. Additionally, for each predictor we fit both of the responses to only that predictor, and several values of the log-odds in the resulting fitted models are reported. Now given a predictor, the difference between the reported fitted values should remain constant if the proportional odds assumption holds. These differences are shown below in Figure 12.

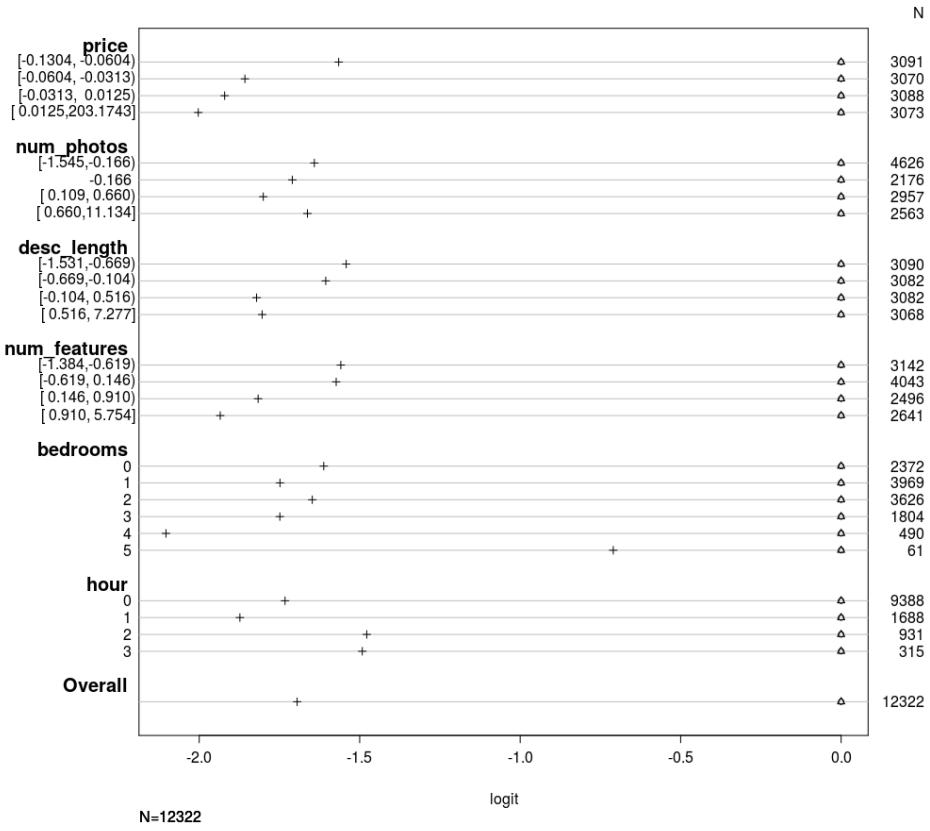

Figure 12: Plot of differences (represented by +) for each predictor. If the proportional odds assumption holds, the points should align for each predictor.

Looking at these difference it seems that most of them are pretty close to each other. There are exceptions, however. For example, after three bedrooms the differences for bedroom differ pretty drastically. Furthermore, it seems like there may be some trend in price since the magnitude of the difference increases as price increases. Because of this, it could be the case that the proportional odds assumption is violated. To further investigate this, one could try performing nominal logistic regression to see if that has better performance.

4.6 Evaluation of Model

It would be beneficial to have a metric that quantifies how well the model fits the data. For this we turn to McFadden's Psuedo R-squared value that will be denoted as R_p^2. This value ranges between 0 and 1 and is equal to the following:

$$R_p^2 = \frac{\ell(M_{min}) - \ell(M)}{\ell(M_{min})}$$

Here M is our final model, M_{min} is the minimum model with only constants and no predictors, and $\ell(M)$ is the value for the log likelihood for the model. In the case of this model, it was found that $R_p^2 = 0.1236374$. While this may look abysmal it should be noted that R_p^2 scores are often lower. According to McFadden, scores between 0.2 and 0.4 represent excellent fit [3]. Therefore, while this model might fit well, it seems that it does have some explaining power.

Like before, we also do cross validation and submit results to Kaggle for evaluation. Overall, it is found that the TPR is 0.6916 and the log loss is 0.70294. Based on these scores, it seems that random forests does a better job at predicting interest levels. That being said, this model does not include any spatial information.

	0	1	2
0	24391	7236	1984
1	1264	1114	839
2	32	44	62

Table 8: Confusion matrix for ordinal logistic regression model. The sum across rows is the total amount the model predicted for that entry.

