**Rental Listing Inquiries**

1. **Project Plan**
   1. **Overview**

Most people had rental experiences. In order to live into an ideal apartment, we often draw support from housing rental websites. Some agents used these websites to post advertisements. However, different advertisements can bring different numbers of click. We are all interested by this phenomenon, so we decided to find out the reason of it.

Two Sigma has launched a competition on Kaggle to predict rental listing popularity. It provides us a very accuracy dataset, which contains about 40,000 instances. Data comes from an apartment listing website, called renthop.com. These apartments are located in New York City. For more details, please refer to:

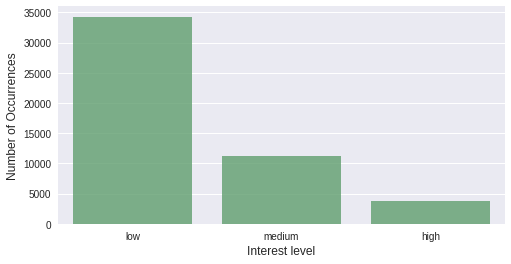
https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/data

Doing so will help agents predict the popularity of a rental listing based on various features, at the same time, it will allow agents better handle fraud control, identify potential listing quality issues, and understand renters’ needs and preferences.

After we get the dataset, we decided to analyze the original data first.

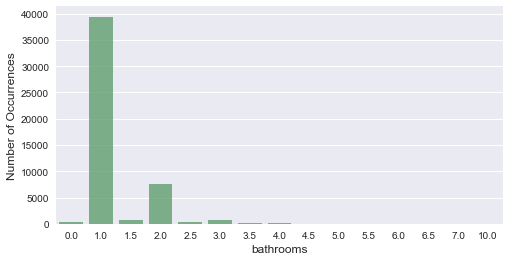
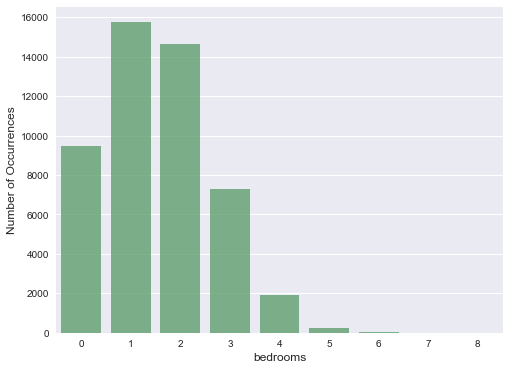
Each instance has 13 features, bathrooms, bedrooms, latitude, longitude, price, photos, description, created time, listing id, manager id, building id, address, and interest level.

Interest level is the target variable. It means the numbers of click, and it has three classes, low, medium, and high.

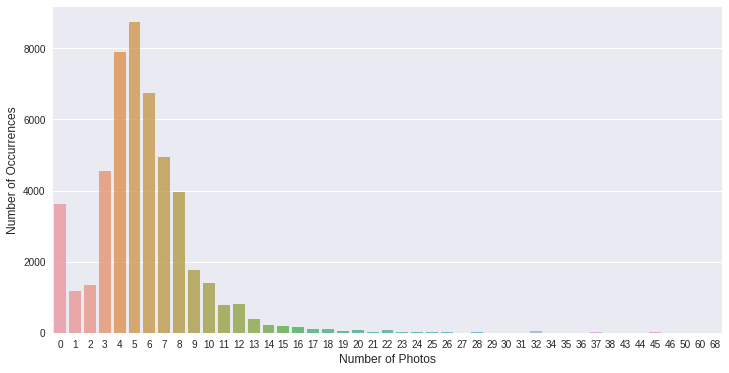


As we can see, the dataset is unbalanced, there are almost 35,000 instances with low interest level, but only lower than 5000 instances with high interest level.

Quantity distribution histograms of bathroom and bedrooms:

For photos feature, in our project, we decide to use the amount of photos as the new feature. We think it is consistent with common sense that the more photos the advertisement has, the more likely for user to click the link.



