1.

* Latitude and longitude: Clustering based on demand
* Targeted Ads
* High demand features – using GBM

**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/

6 ???

**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

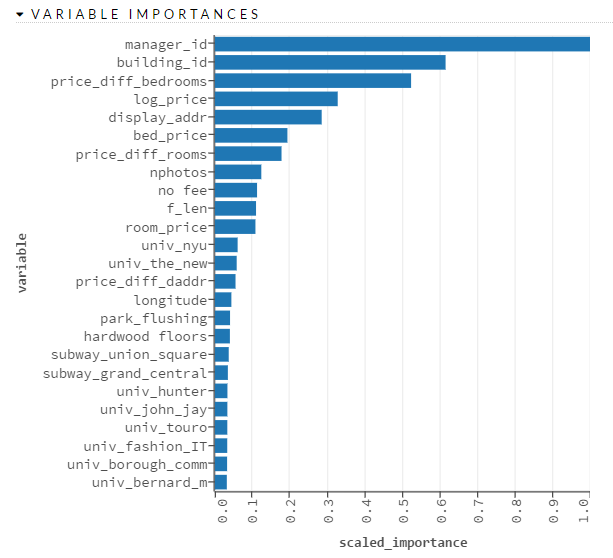
**9) Solution**

**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