# Rental Listing Interest Level Prediction

**Lisha Han**
han.lis@husky.neu.edu
CSYE7245 , Spring 2017, Northeastern University

## 1. Abstract

This research was a competition published on kaggle.com. The main purpose of this report is to make prediction of rental information interest level. In this report I will train and evaluate three different commonly used models-- Sensor Vector Regression, Gradient Random Boosting and Random Forest. And I will select the best fit model for prediction.

## 2. Introduction

This research is based on the data RentHop provided. RentHop is a rental listing website. The prediction on interest level of each listing will help better business of rental listing website. The interest level prediction is mainly based on factors such as number of rooms, location, price and so on. The prediction target would be interest level high, medium and low.

The problem of classification is considered as learning a model that maps instances to class labels. There are lots of commonly used models in prediction. In this report I will use Sensor Vector Regression, Gradient Random Boosting and Random Forest for my research. I will cross-validate the classifiers and evaluate them.

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest)[6]. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default)[5].

Gradient boosting allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced[7][8][9].

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction[10].

# 3. Dataset

## 3.1 Dataset description

The data comes from renthop.com, an apartment listing website. These apartments are located in New York City. The target variable, interest_level, is defined by the number of inquiries a listing has in the duration that the listing was live on the site.

The JSON dataset is a structured database that contains the listing information as the number of bathrooms and bedrooms, building_id, created, description, display_address, features, latitude, listing_id, longitude, manager_id, photos links, price, street_address, and interest_level[1].

## 3.2 Data cleaning

This dataset contains 15 columns and 48535 rows. First step is to check all the outliers in dataset. Fig 3.1 shows distributions of bedrooms, bathrooms and rent price.

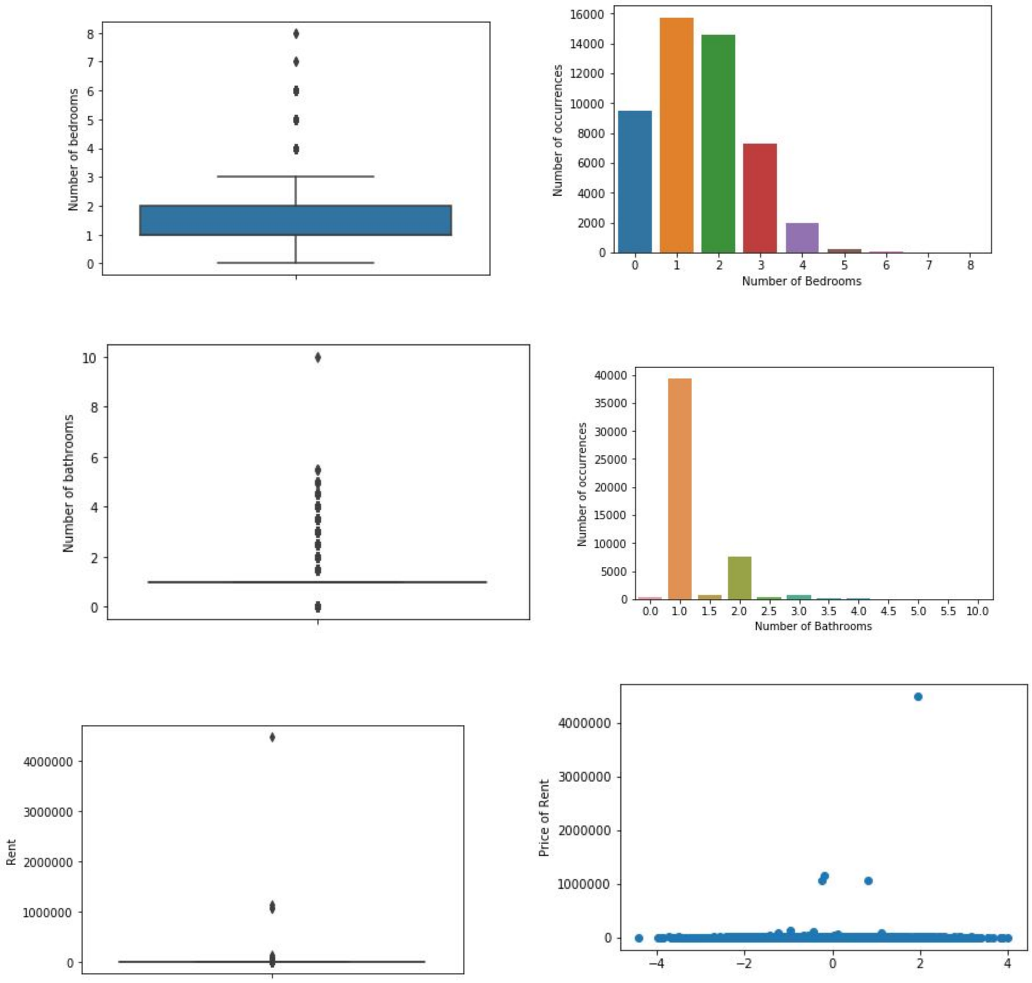
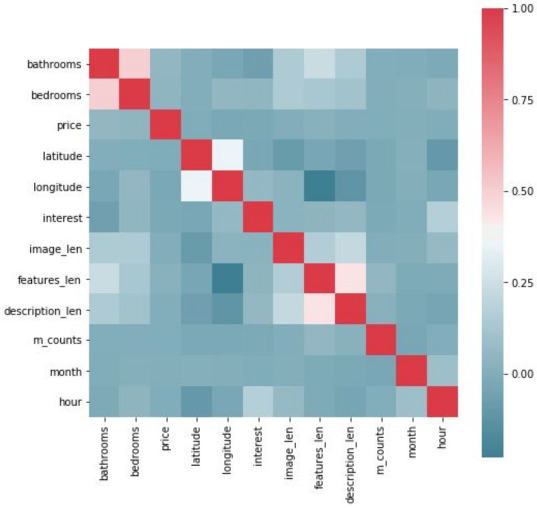