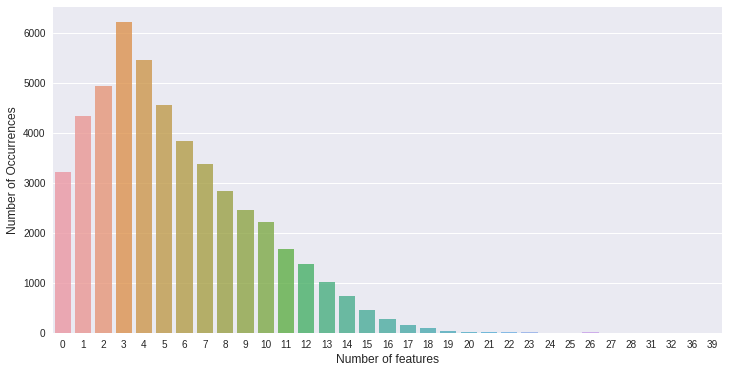
It is interesting to find out there are about 3,800 advertisements, which do not have any photos posted.

There is a feature in our data called “feature”, it is a set of key words which describes the apartment. For a json format example,

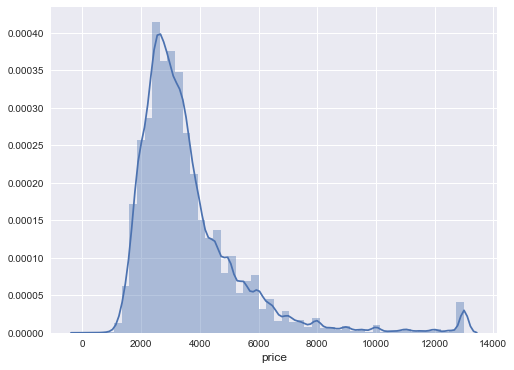
Feature: “[Doorman, Elevator, Fitness Center, Cats Allow]”.

As we can see, this tells us the apartment has doorman, elevator, fitness center in the building, and cats are allowed.

Although, there would be different key words of description, the degree of understanding of the house is determined by the features’ amount. Thus, similarly, we created the “number of features” as a new attribute.



All of us believe the price is the most important reason for users to decide weather to click the link or not.



This graph shows the distribution of the price. As the graph shown, most of the apartments have the price from $2,000 to $4,000.

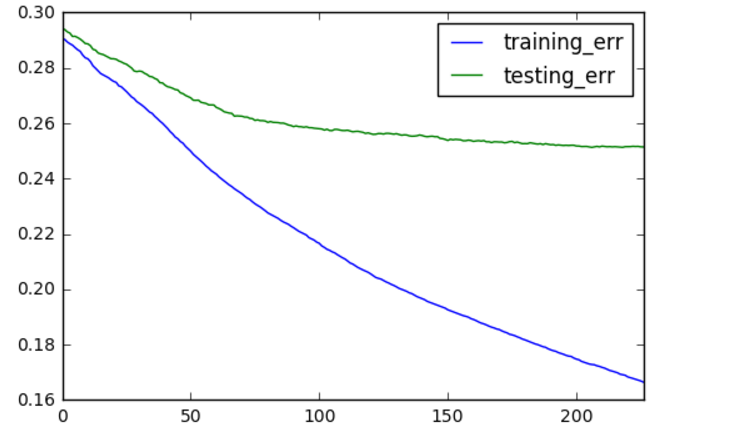
* 1. **Plan**

1. **Train and Test**

We split our data into two parts, 67% of data will be used for training the model and 33% will be used for testing. We trained Random Forest classifier, AdaBoost classifier, and XGBoost classifier respectively and then fit the testing data in each one to get prediction.

