readme.md

ESE 305 FinalProject

Braden Fineberg, Brooke Behrbaum, Brianna Gallo

https://github.com/bfine9618/ese305FinalProject

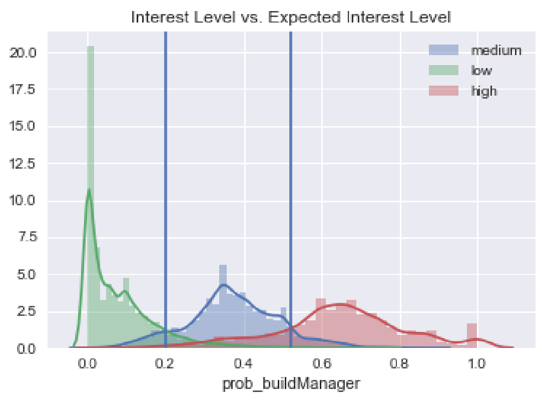
The initial dataset can be found in ./rawData. I have also used the script 'clean data' to standardize columns, rows and names to use for the rest of the analysis. In the final model, there is a function cleanPreprocessData() that does this to all train and test data. Finally, using the NLP script, I was successful in determining the type of rental from 75% of the data. After running a basic OLS regression with typed data, we are able to predict with higher accuracy what the interest level in each home is.

Cleaning Data

We started by creating the "Cleaning Data". It takes in the raw data, drops unnecessary columns, cleans the description column, and outputs a cleaned data file. This allows all of our models to work off of the same data and keep our results consistent.

In a final model, our cleaning function converted an HTML string into a simpel string and ran a basic NLP. In addition, our preprocessing involved generating a range of interaction terms and grouping the data in various ways to determine expectations for each unit. We believe that the interest level in a home can be turned into a simple equation: Net_Interest_Level = Reality-Expectation

The reality consits of the actual characteristics of the unit while expectation is for a unit's given asking price, what features do you expect. The reality of the unit also includes where it is located. Specifically, if a unit is located in a high interest building, the unit is likely high interest. The expectation created a clear separation in the data.



Final Model

Due to the lack of information on the test data, we are unable to effectivly predict "prob_buildInterest" for all of the test data. In this particular test set, 51% of the test data is able to be predicted using this method; therefore, we broke the test data into two sets: high quality data and low quality data. The high quality data is run through a multi-class logrithmic predictor, generating ~92% prediction accuracy. The remaining 49% of data is classified using an SVM with ~71% accuracy. Therefore, our expected accuracy is .91*.51+.71*.49=0.812 or ~81%. This blended strategy gave us the best outcome.

Conclusion

LOCATION, LOCATION, LOCATION. This classification problem is not one of features or price. Ultimately it comes down to which areas of a city are "hot". Which neighborhoods are sellers' markets. Therefore, we believe that the best real-estate classifiers will not classify individual units, but instead classify building or city blocks. This will allow market modelers to more accuratly invest in real estate and renters to understand the value of thier home. A byproduct of this approach, allows extremely high accuracy of logistc prediction, enabling sellers to determing the interest level of their specific unit.

Future Plans

In future models, there may be a way to predict "prob_buildManager" by running a KNN on the lat long data. If we can do this with high accuracy we can improve our overall model.

Initial Exploration

Basic OLS

Unfortunately, the basic OLS resulting in $R^2=.07$. This told us that we needed to in some way improve our data to draw any kind of meaningful conclusions. We had several theses such as rental type (loft vs townhome vs penthouse vs ect) had different features that would make it desirable. We also thought that in each vertical, there was a price/luxury ratio. Essentially, for every additional feature, the market would be willing to pay more. Finally, using the average price/luxury in each home, we are hoping to create a price ratio, comparing the list price to the expected price given our ratio. If this price ratio is above 1, then the house is listed too high; if the price ratio is less than 1, the rental is a good deal; if it is approx. 1, then the listing is at market value. This will hopefully continue to segment our data.

NLP and Improved OLS

In the file 'NLP', we use a basic NLP parser looking for key words, we are able to improve the correlation values in an OLS to upwards of .2. This indicates that our these that rental type does highly impact the interest level given features. We hope to improve our classification, and use this technique to further split trees, neural nets, and regressions. Eventually, we will use a Bayesian classifier to convert these linear values back to 'high', 'medium', and 'low' interest levels. We believe that this technique, combined with trees will allow us to better classify the edge cases.