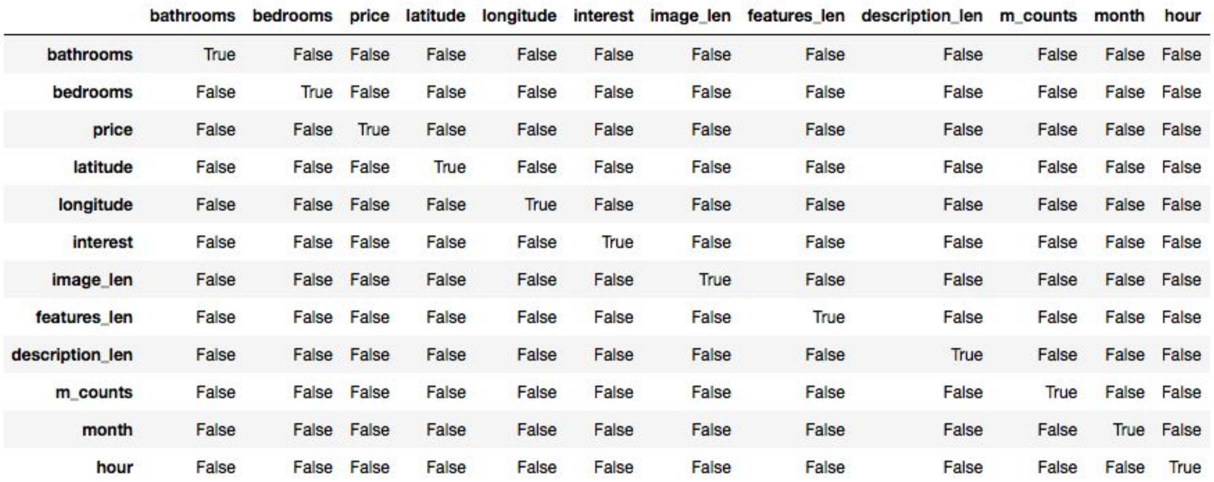
Fig 3.1

I imputed data which have over 3 bathrooms, over 4 bedrooms, rent price over 10k, location data between quartile of 5%~99.5%. Then I categorized interest level with numbers from 0 to 2. And I deleted similar location information, only kept coordinates of locations. My main purpose is to keep most data impute extreme outlier.