* 1. **Train**

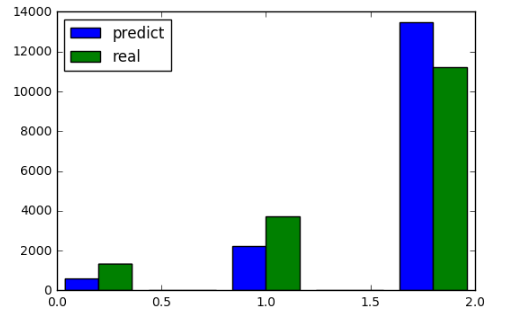
Take XGBoost for example, we need to figure out the best boosting rounds to avoid both underfitting and overfitting. In order to do that, we use cross validation method. We first split the dataset into 5 folds. In each round of boosting, we iteratively use 4 to train and use 1 to test, and get both the testing error rate and training error rate for each boosting round. As you can see from below graph, both testing error and training error are decreasing with the boosting rounds. The testing error stops improving at about 200 rounds. So we think 200 is a proper round.



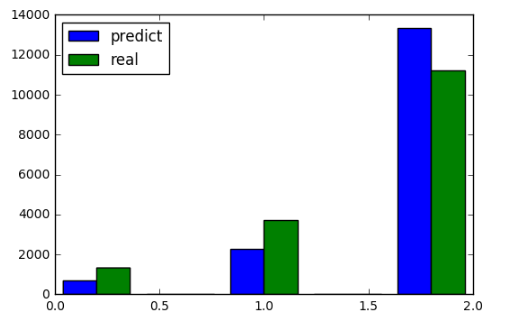
* 1. **Test**

Below graphs are the predictions from each classifier.

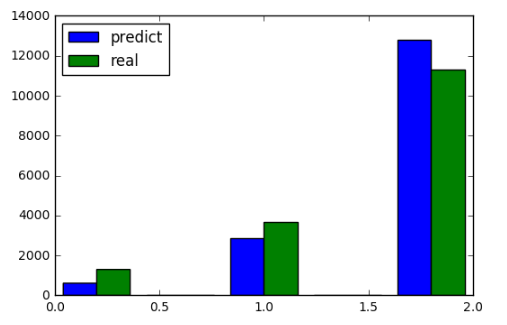
**Random Forest**



**AdaBoost**



**XGBoost**



1. **Evaluation**

To evaluate the performance of our models, we will use confusion matrix, Precision and Recall Curve and ROC.

We get the final evaluation score by comparing the prediction with actual result of testing data.

* 1. **Confusion Matrix and Classification Report**

**Random Forest**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **high** | **medium** | **low** |
| **high** | **319** | **479** | **528** |
| **medium** | **189** | **1099** | **2444** |
| **low** | **73** | **661** | **10495** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **high** | **0.24** | **0.55** | **0.33** | **581** |
| **medium** | **0.29** | **0.49** | **0.37** | **2239** |
| **low** | **0.93** | **0.78** | **0.85** | **13467** |
| **Avg/total** | **0.82** | **0.73** | **0.77** | **16287** |

**AdaBoost**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **high** | **medium** | **low** |
| **high** | **347** | **487** | **492** |
| **medium** | **228** | **1084** | **2420** |
| **low** | **102** | **718** | **10409** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **high** | **0.26** | **0.51** | **0.35** | **677** |
| **medium** | **0.29** | **0.47** | **0.36** | **2289** |
| **low** | **0.93** | **0.78** | **0.85** | **13321** |
| **Avg/total** | **0.81** | **0.73** | **0.76** | **16287** |

**XGBoost**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **high** | **medium** | **low** |
| **high** | **355** | **649** | **294** |
| **medium** | **196** | **1429** | **2073** |
| **low** | **58** | **808** | **10425** |

