Team 7 – Predicting Interest Level for Online Apartment Listings

Ahmet Hatip, Zhao Lan, Di Qin, Scott Smith, Shuming Sun
Department of Statistics and Operations Research
Department of Computer Science
University of North Carolina at Chapel Hill

April 27th, 2018

Summary

Background: Predicting the level of interest in rental listings is important for rental agencies, since this can help agencies understand how to create listings that generate interest. The objective of this project is to predict the interest level for apartment listings provided by RentHop, a New York rental listing agency.

Features: The original data set consisted of both simple features that are directly usable and complex features that need feature engineering to preprocess. The simple features include monthly rent, number of bedrooms, bathrooms, etc. The complex features include text description, apartment characteristics and image information.

Challenges and Solutions: We faced three major challenges. First, the classes of response variables were very unbalanced in our dataset. The solution that yielded the best result to this challenge was under sampling the majority class. Secondly, the image and textual data provided additional challenges in terms of preprocessing needs. We used the VGG16 neural network model (the Keras library in Python) to extract 1000 element vectors from images. Then PCA was used to extract the top 61 Principal Components as features for each listing. For the natural language processing of the textual descriptions, we used the Syuzhet package in R to generate positive and negative sentiment scores as well as ratings on each of the eight universal emotions. Lastly, ordinal classification proved to be another challenging area. Since different types of misclassifications did not deserve the same penalties, the assignment of different penalties to different classification errors would have been necessary for our problem setting. We circumvented this issue due to time constraints by aggregating medium and high interest level to "not low" response level and leaving low interest level as "low".

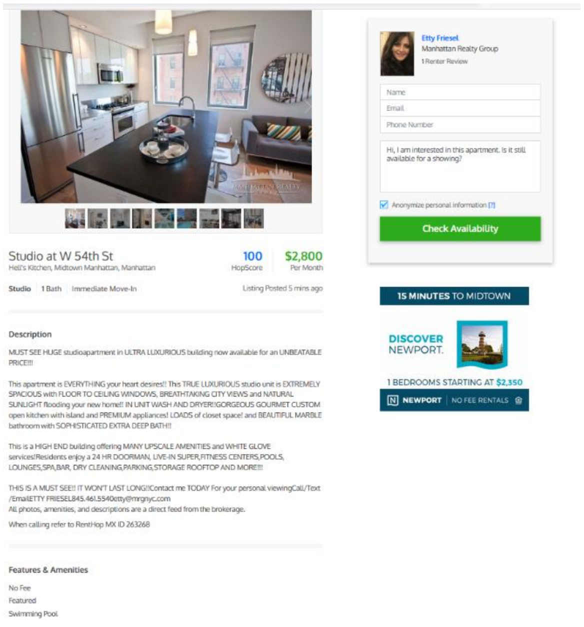
Conclusion: We tested several different models and compared their performance based on overall accuracy as well as class specific accuracy. We found boosting and bagging methods to be the most accurate in terms of the overall accuracy. However, logistic regression with the aid of under-sampling was the most accurate for predicting the minority class. Important predictors include price, several apartment characteristcs, sentiment, and image features.

There are many avenues to increase the overall accuracy and class specific accuracy of our methods. We believe that a true ordinal classification algorithm that is capable of classifying the data into more than two categories with proper assignment of class sensitive penalties could be more useful to our client. In addition, we could potentially improve our results by using over-sampling rather than under-sampling, training an aesthetics model to extract image features, and fine-tuning the classification threshold to meet specific business objectives.

1.0 Introduction

1.1 Description of Problem

RentHop is an apartment listing company operating in New York City. Most of their business is conducted for rental listings in Manhattan, Brooklyn (areas near Manhattan) and Queens (areas near Manhattan). The listings are presented online via RentHop's website and consist of an image, part of the address, a text description, a list of apartment characteristics, and the monthly rent. A link is also provided for the customer to view more pictures and for the customer to 'check availability' by getting in touch with a real-estate agent. Figure 1 shows the top portion of a sample page. RentHop has provided us with all the data on the screen as well as other background data they have for each listing.