Photos, features and description are lists of words. So I created a few columns to count the length of these features. I calculated the interest level by manager ID and created a column to store interest level by manager.

At last, I checked the correlation between features of dataset.

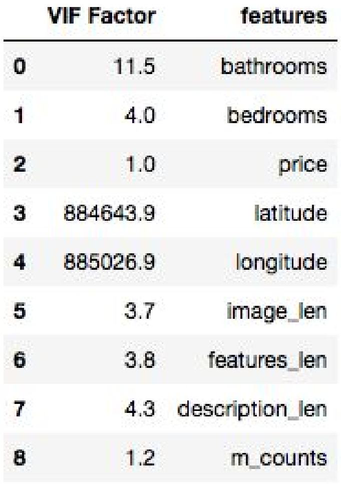




| | bathrooms | bedrooms | price | latitude | longitude | interest | image_len | features_len | description_len | m_counts | month | hour |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **bathrooms** | True | False | False | False | False | False | False | False | False | False | False | False |
| **bedrooms** | False | True | False | False | False | False | False | False | False | False | False | False |
| **price** | False | False | True | False | False | False | False | False | False | False | False | False |
| **latitude** | False | False | False | True | False | False | False | False | False | False | False | False |
| **longitude** | False | False | False | False | True | False | False | False | False | False | False | False |
| **interest** | False | False | False | False | False | True | False | False | False | False | False | False |
| **image_len** | False | False | False | False | False | False | True | False | False | False | False | False |
| **features_len** | False | False | False | False | False | False | False | True | False | False | False | False |
| **description_len** | False | False | False | False | False | False | False | False | True | False | False | False |
| **m_counts** | False | False | False | False | False | False | False | False | False | True | False | False |
| **month** | False | False | False | False | False | False | False | False | False | False | True | False |
| **hour** | False | False | False | False | False | False | False | False | False | False | False | True |

Fig 3.2

I calculated vif of dataset. There is no obvious dependency between features. So there is no need to dispute any feature.



|   | VIF Factor | features |
|---|---|---|
| **0** | 11.5 | bathrooms |
| **1** | 4.0 | bedrooms |
| **2** | 1.0 | price |
| **3** | 884643.9 | latitude |
| **4** | 885026.9 | longitude |
| **5** | 3.7 | image_len |
| **6** | 3.8 | features_len |
| **7** | 4.3 | description_len |
| **8** | 1.2 | m_counts |

Fig 3.3

### 3.3 Summary of data exploration

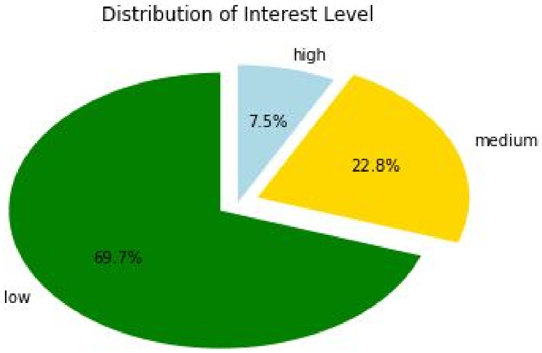


Fig 3.4

The distribution of 3 levels are low-69.7% medium-22.8% high-7.5%. Most of the rental information is not attractive to users.

