Model-Stacking for Apartment Demand Classification

Members

- Melwin Jose – 200150505
- Rohit Nambisan - 200153679

1) Business Value of RentHop data set

RentHop makes apartment search smarter by using data to sort rental listings by quality. But while looking for the perfect apartment is difficult enough, structuring and making sense of all available real estate data programmatically is even harder. The project tries to predict the interest level of new listing receives based on the listing's features, building_id, manager_id, creation date and other input variables. Doing so will help RentHop identify potential listing quality issues, and allow owners and agents to better understand renters' needs and preferences.

Learning objectives:

- How Gradient Boosted Trees can be used for multi-class classification.
- Detect the most significant features that helps in the classification task.
- Use text processing and other methods to reduce the dimensionality/cardinality.
- Learn how to use Model-Stacking to improve accuracy.

2) The Business questions pertaining to our Model:

- **How can the model improve the business of the RentHop website?**

We are proposing a similar kind of business model that is of google AdWords. RentHop could host advertisements from other business group in those webpages in RentHop that can have high traffic. The model can predict which apartments can have higher demand. So, if other business groups could market their products well in those webpages, then they can have more viewer-ship on those products.

- **How can clients of RentHop gain from this model?**

With this model, RentHop could venture into consulting services. They can provide variety of consulting services, like for example, they can advise the realtors which apartment in which area can have high demand, what features will the customer be looking for in a particular area, what can be the pricing for apartments in a particular area,etc.

- **How can this model balance the business for the realtors?**

With this model, we can understand the demand rate for all the apartments. So, we can infer that, higher the demand for an apartment, lower is the amount of advertising needs to be done for that apartment. So keeping this in mind, if we can market more for the low demand apartments, then we might be able to increase the revenue for that realtor. This way, the business for the realtor can be balanced, and there is more chances for all of his apartments getting a sale.

- **How can we increase the traffic to the RentHop website by using this model?**

By marketing the most high demand apartments to other sites, RentHop could earn a trust of providing the best service among all other rental websites.

3) Papers/Tutorials

- XGBoost: A Scalable Tree Boosting System
- Gradient Boosted feature selection
- Model Stacking: http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/

4) The Target Question:

What is the demand for a newly listed Apartment?

5) The possible business value in answering the target question:

The model predicts the demand level for a newly listed apartment. Using these results, the search results can be made efficient by always providing the in-demand apartment ads at top of the search result. In this age when UX and HCI is of great importance, this feature could make the renthop website stand out from all other rental websites.

6) Evaluation of Model from Business and Technological Perspective:

- Improved search results: Once the search results are improved, the users can be shown high demand listings for a given specification of the apartment. This helps a user to find apartment in shorter time. A metrics can be maintained for each search a user makes and number of apartments listings that he/she opened. A lower number for a given search indicates that the user was able to find what he/she was looking for.
- Improved in-site advertising: This can be evaluated by comparing the clicks on the ads that were posted along with high-demand listings vs that of medium and low demand listings.
- Better consultation services: Once the features that contribute to the demand of an apartment are identified, the consultation service provided by RentHop can be improved. This can be measured by the direct feedback from the clients and the impact on their business due to our service.

7) Literature Survey:

- **XGBoost: A scalable tree boosting system**

The main machine learning algorithm used was XGBoost, which stands for extreme gradient boosting. The paper aims at providing a detailed description regarding the system, its architecture and the additional features in XGBoost which are not present in other parallel boosting systems. XGBoost was a favorable tool for the solution because it is a tree ensemble model and they have very high accuracy level. An ensemble model is quite an accurate model as it involves combining the predictions of multiple models. This works perfectly especially in decision tree models, as combining decision trees “smoothes” out the prediction as the decisions made in decision tree model is the consequences of many if-else conditions. Also, according to the paper, 17 of the 29 winning solutions in Kaggle used XGBoost. There are various features in XGBoost that make it stand out from rest of the parallel boosting systems. They are,

Approximate Split Finding Algorithm: The key problem in tree learning is finding the best split. Most existing algorithms like R’s gbm and Python’s scikit-learn uses an algorithm called exact greedy algorithm where the algorithm enumerates all the possible splits and chooses the best one at each point of time. XGBoost uses the approximate algorithm, where, the algorithm begins split finding by proposing candidate splitting points according to the percentiles of feature distribution. Then, the data is mapped into buckets according to the splits and then aggregates these statistics and finds the best solution based on the aggregated statistics.

Sparsity-Aware Split-Finding Algorithm: Another key feature of XGBoost is that it can handle sparse data. A big problem faced while working with a real-time dataset is that the data could be sparse, due to missing values or due to feature engineering techniques like one-hot encoding. Before finding the split points, the algorithm should be aware of the sparsity of data. In order to solve that the algorithm adds a default direction in each node. When a value is missing, the instance is classified into default direction. The positive side is that the algorithm not only handles sparse data, but it also exploits the sparsity to make the computational complexity linear to number of non-missing entries in the dataset.