Studio at W 54th St
Hell's Kitchen, Midtown Manhattan, Manhattan

100 HopScore | $2,800 Per Month

Studio | 1 Bath | Immediate Move-In | Listing Posted 5 mins ago

Description

MUST SEE HUGE studioapartment in ULTRA LUXURIOUS building now available for an UNBEATABLE PRICE!!!

This apartment is EVERYTHING your heart desires!! This TRUE LUXURIOUS studio unit is EXTREMELY SPACIOUS with FLOOR TO CEILING WINDOWS, BREATHTAKING CITY VIEWS and NATURAL SUNLIGHT flooding your new home!! IN UNIT WASH AND DRYER!!GORGEOUS GOURMET CUSTOM open kitchen with island and PREMIUM appliances! LOADS of closet space! and BEAUTIFUL MARBLE bathroom with SOPHISTICATED EXTRA DEEP BATH!!

This is a HIGH END building offering MANY UPSCALE AMENITIES and WHITE GLOVE services!Residents enjoy a 24 HR DOORMAN, LIVE-IN SUPER,FITNESS CENTERS,POOLS, LOUNGES,SPA,BAR, DRY CLEANING,PARKING,STORAGE ROOFTOP AND MORE!!!

THIS IS A MUST SEE!! IT WON'T LAST LONG!!Contact me TODAY For your personal viewingCall/Text /EmailETTY FRIESEL845.461.5540etty@mrgnyc.com
All photos, amenities, and descriptions are a direct feed from the brokerage.

When calling refer to RentHop MX ID 263268

Features & Amenities

No Fee
Featured
Swimming Pool

Etty Friesel
Manhattan Realty Group
1 Renter Review

Name
Email
Phone Number

Hi, I am interested in this apartment. Is it still available for a showing?

[✓] Anonymize personal information [?]

Check Availability

One of RentHop's business objectives is to entice customers to click the 'check availabilty' button to inquire about apartments. We suspect that their buisness model includes payment for 'clicks'. RentHop has requested that we build a model to predict the volume of customers clicking on this button.

RentHop has structured this request as an ordinal classification problem. They have divided inquiry rate into three catgories: 'low', 'medium', and 'high' and requested that we make our predictions based upon these three categories.

Figure 1: The top portion of a rental listing page on the RentHop site, showing an image, a text description, a list of apartment characteristics, a partial address, and a button to 'check availability'.

1.2 Data

RentHop has provided us with data for 49,352 listings, along with the click-thru rate for each of these listings (categorized as 'low', 'medium' and 'high').

1.2.1 Directly Usable Data:

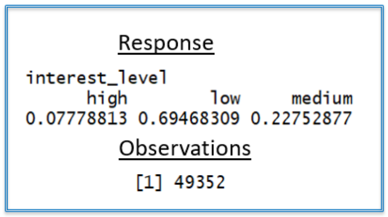
- Monthly rent
- Number of bedrooms
- Number of bathrooms
- Latitude
- Longitude
- Building ID

1.2.2 Data Requiring Feature Engineering

- Text description
- Apartment characteristics
- URLs for images
- Address

1.2.3 Response Data

Response data was provided as either ‘low’, ‘medium’ and ‘high’ representing the inquiry rate for each listing. Figure 2 summarizes the distribution of this response data.



Response

```
interest_level
      high        low     medium
0.07778813 0.69468309 0.22752877
```

Observations

```
[1] 49352
```

Figure 2: Distribution of listing by response data (interest_level)

1.3 Motivation, Challenges, Simplifying Assumptions

1.3.1 Motivation

Besides presenting us with an interesting business problem, this specific project provided us with the unique opportunity to use classification techniques involving data that included more than just numeric and string data. There was one free text field (description) and two semi-structured text fields (apartment characteristics and address). In addition, each listing had several images associated with it. We saw the image and text data as an opportunity to extend what we have learned by applying machine learning concepts to this diverse set of data, and also to experiment with techniques to extract features from this data.

1.3.2 Challenges

The motivations described above provided interesting challenges. First and foremost were the twin challenges of extracting usable features from images and text descriptions. We needed to figure out a way to convert both text and image data into a manageable set of features that would provide our model with meaningful ‘signal’ to use for the classifications. In addition, we needed to figure out how to extract features from the structured text that listed the characteristics of each apartment.

