CS412
INTRODUCTION TO MACHINE LEARNING

Project Report
RENTAL LISTING ENQUIRIES

Team Members (UIN):

Akash Dhirajlal Dobaria (663179817)
Akhauri Prateek Shekhar (653062844)
Anamika Paul (667212344)
Khushbu Durge (661389838)
Swapnil Sagar (677194294)
Urviben Patel (653504836)

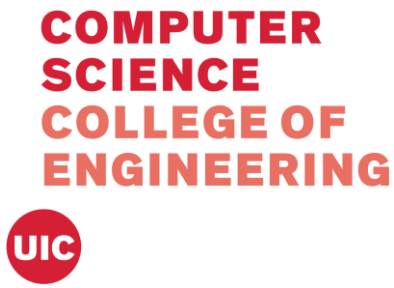


TABLE OF CONTENT

INTRODUCTION

Searching for the perfect home can be tiring; gathering all the details and browsing through endless listings. Structuring and making sense of all the available real estate data programmatically is difficult. To regularize and ease the same, the project will predict how popular an apartment rental listing is based on the listing content like text description, photos of the apartments, number of bedrooms, price, etc. The popularity is quantified as interest level of each listing and can assume values as High, Medium and Low.

DATA

Training data for the problem is provided at Kaggle. The train data consists of 50,000 rental listings (approx.) with attributes such as:

- **bathrooms**: number of bathrooms
- **bedrooms**: number of bathrooms
- **building_id**
- **created:** date when listing was created
- **description**
- **display_address**
- **features**: a list of features about this apartment
- **latitude**
- **listing_id**
- **longitude**
- **manager_id**
- **photos**: a list of photo links.
- **price**: in USD
- **street_address**
- **interest_level**: this is the target variable. It has 3 categories: 'high', 'medium', 'low'

The 3 target classes: [High, Medium and Low] were represented in codes as [0,1,2]. The dataset was highly skewed as it contained 70% listings of class low, 23% listings of class medium and 7% listings of class high.

EVALUATION CRITERIA

Several measures were taken to evaluate the results obtained from different classifiers. We tried various classifiers: MultinomialNB, SVM Linear kernel, SVM Polynomial kernel, AdaBoost, Random Forests and Logistic Regression.

We used 3 different types of evaluation criteria:

1. **Precision, Recall, F-Score and Support**: The input data was split into Train Data (75% of the whole dataset for each candidate) and Test Data (25% of the whole dataset of each candidate) and then precision, recall and f-scores were computed for high, medium and low class labels.
   The precision, recall and f-score were computed as per the following formula:
   - $Precision = \frac{True\ positive}{True\ positive + False\ positive}$
   - $Recall = \frac{True\ positive}{True\ positive + False\ negative}$
   - $F\text{-}Score = \frac{2 * precision * recall}{Precision + Recall}$

2. **10-Fold Cross Validation:** The input data was split into 10 equal sized samples and the classifiers were run 10 times, each time considering a different sample as test data and the remaining 9 samples as training data. Using this approach, 10 different 'f1' scores were obtained from which mean, maximum and minimum values were computed.

3. **Out of Bag Error:** Used to evaluate Random Forest and Ada-Boost classifiers.

APPROACH

Our task here is to build a classifier model which can predict the popularity of an apartment by the interest level, i.e. classify them as High, Medium and Low. To accomplish this task, we tried several different classifiers, ensembles with data preprocessing and finally evaluated our classifier.

1. Data Pre-Processing:

The data in the training set was extracted into a data frame using Python's Pandas library for better visualization and understanding. The training data contained a set of 16 features. These features contained categorical features (interest level, display address, etc.), quantitative features (latitude,

longitude, price, number of bathrooms and bedrooms, etc.) and text based features such as description and a set of commas separated attributes for each listing called features.
Following tasks were performed in data pre-processing step:

a. Input feature space was split into 2 feature sets: Text based Features, Numerical Features.
b. URLs, HTML tags were removed from the text based features.
c. Stop words, delimiters were removed from the text based features.
d. Extra features like number of photos were added to each listing.

2. Using Classifiers:

We used Python's Sklearn library for various classifiers and Pandas Dataframe library and NLTK library for various data-processing operations.

After Data-Preprocessing the cleaned Data was split in the ratio of 75% and 25% into Train Data and Test Data. We then experimented with different classifiers such as MultinomialNB, SVM Linear kernel, SVM Polynomial kernel, AdaBoost, Random Forests and Logistic Regression. We also tried different ensemble methods for different set of features and combined their result. The classifiers which gave us the best results were AdaBoost and Random Forest for both text based and quantitative features.