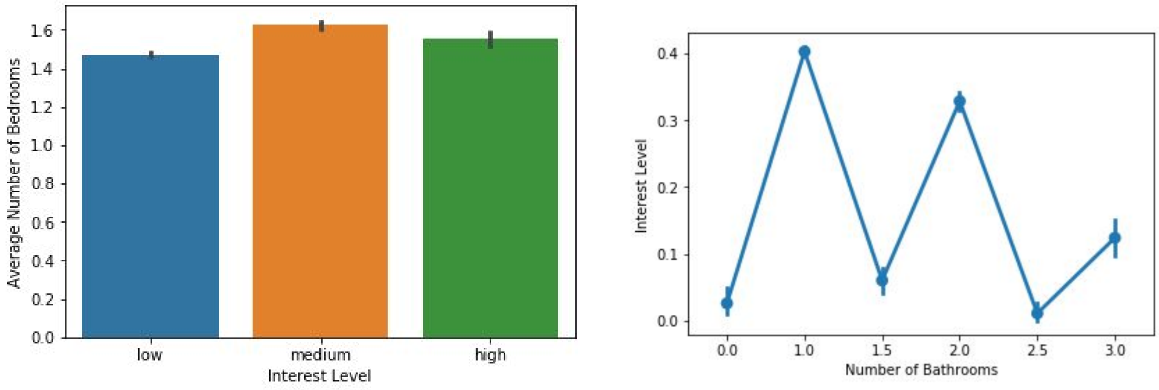


Fig 3.5

Interest level over bathrooms is not very obvious. 1.5 bathrooms is most common. Apartments with 1-2 bedrooms get most interest level.

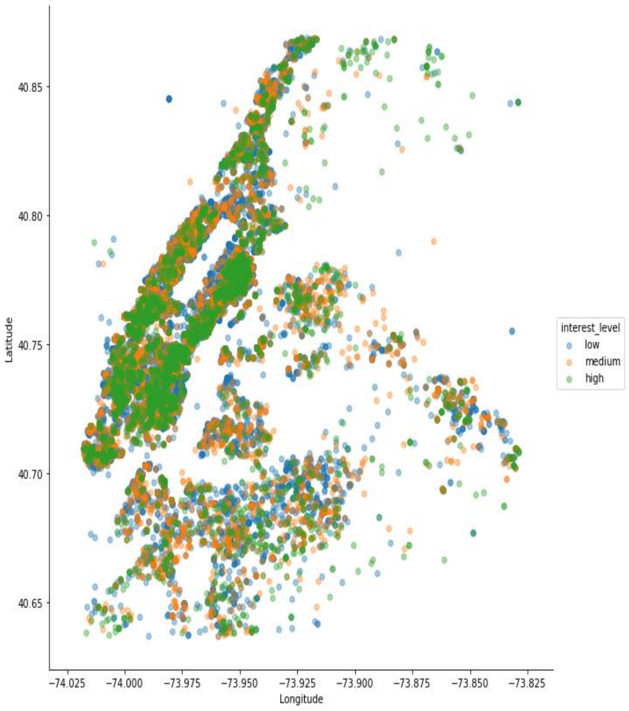


Fig 3.6

Location may not indicate a clear relation between interest level. But we can tell from the plot that most interest clusters in Manhattan. There is no obvious correlation between these important features.

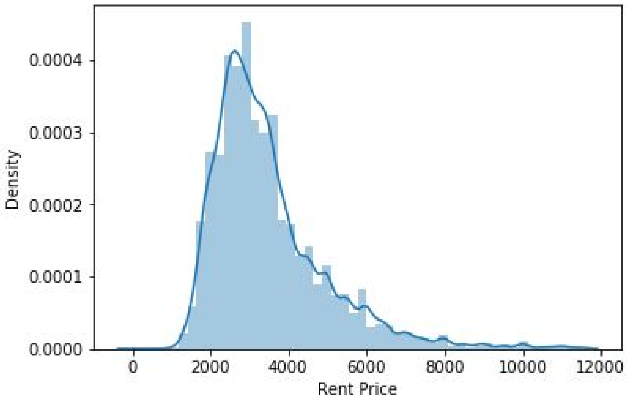


Fig 3.7

Rent price between 2000 and 4000 got more higher interest level. The more photos the higher interest level.