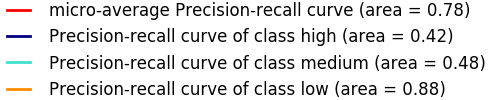
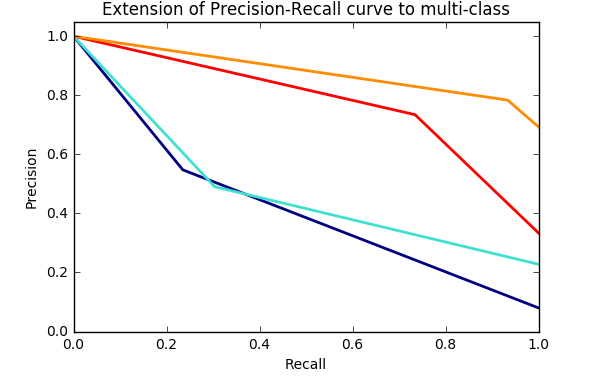
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **high** | **0.27** | **0.58** | **0.37** | **609** |
| **medium** | **0.39** | **0.50** | **0.43** | **2886** |
| **low** | **0.92** | **0.81** | **0.87** | **12792** |
| **Avg/total** | **0.80** | **0.75** | **0.77** | **16287** |

We can see from the Confusion Matrix that all 3 classifiers have difficulty in predicting the high and medium classes. The reason is due to the imbalanced data set. Classifiers get trained pretty well with low class because there are more instances belong to the low class.

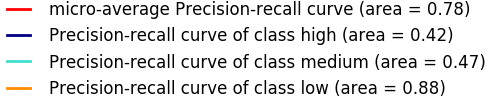
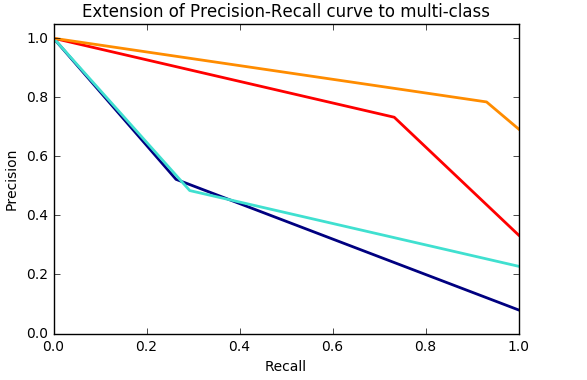
We should try to find solution to overcome the imbalanced data. SMOTE is a good technique for this scenario.

* 1. **Precision and Recall Curve**

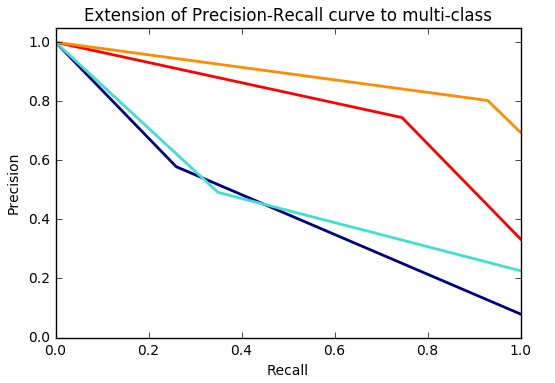
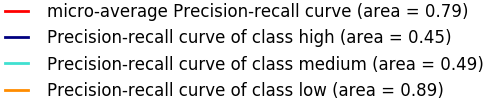
**Random Forest**