3. Evaluating our classifier

We tried several approaches to modify the data we had to get the best classifier. Following are the methods used and their evaluation:

a. Using Quantitative Features: Few classifiers like K-Nearest Neighbour, Multinoulli Naive Bayes, SVM and Logistic Regression were trained only on set of quantitative features while ignoring the categorical features and text based features. The precision, recall and F-score are as shown below for the respective classifiers:

i. **K Nearest Neighbour**:

	precision	recall	f1-score	support
0	0.29	0.23	0.26	780
1	0.75	0.88	0.81	6966
2	0.33	0.18	0.23	2168
avg / total	0.62	0.67	0.64	9914

Output of KNN classifier

ii. Multinoulli Naive Bayes:

	precision	recall	f1-score	support
0	0.12	0.50	0.19	1005
1	0.74	0.42	0.53	8527
2	0.23	0.26	0.24	2806
avg / total	0.58	0.39	0.44	12338

Output of Multinoulli Naive Bayes

iii. Logistic Regression:

	precision	recall	f1-score	support
0	0.33	0.00	0.01	316
1	0.73	0.97	0.83	3209
2	0.42	0.11	0.18	1009
avg / total	0.63	0.71	0.63	4534

Output of Logistic Regression

iv. Support Vector Machines:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	960
1	0.76	0.45	0.56	8571
2	0.26	0.66	0.37	2807
avg / total	0.59	0.46	0.48	12338

Output of SVM with Linear Kernel

b. Using Extended set of Features: there is a column named “features” in our dataset which contained nominal attributes, for example: fitness center, elevator, watchman, pets allowed, etc. for each listing. These set of nominal attributes were converted into binary attributes where each attribute was added as a column in the data frame. If the listing has that attribute then the corresponding column has value 1 and 0 otherwise. After this extension, our total feature space was of size 300. Also, while extending the features, it occurred that there were many redundant features like “pets allowed”, “cats/dogs allowed”, “cats allowed”, “dogs allowed”, etc. These kinds of features were removed and merged into one single column and then the classifiers were trained and tested on the dataset.

	bathrooms	bedrooms	no_of_photos	price	dish_washer	fitness_center	hardwood_floors	dining_room	elevator	doorman	...	laundry	p
0	1.5	3	5	3000	0	0	0	0	0	0	...	0	0
1	1.0	2	11	5465	0	1	0	0	1	1	...	0	0
2	1.0	1	8	2850	1	0	1	0	0	0	...	0	0
3	1.0	1	3	3275	0	0	1	0	0	0	...	0	0
4	1.0	4	3	3350	0	0	0	0	0	0	...	0	1
5	2.0	4	5	7995	0	0	0	0	0	0	...	0	0
6	1.0	2	10	3600	0	0	0	0	1	0	...	0	1
7	2.0	1	5	5645	1	0	1	0	1	1	...	1	1
8	1.0	1	5	1725	0	0	0	0	1	0	...	0	0
9	2.0	4	9	5800	1	0	1	0	0	0	...	0	0
10	1.0	0	1	1950	0	0	0	0	0	0	...	0	0
11	1.0	1	5	1675	1	0	0	0	0	0	...	0	1
12	1.0	2	4	3000	0	0	0	0	1	1	...	0	0
13	2.0	2	6	6895	0	1	0	0	0	1	...	0	0
14	1.0	1	6	3050	1	0	1	0	1	0	...	0	0
15	1.0	0	2	2350	0	1	0	0	1	1	...	0	0

Sample Data Frame with extended features

We tried 2 different classifiers on this dataframe with 300 features. Following are the results obtained from them:

i. Logistic Regression:

	precision	recall	f1-score	support
0	1.00	0.77	0.87	1270
1	0.97	0.99	0.98	12886
2	0.98	0.96	0.97	3981
avg / total	0.97	0.97	0.97	18137

Output of Logistic Regression

ii. Random Forests:

	precision	recall	f1-score	support
0	1.00	0.01	0.01	794
1	0.91	1.00	0.95	8054
2	1.00	1.00	1.00	2488
avg / total	0.94	0.93	0.90	11336

Output of Random Forests

The extremely high values of precision, recall and f-scores clearly indicates *Over-fitting*. The model did not perform well when we split our data in the ratio of 70% (train), 15% (validation) and 15% (test).