Ordinal classification provided another challenging problem. We needed to not only assign listings to one of three classes, we also needed to figure out a penalty that weighted different classification errors differently. For example, the penalty for incorrectly classifying a ‘low’ as a ‘medium’ or a ‘high’ as a medium should be lower than incorrectly classifying ‘low’ as a ‘high’ or vice-versa.

Extracting a geographic ‘signal’ proved to be demanding. Zip code is a useful indicator of geographic influence on class assignment. However, the ‘address’ field did not contain a zip code.

Lastly, the classes were very unbalanced. Our experience indicated that this would result in models over-predicting the majority class (since the objective is to maximize overall accuracy). We needed to deal with this in order to deliver a useful model.

1.3.3 Simplifying Assumptions

There is plenty of literature on ordinal classification, but we found the existing literature to be difficult to implement given the time constraint. The packages and techniques we tried did not yield useful results in the limited time we had, so we decided to combine the two minority classes ‘medium’ and ‘high’ into a single class and turn the problem into a simpler two class challenge. We felt that our client could still extract significant value from our work by understanding which listings yielded a ‘low’ response rate versus a ‘not-low’ response rate.

For geographical input to the model, we attempted clustering listings based upon latitude and longitude. However, this method yielded a few populous clusters and many clusters with small numbers of listings. We adjusted our algorithm to address this issue, but in the end, adding ‘cluster’ as a feature provided no value to our model so we did not use this feature.

The ‘address’ field did not include zip code but still had the potential of providing us with geographic ‘signal’. There is much written on parsing addresses, but in the short time we had we could not make use of this data. In the end, we settled on using latitude and longitude as the only geographic inputs to the model.

2.0 Project Strategy and Feature Engineering

2.1 Overall Strategy – Two Stages

We approached this project with a two-stage strategy. We felt that a good portion of the useful information would be contained in the image, free-text description, and structured-text apartment features. So, the first stage of the project was feature engineering. Wwe worked with the existing data to extract a manageable set of features from the image, free-text, and structured text fields. In addition, we created interaction variables that we thought would be valuable in our model.

The second stage of the project consisted of the actual modelling. Of the various techniques we had at our disposal, we chose to build models using logistic regression, decision trees, bagging, random forest, and boosting. We chose not to use KNN because of the potential for a large number of features. We also chose not to use LDA or QDA because we didn’t think the distribution of the data is not consistent with the assumptions required to use these techniques (such as normality and homogeneous variance).

2.2 Feature Engineering

2.2.1 Feature Extraction from Images

Each apartment listing had either zero or a small number of images (represented by a series of URLs for jpg files) associated with the listing. Since the listings represented in the data were still available online, we verified that the first URL in the ‘picture’ column corresponded to the picture that was displayed when the listing was opened. Shourabh Rawat from Tulia refers to this first image as the ‘hero image’ and postulates that this image has the highest influence in determining whether a customer will click though to get more information.[1] Zhang, Lee et al from Carnegie Mellon hypothesize that image aesthetics play a key role in encouraging ‘click through’ for Airbnb[2]

We were unable to find a trained model that could extract aesthetic features from the images we had from RentHop (we suspect that these models are proprietary). We were able to find several pre-trained models that could extract features based upon the 1000 ImageNet classes from ImageNet Large Scale Visual Recognition Challenge 2014[3]. We chose to use the VGG16[4] neural network model using the Keras library for Python.

Each ImageNet class represents an object (e.g. ‘African elephant’, 'toaster'), and for each new image VGG16 predicts a 1,000 element vector with each element having a value between 0 and 1 representing the probability that the image is of the specific class. We extracted this 1,000 element for the first image for each listing. (If a listing did not have an image, we set this vector to be all zeros.) We then used PCA to extract the top 61 Principal Components representing 95% of the overall variance. We added these 61 Principal Components as features for each apartment listing.

2.2.2 Feature Extraction from Text

Each listing had a free text column consisting of the description of the apartment listing. We used the Syuzhet[5] package in R to convert the apartment descriptions into eight specific emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust) This package enabled us to assign each listing a score on each of these eight emotions as well as an overall score in terms of positive sentiment, negative sentiment and valence (positive sentiment minus negative sentiment). We added these 11 scores as features for each apartment listing.