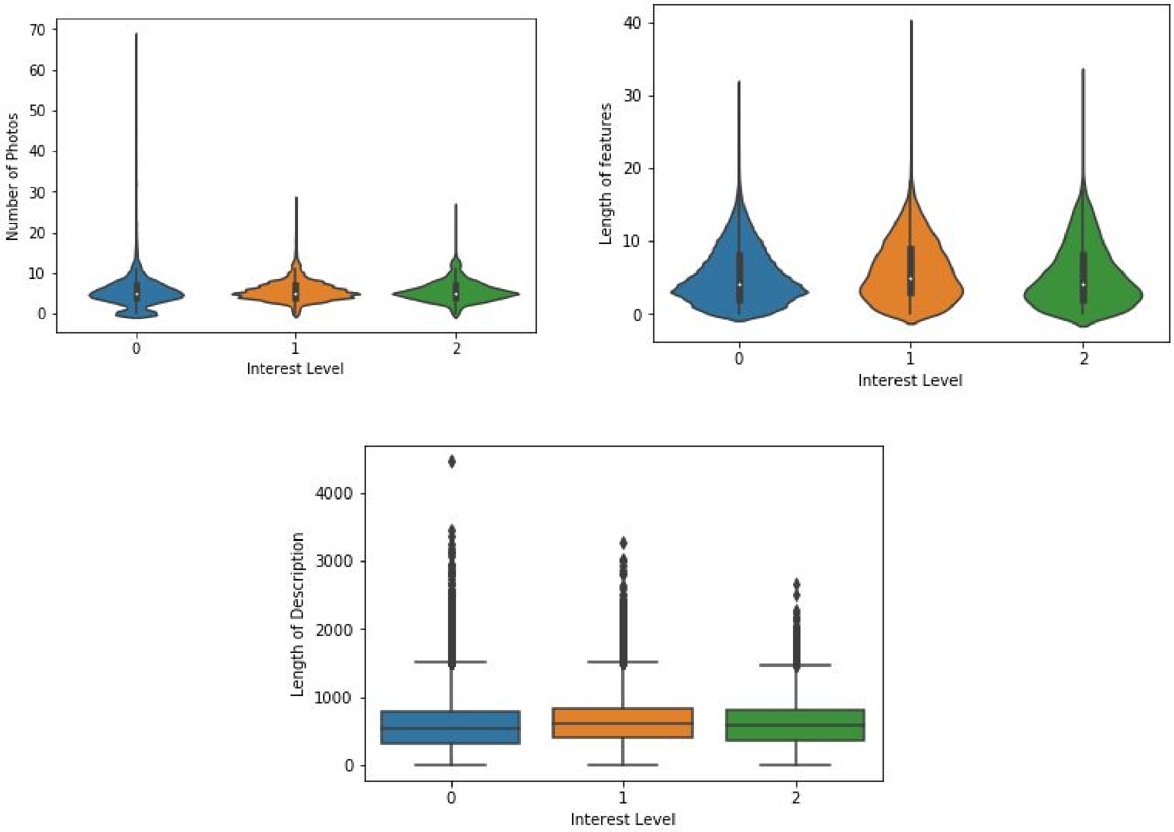


Fig 3.8

Photo count around 5 gets more attention. The more feature description the higher interest level. Features count around 3 gets more attention. Length of description under 1000 gets the most interest.