**AdaBoost**

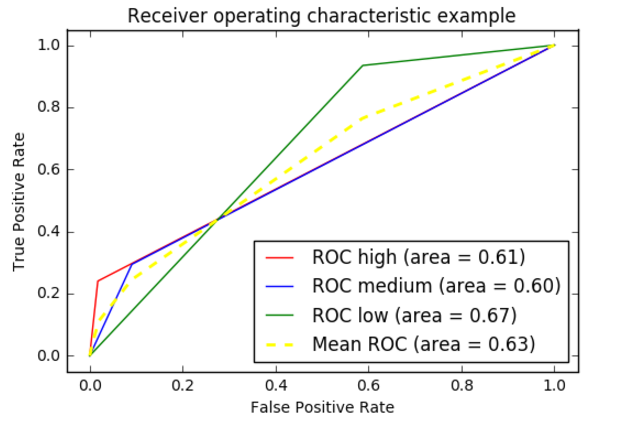


**XGBoost**

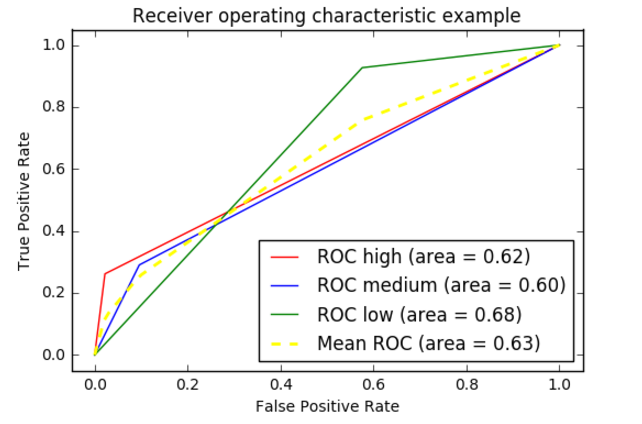
 

* 1. **ROC Curve**

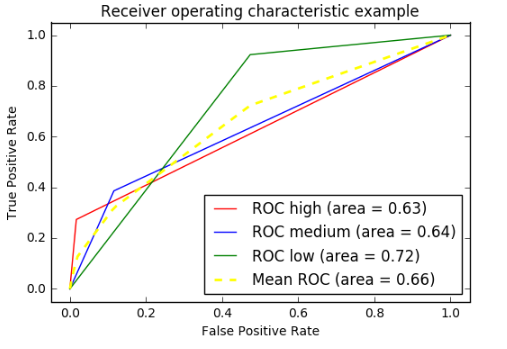
**Random Forest**



**AdaBoost**



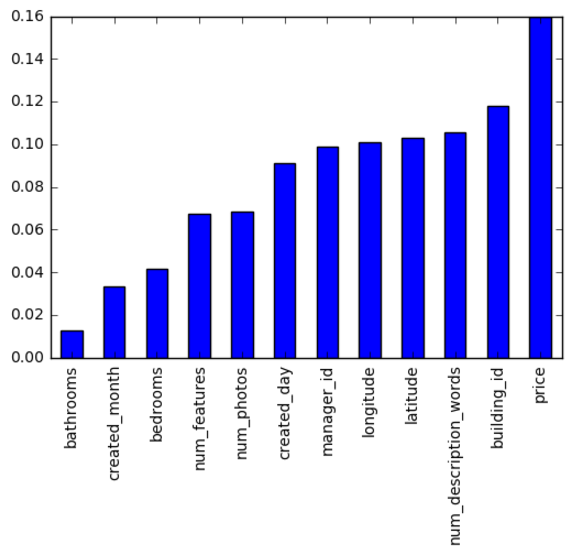
**XGBoost**



From the Precision-Recall curve and ROC curve we can conclude that the average predict precision of all 3 classifies is fine. They all perform well when predicting an instance that belongs to low class. They have some problems predicting high and medium instances. The XGBoost classifier has a higher precision than the other two.

1. **Summary**
   1. **Conclusion**

We get an internal view of the importance of each feature from Random Forest model. Price, building id, number of description words, and location are the more important than other features. The useful information is in order to make a rental housing popular, the agent manager should probably consider a reasonable price and describing more.



* 1. **Potential Improvement**

As you can see in above analysis, our model doesn’t perform well for instances that belong to high and medium classes. The imbalance dataset prevents our model from predicting precisely. We have two solutions for this problem. First, remove some instances that belong to low class. Second, create more instances that belong to high and medium classes. SMOTE is a proper technique here.

Besides, we can take advantage of image features to train our models. It is obvious that a clean and well decorated house will be more popular than an under decorated one, given other conditions the same. In the future, we should consider including some image processing in our model.