2.2.3 Feature Extraction from Structured Text

Each listing had a structured text column containing specific characteristics of the apartment for rent (e.g. ‘hardwood floors’, ‘no-fee’). When we parsed this data, we found over 1,000 different characteristics, many of which were only applicable to very few (or only one) apartments. We sorted these characteristics based on the number of listings using them (and also manually combined similar characteristics) and then selected only those characteristics used by ten or more listings. This left us with 152 unique characteristics. We turned these 152 characteristics into 152 features with binary values. If an apartment had a feature, it received a 1 in the column for that feature. If not, it received a 0 for that feature.

---

[1] What Makes a Photo Click: Selecting Hero Images with Deep Learning
Shourabh Rawat - https://www.trulia.com/blog/tech/selecting-hero-images-with-deep-learning/

[2] Zhang, S., Lee, D., Singh, P. V., & Srinivasan, K. (2017). How Much Is an Image Worth? Airbnb Property Demand Estimation Leveraging Large Scale Image Analytics. *SSRN Electronic Journal*. doi:10.2139/ssrn.2976021

[3] Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). (n.d.). Retrieved from http://image-net.org/challenges/LSVRC/2014/

[4]S., K., Z., & A. (2015, April 10). Very Deep Convolutional Networks for Large-Scale Image Recognition. Retrieved from https://arxiv.org/abs/1409.1556v6

[5] Jockers, M. (2017, December 13). Retrieved from https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html

2.2.4 Other Derived Features

We counted the number of image URLs for each listing and created a feature that contained this count of images. We also created a binary feature indicating whether the 'building id' column was populated or not, as well as features representing rent per bedroom and rent per total rooms (adding 1 to the denominators to account for zero bedrooms).

2.2.5 Data Splitting

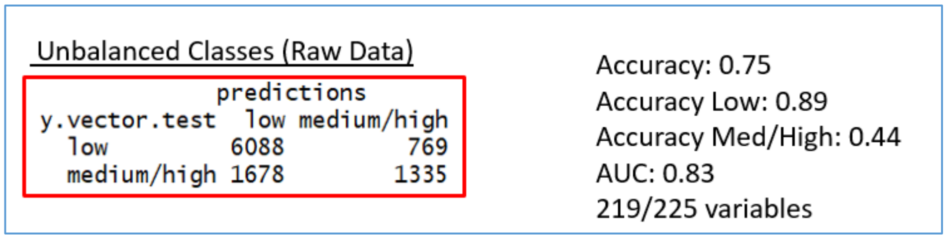
We split the data into 80% training and 20% test data, ensuring that both test and training data had equal proportions of the response variable ('low' or 'medium/high').

3.0 Solution – Model Details

3.1 Logistic Regression Models

We built an L1 penalized logistic regression model (Lasso) and an L2 penalized logistic regression model (Ridge) using 5-fold cross validation to select the best value of lambda. We then used this model to predict the response variable for the test data.

The Ridge model did not yield better characteristics than the null classifier so we abandoned the Ridge model. Figure 3 below provides a summary of our results for the **Lasso** model.



Unbalanced Classes (Raw Data)

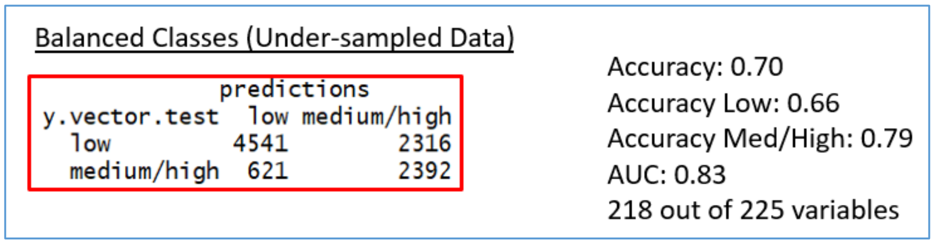
```
               predictions
y.vector.test   low medium/high
  low          6088         769
  medium/high  1678        1335
```

Accuracy: 0.75
Accuracy Low: 0.89
Accuracy Med/High: 0.44
AUC: 0.83
219/225 variables

Figure 3: Confusion Matrix and accuracy measures for logistic regression maintaining unbalanced classes.