Moreover, it was difficult to combine attributes like “one small dog allowed”, “pets allowed”, “dogs/cats allowed”, etc. into one single attribute. We had to write a number of custom rules in order to join these attributes.

c. Segregation of features into multiple feature sets:

With a view to enhance the output of the classifier, the categorical features (description, street address, display address, etc.) were converted into quantitative values. These were then joined with numerical set of features (no of bedrooms, no of bathrooms, price, etc.).

Next, the features were segregated into 2 different sets:

i. Numerical feature set

ii. Text based feature set.

We then used Random Forest and AdaBoost ensemble methods on each of the feature sets to get the precision, recall and f-1 score measures. The result was then derived by taking the weighted mean or the mode of precision, recall and f1-score obtained from both classifiers for different feature sets.

The results of these are as shown below.

i. Random Forests:

Quantitative features

Numerical features consisted of 'no of bedrooms', 'no of bathrooms', 'categorized display address', 'categorized latitude', 'categorized longitude, 'categorized manager id', 'no of photos' and 'price'

	precision	recall	f1-score	support
0	0.64	0.03	0.05	960
1	0.71	0.99	0.83	8571
2	0.54	0.08	0.14	2807
avg / total	0.67	0.71	0.61	12338

Output of Random Forests Quantitative Features

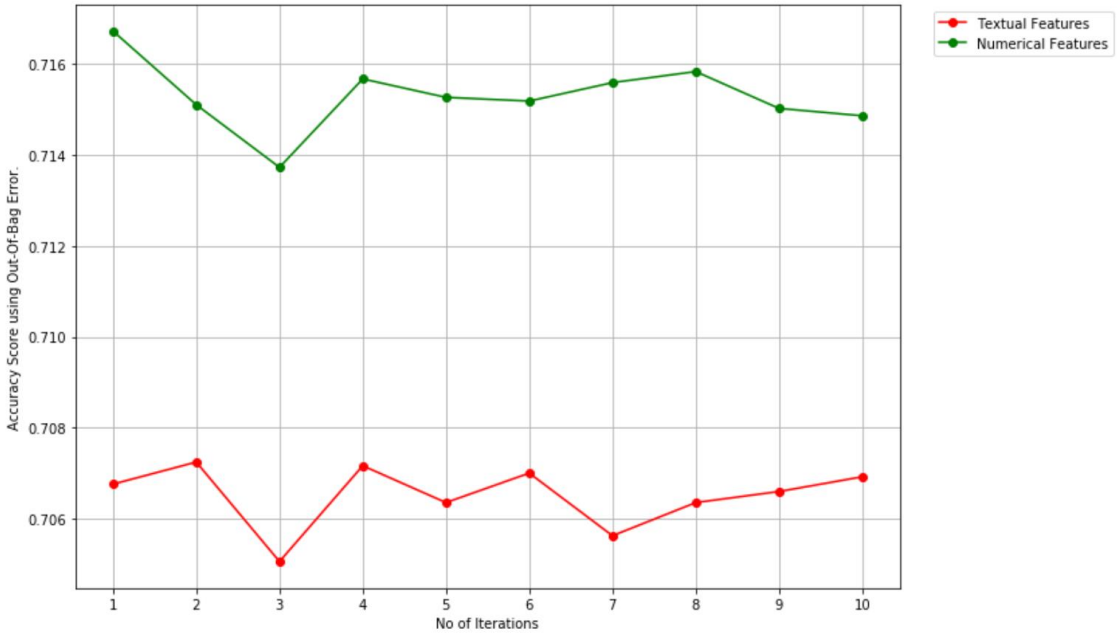
Text features

Text based features consisted of 'description' joined with 'features' column

	precision	recall	f1-score	support
0	0.64	0.03	0.05	960
1	0.71	0.99	0.83	8571
2	0.54	0.08	0.14	2807
avg / total	0.67	0.71	0.61	12338

Output of Random Forests Text based Features

Comparison of Random Forests output - Textual features and Quantitative features:



ii. AdaBoost:

Quantitative Features

Numerical features consisted of 'no of bedrooms', 'no of bathrooms', 'categorized display address', 'categorized latitude', 'categorized longitude, 'categorized manager id', 'no of photos' and 'price'

	precision	recall	f1-score	support
0	0.44	0.30	0.36	960
1	0.80	0.83	0.81	8571
2	0.38	0.38	0.38	2807
avg / total	0.68	0.69	0.68	12338