4.7 Clustering Neighborhoods

As mentioned before, this model ignores all spatial data, which is an important feature in our dataset. In order to remedy this we reuse the CLARA clustering algorithm to group apartment listings by location into 10 different neighborhoods. We choose to refit the model to each of the 10 different clusters since the relationship between interest and each of the predictors may vary drastically based on the cluster. This leaves us with 10 estimates of coefficients as one can see in Figure 9 and Figure 10.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
price	-36.35	-35.32	-32.75	-31.63	-22.31
num_photos	0.11	0.26	0.18	0.09	0.16
desc_length	0.22	0.27	0.33	0.12	0.26
num_features	0.30	0.56	0.74	0.38	0.60
bedrooms1	2.80	-3.82	3.18	3.83	0.92
bedrooms2	-2.22	-10.74	-2.91	-0.24	-4.53
bedrooms3	0.10	-6.18	-0.47	-0.93	-1.99
bedrooms4	-0.40	-3.31	-0.24	-0.06	-0.92
bedrooms5+	0.17	-0.82	0.10		-0.27
hour1	0.48	0.62	0.67	0.35	0.91
hour2	-0.04	-0.43	-0.40	-0.08	-0.35
hour3	-0.14	-0.28	0.09	0.28	-0.05
price:bedrooms1	46.13	58.18	46.67	41.38	40.42
price:bedrooms2	-0.45	-9.60	-13.30	14.85	-8.15
price:bedrooms3	16.21	8.92	0.79	-4.18	8.43
price:bedrooms4	5.56	-1.57	-2.31	-4.40	-2.64
price:bedrooms5+	8.92	4.12	4.60		0.38
R2.McFadden	0.14	0.18	0.17	0.14	0.16

Table 9: Results of training model on first 5 clusters. Note that missing entries are from there being no apartments with 5 bedrooms

	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
price	-28.64	-29.22	-21.83	-77.63	-93.79
num_photos	0.16	0.23	0.18	0.42	0.20
desc_length	0.09	0.19	0.14	0.08	0.31
num_features	0.34	0.39	0.21	-0.17	-0.03
bedrooms1	-4.90	3.97	1.89	1.99	-2.80
bedrooms2	-10.79	-1.63	-1.35	-10.11	5.73
bedrooms3	-5.97	-0.53	-0.08	-5.18	-7.71
bedrooms4	-3.01	-0.01	0.14	-1.40	3.10
bedrooms5+	-1.09	0.21	-0.04	-2.37	
hour1	-0.26	0.68	0.87	0.75	-0.01
hour2	0.06	-0.41	-0.27	-0.55	-0.62
hour3	0.20	-0.38	0.05	0.35	-0.36
price:bedrooms1	58.23	37.44	20.85	85.53	-73.42
price:bedrooms2	-16.72	-3.48	-16.38	-3.81	32.98
price:bedrooms3	2.42	8.86	-8.25	22.73	-87.27
price:bedrooms4	1.89	-7.27	-2.50	23.19	21.93
price:bedrooms5+	-2.30	-10.01	-1.97	-9.74	
R2.McFadden	0.20	0.16	0.11	0.19	0.21

Table 10: Results of training model on last 5 clusters. Note that missing entries are from there being no apartments with 5 bedrooms

One can see that by separating the data by neighborhood we are able to achieve better fitting models. This is apparent from the fact that the R_p^2 values for the models are generally higher than the previous, single model. Furthermore, it seems that the majority of the clusters exhibit the same relationship between interest level and the predictors as the original model, although to varying degrees. An exception to this seems to be cluster 10. As opposed to what has been previously seen, it seems that users seeking apartments in this area are more interested in two bedroom apartments than one bedroom apartments. More investigation needs to be performed in order to understand why that neighborhood would have a different trend than the rest.

While this seems to be a superior method than having a single method (unfortunately cross validation could not be done to show this), there are some improvements that could be made. For instance, it was assumed that the previous model found is still the best model for each of the different clusters; however, there may be more fine tuning that needs to be done in order to better each model. Additionally, in the future it may be beneficial to create a better clustering algorithm that will decide cuts between neighborhoods based on other factors than solely location in order to best separate out the different types of apartments users are looking for.

Lastly, since it seems that many of the relationships between the log odds and the predictors seem to hold across clusters, the attempt was made to include cluster number as a factor. However, this significantly decreased the accuracy of the model, and thus cluster number as a factor was excluded.

5 Conclusion

In this paper we have explored predicting interest for apartment listings using two different models: random forests and ordinal logistic regression. While ordinal logistic regression is more interpretable (i.e. can analyze coefficients for predictors), it seems that the final random forest model performs a good deal better than the ordinal logistic regression model. Not only were we more easily able to incorporate spatial information with random forests, but it is likely that random forests picks up on some intricacies that ordinal logistic regression cannot. However, since each model has its strengths, it would most likely be best two have an ensemble model that incorporates some weighting between the two models.

References

[1] "Home-ownership in the united states." https://en.wikipedia.org/wiki/Home-ownership_in_the_United_States. Accessed: 2017-05-05.

[2] "Ordinal logistic regression — r data examples." http://stats.idre.ucla.edu/r/dae/ordinal-logistic-regression/. Accessed: 2017-05-05.

[3] "Mcfadden's pseudo-r2 interpretation." https://stats.stackexchange.com/questions/82105/mcfaddens-pseudo-r2-interpretation. Accessed: 2017-05-05.