The Lasso model over-predicted the majority class 'low' and had very low accuracy in predicting listings in the minority class 'medium/high'. Overall accuracy was better than the null classifier by over 5 percentage points.

We decided to try 'under-sampling' the training data to balance the classes. This involved only keeping a subset of the training data from the majority class 'low' such that the training data had an approximately equal proportion of 'low' listing and 'medium/high' listings. We then re-ran the Lasso cross-validation to again select the best value of lambda and then made predictions on the test data. The results are shown in Figure 4 below:



Balanced Classes (Under-sampled Data)

```
               predictions
y.vector.test   low medium/high
  low          4541        2316
  medium/high   621        2392
```

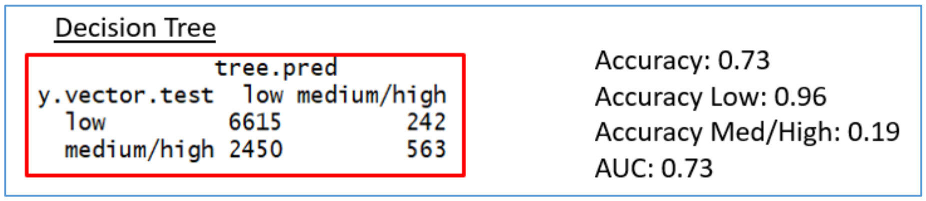
Accuracy: 0.70
Accuracy Low: 0.66
Accuracy Med/High: 0.79
AUC: 0.83
218 out of 225 variables

Figure 4: Confusion Matrix and accuracy measures for logistic regression model using balanced classes

The results were very interesting. Although this model had only slightly better accuracy than the null model, it had excellent accuracy predicting ‘medium/high’ listings correctly and had reasonable accuracy predicting ‘low’ listing correctly (although less than the null-classifier). This model could be very useful if the cost of a prediction error is high for ‘medium/high’ listings

3.2 Decision Tree, Bagging, and Random Forest Models

Figure 5 below shows the results of the pruned decision tree model. Note that this model used the raw (unbalanced) training data.



Decision Tree

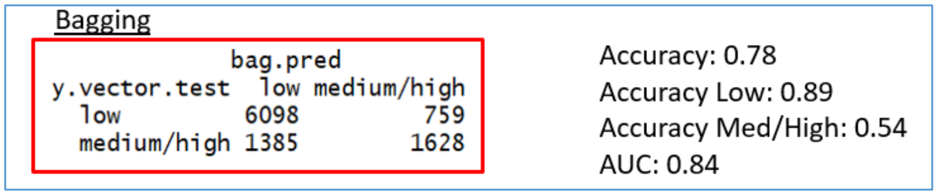
```
                 tree.pred
y.vector.test   low medium/high
  low          6615         242
  medium/high  2450         563
```

Accuracy: 0.73
Accuracy Low: 0.96
Accuracy Med/High: 0.19
AUC: 0.73

Figure 5: Confusion Matrix and accuracy measures for pruned decision tree model using un-balanced classes

The decision tree model was only slightly better than the null model, over-predicting ‘low’ and performing very poorly on ‘medium/high’. As a follow-on, we would like to build a decision tree model using under-sampled training data.

Figure 6 below shows the results of the bagging model. Note that this model used the raw (unbalanced) training data



Bagging

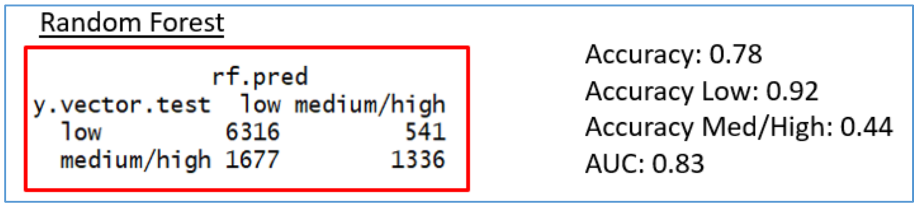
```
                 bag.pred
y.vector.test   low medium/high
  low          6098         759
  medium/high  1385        1628
```

Accuracy: 0.78
Accuracy Low: 0.89
Accuracy Med/High: 0.54
AUC: 0.84

Figure 6: Confusion Matrix and accuracy measures for bagging model using un-balanced classes

The bagging model performed very well, with a much higher accuracy than that of the null classifier. Accuracy was very high when predicting listings with a ‘low’ inquiry rate and was mediocre predicting listings with a ‘medium/high’ inquiry rate. If the cost of an error for the majority ‘low’ class is high compared to the minority ‘medium/high’ class, this might be a good model to use.

