Math 390 Final Project

Moshe Weiss

5/10/2019

In this model, as in all models, we seek to abstract reality from an unknowable number of unmeasurable affecting factors into several observable features. Using the data from 2016 and 2017 apartment sales in Queens, New York, along with the aid of algorithms and largescale computations, we seek to predict sale prices for apartments in the future. The three modelling techniques chosen are Linear Modelling, Regression Tree Modelling, and Random Forest Modelling. At a high level, these function as follows: Linear Modelling seeks to draw out the mean of each feature from the mean of our observed variable, in order to understand how much influence each observed feature has on sale price Regression Tree Modelling seeks to find the most efficient and significant ways to split the dataset, and constructs a decision tree based on those splits. Random Forest Modelling performs many Regression Trees while augmenting our dataset by capitalizing on the Bias-Variance tradeoff. Ultimately, it returns an aggregated model that is highly reflective of our data. If our sample is large enough, this model carries the weight of our data the farthest in terms of predictive power

```r
#Dependencies
pacman::p_load(dplyr, tidyr, magrittr, mlr, missForest)
#pacman::p_install_gh("kapelner/YARF", subdir = "YARF", ref = "dev")
#pacman::p_install_gh("kapelner/YARF", subdir = "YARFJARs", ref = "dev")
#pacman::p_install_gh("kapelner/YARF", subdir = "YARF", ref = "dev")
pacman::p_load(YARF)
```

The data used for our model is from Multiple Listing Services Listings. The data contains 2,300 rows and 55 columns, of which 15 columns are used to make predictions. Our features include categorical variables (kitchen_type, garage_exists, dining_room_type, fuel_type, cats allowed, dogs_allowed, coop_condo), continuous variables (mainteance_cost, sq_footage, num_floors_in_building, approx_year_built, num_full_bathrooms, num_total_rooms, num_bedrooms), and an aggregate metric (walk_score).

```r
#Data Cleaning and Imputation
housing_data = read.csv("housing_data_2016_2017.csv", stringsAsFactors = FALSE)
#fix feature selection
q_housing = as.data.frame(housing_data)
```

Overall, the names of the apartment features are very telling. Num_bedrooms, num_full_bathrooms, num_total_rooms, num_floors_in_building, approx_year_built, and sq_footage need no further explanation. Kitchen_type is either eat in, efficiency, or none. Dining_room_type is either combo, formal, or other. Fuel_type is either electric, gas, oil, or other. Garage_exists, dogs_allowed and cats_allowed are all encoded as binary. Maintenance_cost refers to the building's monthly maintenance charges. Coop_condo is whether the apartment is part of a co-op or condominium. Walk_score is an aggregate metric refers walkability with respect to nearby amenities, on a scale of 1 to 100.

```r
q_housing %<>%
  select(num_bedrooms, num_floors_in_building, kitchen_type, maintenance_cost, num_full_bathrooms, num_

  mutate(kitchen_type = tolower(kitchen_type)) %>%
  mutate(kitchen_type = ifelse(substr(.$kitchen_type,1,3) == "eff","eff", ifelse(substr(.$kitchen_type,
  mutate(kitchen_type = factor(kitchen_type, ordered = FALSE)) %>%
  mutate(dining_room_type = factor(tolower(dining_room_type), ordered = FALSE)) %>%
  mutate(fuel_type = factor(tolower(fuel_type), ordered = FALSE)) %>%
  mutate(maintenance_cost = as.numeric(factor(maintenance_cost, ordered = FALSE))) %>%
  mutate(dogs_allowed = ifelse(substr(.$dogs_allowed, 1, 1) == "y", 1, 0)) %>%
  mutate(cats_allowed = ifelse(substr(.$cats_allowed, 1, 1) == "y", 1, 0)) %>%
  mutate(sale_price =  as.numeric(gsub('[$,]', '', q_housing$sale_price))) %>%
  mutate(coop_condo = factor(tolower(coop_condo))) %>%
  mutate(garage_exists = ifelse(is.na(garage_exists), 0, 1)) #%>%
#ID col for selecting missing y's
  #mutate(id = 1:nrow(q_housing))

#place 2 none values in other
  q_housing$dining_room_type[q_housing$dining_room_type == "none" & !is.na(q_housing$dining_room_type)]

#place 2 dining area values in other
  q_housing$dining_room_type[q_housing$dining_room_type == "dining area" & !is.na(q_housing$dining_room_

#place 3 none values for fuel_type in other
  q_housing$fuel_type[q_housing$fuel_type == "none" & !is.na(q_housing$fuel_type)] = "other"
```