- **Gradient Boosted Feature Selection**

One of the most important part of our project was feature selection and proper feature engineering to improve the accuracy and efficiency of our model. With the given data, we were finding it hard to produce a highly accurate and efficient model, and with feature engineering we realized how proper feature selection made the model more accurate and use less memory, which consequently made the training phase faster. So, after proper research into this paper we realized where our shortcomings were and moved towards this direction. In this paper the authors propose a novel feature selection algorithm called Gradient Boosted Feature Selection(GBFS) and they evaluated the algorithm on many real world datasets and showed how it outperformed various other feature selection algorithms.

Before GBFS, algorithms like LARS are excellent detecting linear dependencies between features and labels but, they fail when features interact in nonlinear ways. Algorithms like random forest is really good in identifying nonlinear dependencies, but their computation and memory complexity increases with increase in training set sizes. GBFS tries to balance this issue of scalability and nonlinear feature selection issues.

In GBFS, trees are built with greedy CART algorithm, and when an important change in the impurity function results in feature selection. Also, when new features are splitted, a cost of $\lambda > 0$ is incurred, whereas reuse of old features incurs no penalty.

Some of the properties of GBFS are,

(1) As it learn an ensemble of regression trees, it can naturally discover nonlinear interactions between the features.

(2) GBFS has a time and space complexity of $O(dn)$, where d is the number of feature dimensionality and n is the number of data points.

(3) GBFS unifies feature selection and classification into single optimization.

(4) GBFS can incorporate pre-specified feature cost structures.

(5) GBFS is based on variation of gradient boosting of limited depth trees. This approach allows the algorithm to scale naturally to large datasets and combines learning a powerful classifier and performing feature selection in a single step.

The only bottleneck could be that, the time and space complexity is $O(dn)$, which could be a problem if there is a million features.

- **A Kaggler's Guide to Model Stacking in Practice**

After a great amount the feature selection and performance tuning the model seems to get saturated. Since then, it is very hard to get greater performance from the model. At that moment, we researched regarding how to improve the model further and we hit upon stacking. Stacking is a model ensembling technique used to combine information from multiple predictive models to generate a new model. It is also called meta-ensembling. Often time the stacked model do perform better than the previous model. Stacking becomes effective when the two models are entirely effective. Then the stacked model understands each of the model, and it enhances the base model capability where it performs better, and discredits where it performs poorly.

There are many ways for model stacking, we chose the one most favorable for our case.The steps involved where,

(1) Build the base models, with preferably entirely different models, predict the results.

(2) Form a training and test metadata. They are formed by combining the prediction results of the base models along with the original training and test data.

(3) The formed training and test metadata is then passed to the upper algorithm of the stacked model.

The good thing about stacking is that, it can identify the positive and negative aspects of each of its base models and make the decisions accordingly.

8) Feature Selection

Direct Features

- bathrooms
- bedrooms
- building_id
- latitude
- longitude
- manager_id
- log(price)

Engineered Features

- price_diff_bedrooms: price – mean price of apartments with same number of bedrooms
- total_rooms: bathrooms+bedrooms
- price_diff_rooms: price – mean price of apartments with same number of total_rooms
- bathbed: bathrooms/bedrooms
- bed_price: price/bedrooms
- room_price: price/total_rooms
- distances to the following places:
    - Parks: Central Park, Prospect, Pelham, Van Cortlandt, Flushing
    - Universities: New York University, Borough of Manhattan Community College, Columbia Unviersity, Hunter College, Kingsborough Community College, Bernard M Baruch College, Brooklyn College, New York, New York City College of Tech, City College, Touro College, John Jay College of Criminal Justice, Pace university, The New School, Fashion Institute of Technology
    - Subway Stations: Times Square, Grand Central, Herald Square, Union Square, Penn
- nphotos: number of photos
- freq_features: one hot encoding of hardwood floors, laundry, common laundry, no fee, pre-war
- display_addr: similar to display_address but with reduced cardinality using text processing
- price_diff_addr: price – mean price of apartments in the neighborhood