Figure 7 below shows the results of the random forest model. We used the approximate square root of the number of predictors to arrive at our feature subset number of 16. Note that this model used the raw (unbalanced) training data



Random Forest

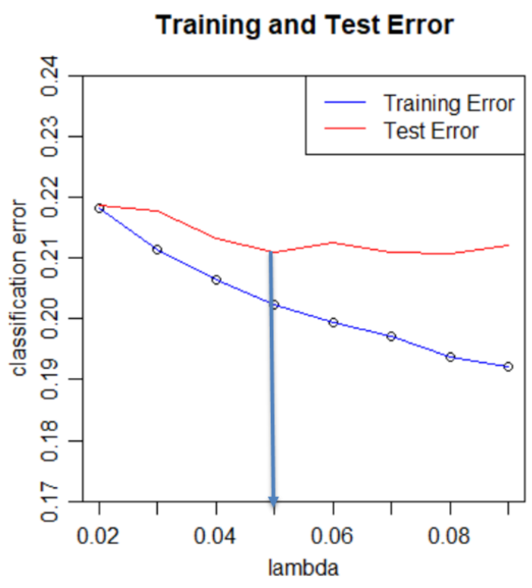
```
                 rf.pred
y.vector.test   low medium/high
  low          6316         541
  medium/high  1677        1336
```

Accuracy: 0.78
Accuracy Low: 0.92
Accuracy Med/High: 0.44
AUC: 0.83

Figure 7: Confusion Matrix and accuracy measures for random forest model using unbalanced classes

The random forest model performed similarly to the bagging model but was more skewed to over-predicting the majority ‘low’ class. If the cost for an error for the ‘low’ class is very high compared to the minority ‘medium/high’ class, this may be a good model to use. Again, this model was run on the raw training data.

3.3 Boosting Model



For boosting, we ran cross-validation to arrive at the best value for the shrinkage parameter lambda of 0.05. Figure 8 to the left shows the results in terms of both training and test error. We then used this value of lambda to build a model and predict classes for the test data. Note that this model used the raw (unbalanced) training data.

The results for the boosting model are seen in Figure 9 below. Overall, boosting provided very similar results as bagging and would again be a good choice if the cost of an error on the majority class ‘low’ is higher than that on the minority class ‘medium/high’.

Figure 8: For the boosting model, training error declines monotonically, and test error is minimized when lambda=0.05

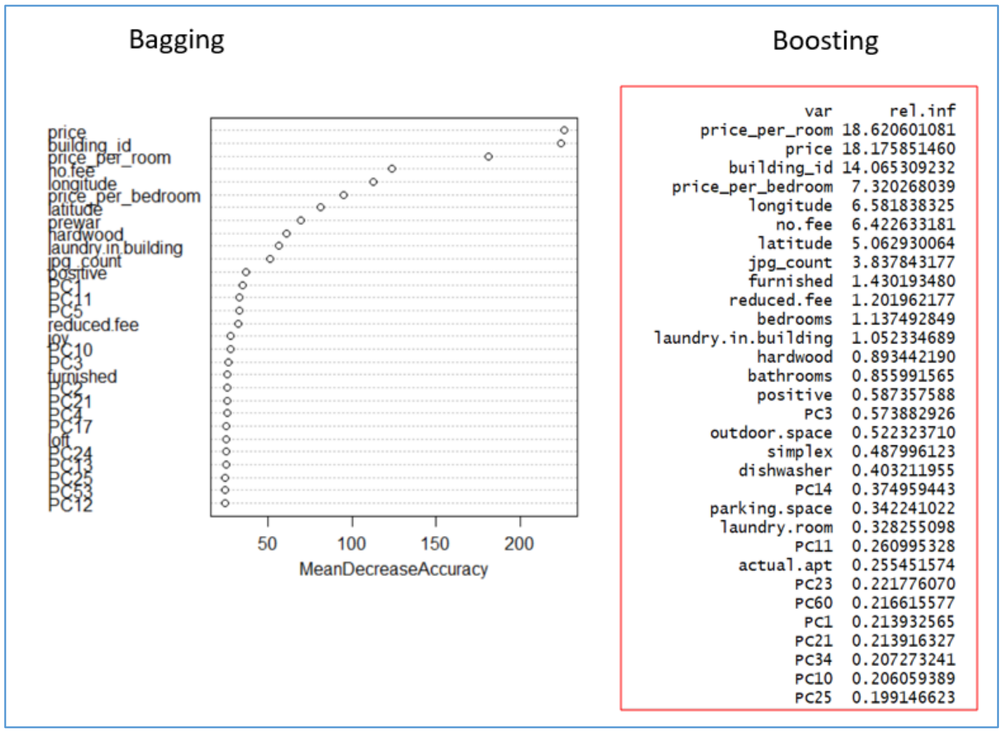
Boosting (1=‘low’, 0 = ‘medium/high’, lambda=0.05)

```
             predictions
y.binary.test    0    1
            0 1583 1430
            1  696 6161
```