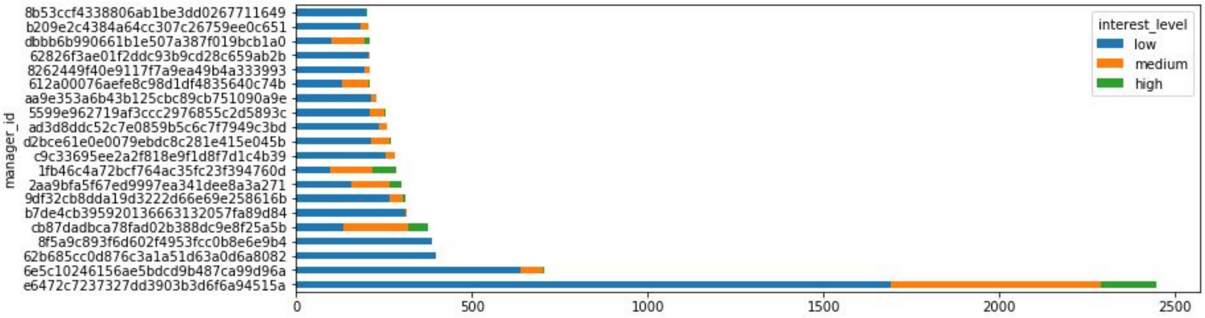


Fig 3.9

The most popular rental manager could get 2447 in total of interest level. Rental manager is important and related to our target.

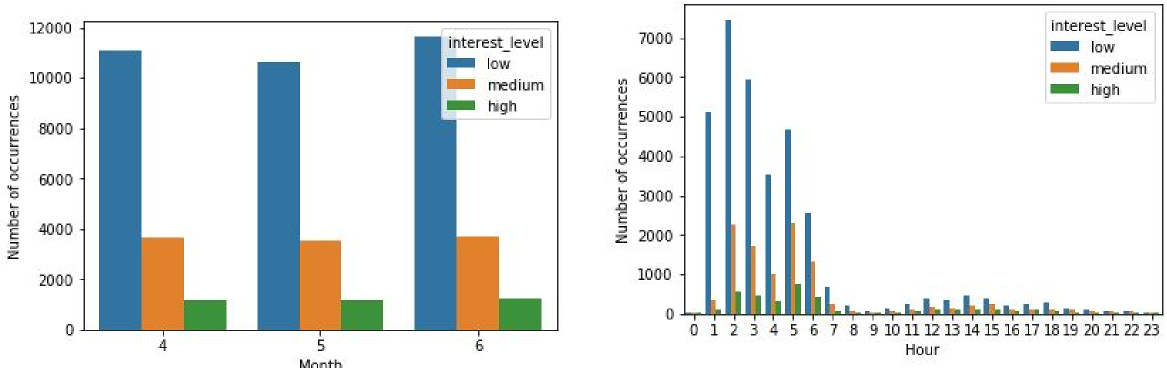
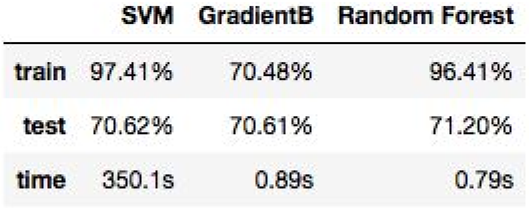


Fig 3.10

April, May and June are the most popular time of this website. The main rental hunting clusters around one to seven in a day. Most important features in this research are bathrooms, bedrooms, price, location, number of images, number of features, length of description and rental managers.

## 4. Results

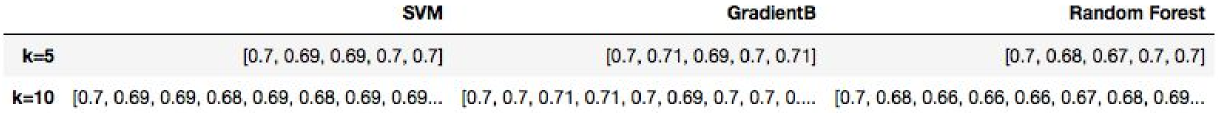
Fig 4.1 shows the training and testing result of three different models. Support vector machine have the best training accuracy. But it consumes lots of time to finish training with 30k rows of data. Random forest model have less train accuracy. But the test accuracy is the highest. And It only took less than a second to finish the training.