Output of AdaBoost Quantitative Features

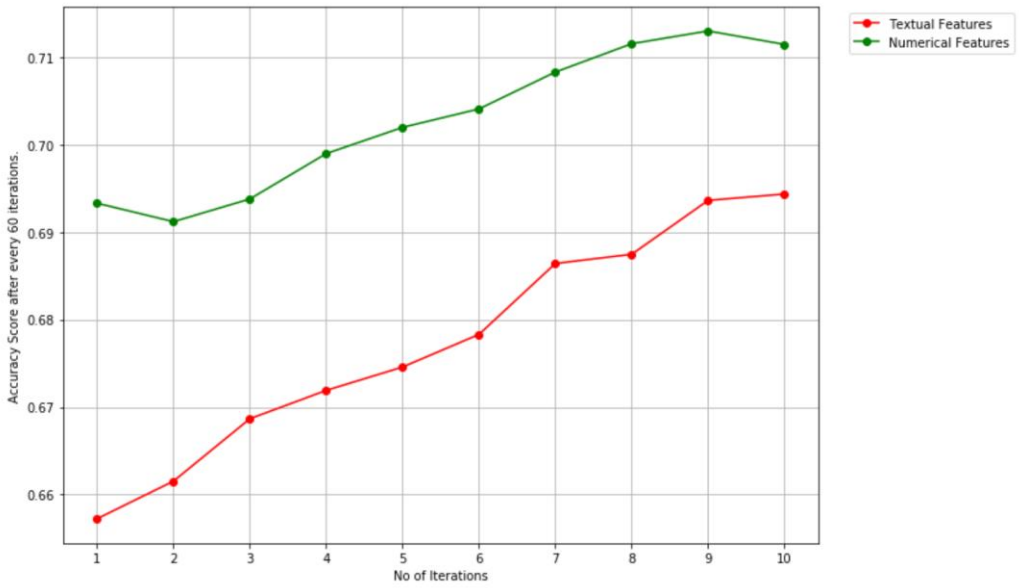
Text Features

Text based features consisted of 'description' joined with 'features' column

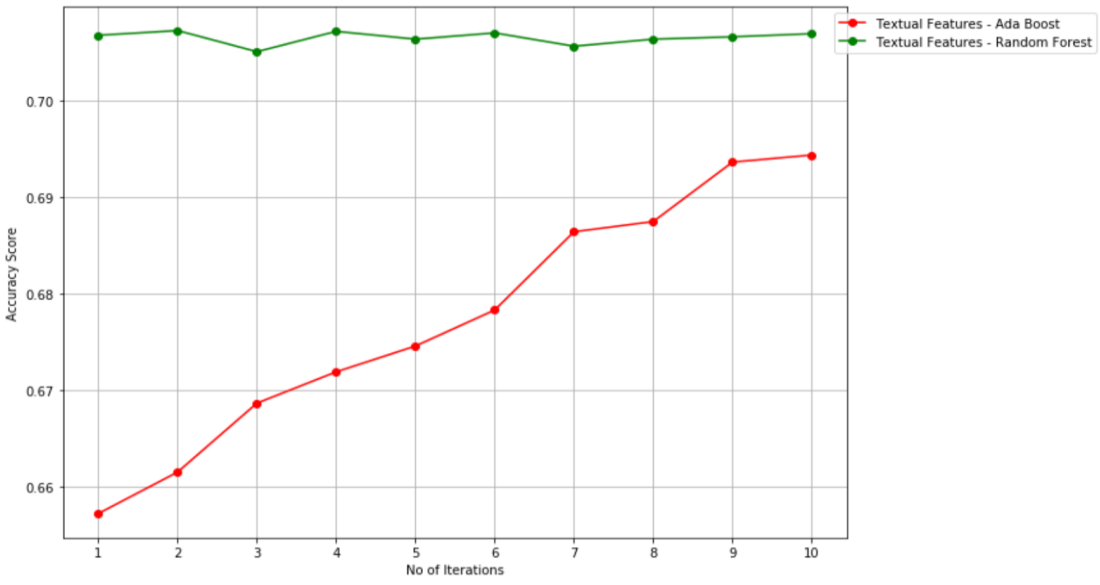
	precision	recall	f1-score	support
0	0.32	0.17	0.22	960
1	0.77	0.81	0.79	8571
2	0.35	0.35	0.35	2807
avg / total	0.64	0.66	0.65	12338