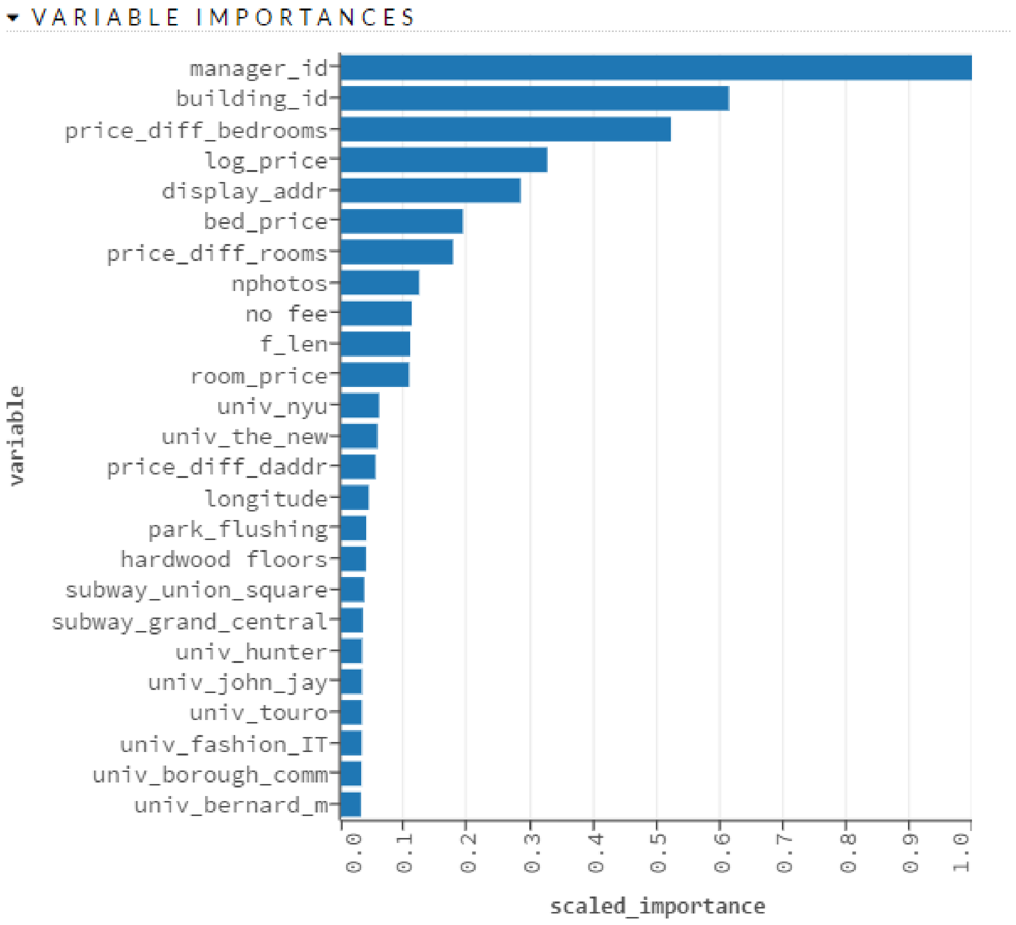
9) Solution

Text Processing for cardinality reduction

- The 'features' columns contains 1556 unique features, and a lot of them are duplicates. For example, "hardwood", "hardwood floor", "hardwood flooring" are duplicates of "hardwood floors". Similarly, other duplicate features can also be mapped to reduce the unique types of features. The list `freq_features_map` contains the mapping which is used to reduce them to 778 unique features.
    - Code: final_submission.R starting @line:388
- Similarly, the `display_address` has 8826 categories. As this column contains information other than the address and it can be processed to reduce the cardinality and to detect if the string contains non-address details. If it does, it is marked as "SKIP". A list of non-address words is stored in `skip_words` and these are remove along with non-alphabetic characters/words to reduce the cardinality to 1916.
    - Code: final_submission.R starting @line:733

Gradient Boosted Machine for feature Selection

- To select the features that were played the most significant role in predicting the results, we have used the "variable importance" plot from h2o's GBM. Variable importance is determined by calculating the relative influence of each variable: whether that variable was selected during splitting in the tree building process and how much the squared error (over all trees) improved as a result. More info: section "*8.1 Relative input of input variables*" in the paper "*Greedy Function Approximation: A Gradient Boosted Machine*"



- The above strategy was used to select the 45 most significant features by building a model on 100+ features, both direct and engineered ones, and dropping all those whose importance level was below threshold.
- Code: final_solution.R @line:890

Hyper-Parameter Tuning

- To find the best hyper-parameters for the XGBoost we train and tested the model on the following:

parameter	**values**	**comments**
nrounds	200, 400, 600, **800**, 100	number of rounds
subsample	0.2, 0.4, 0.6, **0.8**	subsample ratio of the training instance
colsample_bytree	0.3, 0.4, **0.5**, 0.6, 0.7	subsample ratio of columns when constructing each tree
min_child_wieght	50, **100**, 200	minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.
eta	0.0125, **0.025**, 0.05,	learning rate

The ones in bold gave the best accuracy with 10-fold cross validation and were used to train the level-2 XGBoost

Stacking

- In order into improve the accuracy of the classification and to reduce the variance, Model Stacking have been used. Our stack consists of the following two levels:
  - Level-1: It builds the following models on the train and make prediction both train and test.
    - Generalized Linear Model
    - Neural Network – 2 hidden layers
    - Random Forest
    - H2O's Gradient Boosted Machine
  - Level-2: uses the results from level-1 along with the train data to make predictions on the test data
    - Trained 40 XGBoost's with different seeds and depths (10-fold cross-validation). And the results were averaged.
- Code: final_solution.R @line:948

10) Project Materials

- Github link
  - Folder Structure:
    - papers/ : research papers
    - final_submission.R : our solution to the problem
    - lvl-1_<model>_[train|test].csv : predictions from models from level-1
    - ../data/*.json : train and test data
- Member Contribution
  - Code:
    - Melwin:
      - Text Processing for cardinality reduction of display_addr and features
      - Feature Selection
      - Model Stacking
    - Rohit:
      - Feature engineering of places
      - Hyper-Parameter Tuning