Interaction Terms

The file interaction_Terms explores adding interaction terms to the data and running linear regression to improve the model fit. Interaction terms were added to combine specific related predictors, such as allowing pets in general by synthesizing allowing cats and allowing dogs. By creating interaction terms, we hope to find an interaction term which allows us get stratification across interest level. One interaction term we added, outdoor_score, achieved some stratification. This term calculates an average score for the number of outdoor features a listing has. We discovered that only high interest listings have all of these features.

Tree Exploration

The file Tree explores various tree models on the original dataset, starting at a standard DecisionTreeClassifier(), and then performing hyper parameter optimization, using K-Fold to perform cross validation on the tree at different depths. The results were analyzed and from there we moved on to bagging. The bagged tree performed better than the Decision Tree by almost 2%. In an effort to find a tree model that, on all the predictors, unchanged, would perform a better classification, we then moved on to Random Forests. The hyper parameters were chosen again by K-Fold cross validation; however, some difficulties were reached on having sufficient computational power to run the cross validation. After using warm_start, the number of optimal trees was determined to be roughly 170, although the cross-validation results showed slightly better accuracies for larger numbers (>300). Moving on to the max number of features to be used, another K-Fold cross validation was run, however warm_start no longer benefitted us, so the best results we were able to obtain are from running each tree only once and testing its accuracy (without using K-Fold). The results are unsatisfactory, but moving forward we will be attempting tree-based classification with better training data that will ideally speed up the hyper-parameter selection process. After the above tree based classification methods were simulated, we briefly studied the results of the most optimal (at this point in time) RandomForestClassifier, including the predicted probabilities for each class for each sample. Moving forward, we are going to study how potentially changing the threshold for classification may change our accuracy. As it stands, the tree based models have a very high accuracy for "low" interest houses at the expense of medium and high ones.

Tree Exploration, Part 2

This file explores the results of the data processed using Natural Language Processing techniques that was used to classify listings based on the type of home. In this file, we first split up the dataset (only the subset that was successfully classified) into many smaller datasets based on type (effectively partitioning the data), and then train a Random Forest Classifier on each partition. From the results of our preliminary work on trees (see:), a Random Forest Classifier was used because it yielded the highest classification accuracy (roughtly .72). The classification accuracy for each individually fit trees was then compared to two separate Random Forest Classifier models. The first model the partitioned trees were compared to is a basic Random Forest Classifier trained without 'typed' data. The second is a Random Forest Classifier trained with all the 'typed' data (binary columns of type). The results of partitioned models were then compared, to see if a partitioned Random Forest performed better than the general, all inclusive random forest. We were trying to answer the question: are the features that are predictive of listing interest different for a different type of listing (apartment, penthouse, loft, etc)?

We answered this question by testing the two all-inclusive trees against the known typed data and comparing these trees to the performance of the typed-exclusive trees on their respective typed listing.

Support Vector Machines

This file explores fitting the data with support vector machines. It initially tries to fit a model to all of the cleaned data without interaction terms and gets 69.8% prediction accuracy. It then explores the hypothesis that we the interaction terms would add additional prediction accuracy. With interaction terms, the SVM model again gets 69.8% prediction accuracy, indicating that the interaction terms do not improve model fit. We then explored if fitting an SVM model to each listing type (studio, penthouse, townhome, etc.) would improve the prediction accuracy of listings. For some listings, such as lofts and other, the prediction accuracy improved, but for most of the listings, such as studios, apartments, condominiums, penthouses, townhomes, and walk ups, the prediction accuracy decreased. This file also explores adding interaction terms to each of these models by home type, but that worsened the prediction accuracy even more. Therefore, this file allowed us to conclude that an SVM model is a pretty accurate way of classifying the listings when not broken out by home type. Interaction terms can be included or not included since they don't improve the prediction accuracy that much.

(Brief) Neural Net Exploration

This file briefly explores using Neural Nets to classify housing listings into three different interest levels. In finding the best model, there are several parameters that can be tuned (using sklearn). The first, and primary, parameter studied in this file is alpha, the regularization constant. Alpha is tuned using k-fold cross-validation with just three trials each, as more trials took a very long time (did not wait to complete). The results are spotty, as the optimal alphas varied depending on the number of layers used. Overall, it seems that alpha = .1 performed the best, which would suggest that there are a number of predictors in the dataset that are insignificant, perhaps. Nonetheless, the results of the cross valdiation were not positive, as even the best performing NN with alpha = a had an accuracy of less than 50% (worse than a dummy classifier). As a result, we decided to stop pursuing neural nets and instead focus on more promising models/paths.