Output of AdaBoost Text based Features

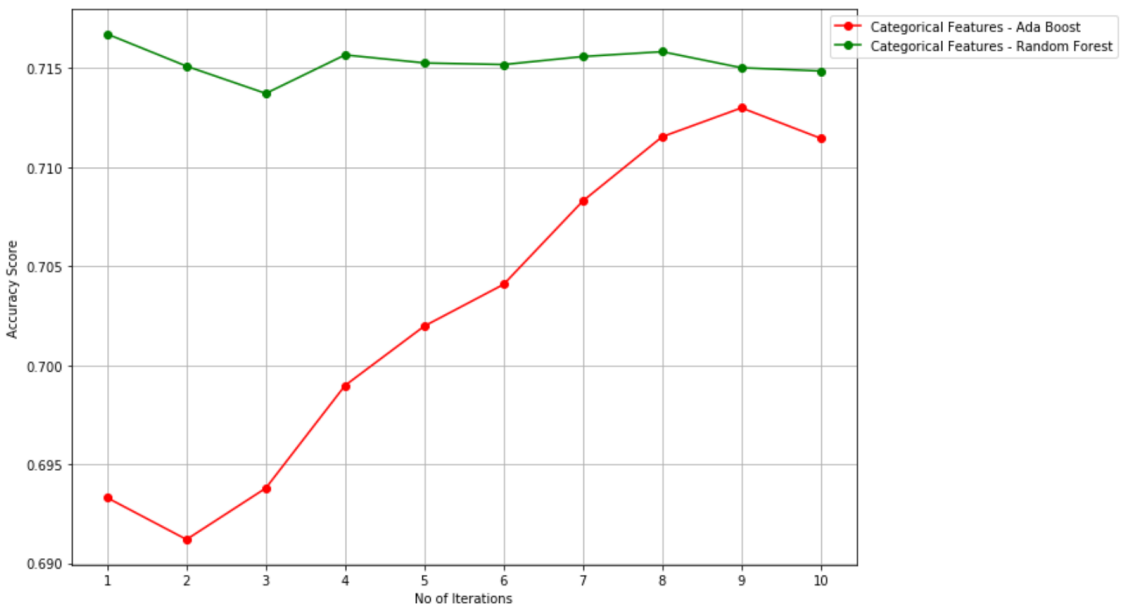
Comparison of AdaBoost output - Textual features and Quantitative features:



Comparison of AdaBoost vs Random Forest output - Textual Features:



Comparison of AdaBoost vs Random Forest output - Categorical Features:



CONCLUSIONS AND LEARNINGS:

- Data distribution: Training data should be sampled properly; it should not be skewed. If the data is skewed, then it becomes difficult to get good measures of accuracy, precision, recall and F-1 scores of individual target classes.

- No perfect recipe: There is no perfect classifier. Every classifier needs to be trained and tuned separately for different data. For example, while using text based features we had to pre-process the data, convert it to matrix containing TF-IDF values and then feed it into our classifier. On the other hand, for numerical features, we directly applied the data to our classifier. Eventually, we got different measures of accuracy and score for both text and numerical based features.

- Overfitting: Many features in a dataset caused the model to bias and thus overfitting. Dimensionality reduction did not solve the problem of overfitting for this case. For example, when we tried to add new features by converting nominal attributes to binary values our classifier gave us result of over 90% in precision, recall and F-1 score for the split of data into 75% (train) and 25% (test) set but the same classifier performed poorly when the same data set was split into 70% (train), 15% (validation) and 15% (test) set giving us a value in the range of 60% to 70% for the value of precision, recall and F-1 score.

- Data pre-processing to clean noisy data: Data should be as clean as possible before we start processing it. It is efficient to convert all categorical data to quantitative values to better analyze and classify. For example: The columns “Latitude” and “Longitude” were converted to numerical codes, photos feature was processed and a separate numerical column containing the number of photos were added to our dataset. This clearly increased the performance of our classifier. Earlier the f1-score was in the range of 45% to 65% for different classifiers but after addition of new features like number of photos, latitude and longitude converted to numerical codes, our classifier’s performance increased and we got F1-score in the range of 70%.

REFERENCES

1. scikit-learn: http://scikit-learn.org/stable
2. Python Regular Expressions: https://docs.python.org/2/library/re.html
3. TF-IDF Vectorizer: scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
4. SGD Classifier: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
5. Multinomial Naïve Bayes: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
6. SVM Linear/Polynomial SVC: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
7. AdaBoost: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
8. Random Forest: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
9. Feature Selection: http://scikit-learn.org/stable/modules/feature_selection.html
10. NLTK Stopwords: http://www.nltk.org/book/ch02.html
11. Pandas: http://pandas.pydata.org/