Accuracy: 0.78
Accuracy Low: 0.90
Accuracy Med/High: 0.53
AUC: 0.84

Figure 9: Confusion Matrix and accuracy measures for boosting model using un-balanced classes

Figure 10 below lists the most important features for bagging (on the left) and boosting (on the right). We can see that the lists are very similar. Price seemed to play the most important role, with characteristics such as hardwood floors and no rental fee also figuring significantly in our models. Interestingly, positive sentiment was also important, along with the number of pictures in the listing and many principal components of the first image. An interesting finding is that latitude and longitude showed up as very important features in both bagging and boosting. This is because these methods, along with other tree based methods, can create small hyper-cubes in the feature space to differentiate class predictions to a very fine level. In New York City, longitude and latitude seem to have defined small rectangles in the geographic space that had significant influence on the predicted inquiry level.



```
                var    rel.inf
     price_per_room 18.620601081
              price 18.175851460
        building_id 14.065309232
  price_per_bedroom  7.320268039
          longitude  6.581838325
             no.fee  6.422633181
           latitude  5.062930064
          jpg_count  3.837843177
          furnished  1.430193480
        reduced.fee  1.201962177
           bedrooms  1.137492849
laundry.in.building  1.052334689
           hardwood  0.893442190
          bathrooms  0.855991565
           positive  0.587357588
                PC3  0.573882926
      outdoor.space  0.522323710
            simplex  0.487996123
         dishwasher  0.403211955
               PC14  0.374959443
      parking.space  0.342241022
       laundry.room  0.328255098
               PC11  0.260995328
         actual.apt  0.255451574
               PC23  0.221776070
               PC60  0.216615577
                PC1  0.213932565
               PC21  0.213916327
               PC34  0.207273241
               PC10  0.206059389
               PC25  0.199146623
```

Figure 10: The graph on the left shows the most important variables for bagging, while the graph on the right shows the most important variables for boosting.

4.0 Conclusion and Lessons Learned

It's apparent that boosting and bagging yielded the highest overall accuracy, area under the ROC curve (AUC) and a reasonable balance of accuracy across the two classes. That being said, logistic regression with under-sampling yielded the best balance of accuracy across the two classes but had reduced overall accuracy due to less training data.

Overall, we conclude that there is no best model for all settings. The relative costs of errors for each class should play an important part in model selection.

We also learned that while under-sampling improved the accuracy balance between the majority and minority classes, overall accuracy decreases due to fewer training observations.

To improve the model and its usefulness to the client, we recommend building a true ordinal classification model that takes into account the different errors for different misclassifications. In addition, we think accuracy could be improved if we could make use of the address data to model more geographic effects. In addition, training a real aesthetics image model (on top of the VGG16 feature extraction layers) would be beneficial. Lastly, over-sampling the minority class in the training data has the potential to balance the classes without impacting overall accuracy. Final model tuning could be accomplished by moving around the ROC curve, varying the threshold for predicting each class so the model could be fine-tuned to meet the business needs of our client.