| | SVM | GradientB | Random Forest |
|---|---|---|---|
| **train** | 97.41% | 70.48% | 96.41% |
| **test** | 70.62% | 70.61% | 71.20% |
| **time** | 350.1s | 0.89s | 0.79s |

Fig 4.1

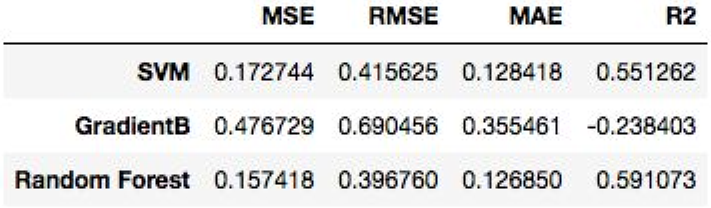
Fig 4.2 shows validation result. I used k-fold cross-validation. I set k to 5 and 10 to see if there is any overfitting and underfitting. We can tell from the result that there is no overfitting or underfitting.



| | SVM | GradientB | Random Forest |
|---|---|---|---|
| **k=5** | [0.7, 0.69, 0.69, 0.7, 0.7] | [0.7, 0.71, 0.69, 0.7, 0.71] | [0.7, 0.68, 0.67, 0.7, 0.7] |
| **k=10** | [0.7, 0.69, 0.69, 0.68, 0.69, 0.68, 0.69, 0.69... | [0.7, 0.7, 0.71, 0.71, 0.7, 0.69, 0.7, 0.7, 0.... | [0.7, 0.68, 0.66, 0.66, 0.66, 0.67, 0.68, 0.69... |

Fig 4.2

Fig 4.3 is evaluation of three models.



| | MSE | RMSE | MAE | R2 |
|---|---|---|---|---|
| **SVM** | 0.172744 | 0.415625 | 0.128418 | 0.551262 |
| **GradientB** | 0.476729 | 0.690456 | 0.355461 | -0.238403 |
| **Random Forest** | 0.157418 | 0.396760 | 0.126850 | 0.591073 |

Fig 4.3

## 5. Discussion

Among the models tested, Random Forest and Support Vector Machine have similar performance. They both have r-square around 0.5 and low mean-square error around 0.15. Their mean-average errors and root-mean-square deviation have no big difference. But the training time Random Forest took is only 0.2% of SVM. Random forest model is the best in this research. For future investigation, more features may be used in training. I can include date as a factor. To get a good random forest classifier. The training data is very important. I may use methods of GridSearch or Bayesian optimization to find best combination of training data. Development of algorithms is another option. I could use enhanced random forest tree to improve accuracy. The test accuracy is around 70%. Improvement of it is promising.

## 6. References:

[1] *Rent Interest Classifier, Elisabeth Kim, April. 2017.*

[2] *Data Exploration Two Sigma Renthop, Sergei Neviadomski, Sep.2017*

[3] *Predictive Modeling of NBA Player Value Using AdaBoost and Stochastic Gradient Descent for Applications to Fantasy Basketball Betting, Thomas Kalnik, 2017.*

[4] *Google Stock Prediction and Time Series Analysis, Aswathnarayan Muthukrishnan Kirubakaran, Meenakshi Muthiah, 2017*

[5]*http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*

[6] *L. Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.*

[7] *J. Friedman, Stochastic Gradient Boosting, 1999*

[8] *J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, Vol. 29, No. 5, 2001.*

[9] *T. Hastie, R. Tibshirani and J. Friedman. Elements of Statistical Learning Ed. 2, Springer, 2009.*

[10] *http://scikit-learn.org/stable/modules/svm.html*

**Code of this research is available in my github.**

https://github.com/lishahan/CSYE7245_2018Spring