The data had vast swathes of missing, nonsense, not useful, or misspelled entries. Taxes of each apartment had 650 entries, of which many were ridiculously low (as low as $13 a year). For example, some inputs for taxes are $13 per year. As such, we did not include taxes in our model. In the case of community district, there was not enough occurrences of variations in the data to successfully predict on it categorically, and it was dropped for this technical reason. In the case of the garage_exists column which we included, 1,800 entries were missing. The other 500, however, were all "yes", and we surmised NA encoded to "no". The rest of our utilized features were supplemented using Random Forest imputation with the missForest package.

```
#identifying missing y's

# naEntries = q_housing %>%
#               filter(is.na(sale_price))
# modellingEntries = setdiff(q_housing, naEntries)

missingTable = tbl_df(apply(is.na(q_housing), 2, as.numeric))

colnames(missingTable) = paste("is_missing_", colnames(q_housing), sep = "")
missingTable = tbl_df(t(unique(t(missingTable))))
missingTable %<>%
  select_if(function(x){sum(x) > 0})

imp_q_housing = missForest(q_housing, sampsize = rep(525, ncol(q_housing)))$ximp
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
##   missForest iteration 6 in progress...done!
##   missForest iteration 7 in progress...done!
##   missForest iteration 8 in progress...done!
##   missForest iteration 9 in progress...done!
##   missForest iteration 10 in progress...done!
```

```
#Includes missingness in dataset. No significant difference observed.
  #imp_q_housing = cbind(imp_q_housing, missingTable)


#Remove all rows with imputed sale_price
  #This lowers our R squared by a lot, but is probably more representative of our
  #predictive power.

#imp_q_housing = imp_q_housing[modellingEntries$id,]

#imp_q_housing$id = NULL
```

The Ordinary Least Squares model performed with an R-squared of 85% and an aggregated RMSE of 64,00. This linear model performs 85% better than simply taking the average price and using it to predict, and our out of sample estimates are generally within +/- $64,000 of the actual price. Interpreting the coefficients of this linear model imply that the most expensive factor to increase in one unit (holding other factors constant) is whether that apartment is part of a co-op or condo. This is followed by the number of bathrooms, and then by the number of bedrooms. Square footage will likely be among these as well, as these are always increased in units larger than 1. A linear model will be good, but not be ideal for predicting sale price. This is because the factors interact complexly, and don't necessarily yield linear growth in each situation. Take luxury apartments for example; when a minor increases is made in square footage, there is likely an disproportionately large increase in price.

```
#REGRESSION OLS
modeling_task = makeRegrTask(data = imp_q_housing, target = "sale_price")
```

```
## Warning in makeTask(type = type, data = data, weights = weights,
## blocking = blocking, : Empty factor levels were dropped for columns:
## dining_room_type,fuel_type
```

```
algorithm = makeLearner("regr.lm")
validation = makeResampleDesc("CV", iters = 5) #instantiate the 5-fold CV
pred = resample(algorithm, modeling_task, validation, measures = list(rmse))
```

```
## Resampling: cross-validation

## Measures:             rmse

## [Resample] iter 1:    60498.2512087

## [Resample] iter 2:    58860.2577249

## [Resample] iter 3:    65073.9836997

## [Resample] iter 4:    60610.1090969

## [Resample] iter 5:    79113.8875527

##

## Aggregated Result: rmse.test.rmse=65256.0866460

##
```

```
mean(pred$measures.test$rmse)
```

```
## [1] 64831.3
```

```
sd(pred$measures.test$rmse)
```

```
## [1] 8311.112
```

```
linmod = lm(sale_price ~ ., imp_q_housing)
summary(linmod)
```

```
##
## Call:
## lm(formula = sale_price ~ ., data = imp_q_housing)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -684817  -34954   -1465   34519  406789
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.099e+06  1.920e+05  -5.723 1.19e-08 ***
## num_bedrooms            4.356e+04  3.271e+03  13.314  < 2e-16 ***
## num_floors_in_building  7.171e+03  2.461e+02  29.144  < 2e-16 ***
## kitchen_typeeatin       2.125e+03  3.818e+03   0.557   0.5778
## kitchen_typeeff        -2.342e+04  3.930e+03  -5.959 2.94e-09 ***
## kitchen_typenone       -1.062e+04  1.361e+04  -0.781   0.4350
## maintenance_cost       -7.636e+01  1.007e+01  -7.582 4.98e-14 ***
## num_full_bathrooms      5.430e+04  4.484e+03  12.110  < 2e-16 ***
## num_total_rooms         3.795e+03  1.868e+03   2.032   0.0423 *
## sq_footage              1.201e+02  7.305e+00  16.439  < 2e-16 ***
## walk_score              7.063e+02  9.765e+01   7.233 6.47e-13 ***
## dining_room_typeformal  2.150e+04  3.232e+03   6.651 3.66e-11 ***
## dining_room_typeother   1.918e+04  4.780e+03   4.012 6.21e-05 ***
## fuel_typegas           -1.874e+04  8.304e+03  -2.257   0.0241 *
## fuel_typeoil           -1.878e+04  8.488e+03  -2.213   0.0270 *
## fuel_typeother          1.798e+04  1.280e+04   1.404   0.1603
## cats_allowed            1.732e+03  4.099e+03   0.422   0.6728
## dogs_allowed            1.835e+04  4.565e+03   4.019 6.04e-05 ***
## coop_condocondo         1.519e+05  4.596e+03  33.045  < 2e-16 ***
## approx_year_built       5.395e+02  9.789e+01   5.512 3.97e-08 ***
## garage_exists          -7.400e+03  3.796e+03  -1.950   0.0513 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 62520 on 2209 degrees of freedom
## Multiple R-squared:  0.849,  Adjusted R-squared:  0.8476
## F-statistic: 620.9 on 20 and 2209 DF,  p-value: < 2.2e-16
```

Regression trees indicate the most important features at several levels by optimizing divisions of the dataset. This technique enables us to quickly observe the most telling variables of sale price, and encapsulates interactions between features intuitively. The most significant split in the tree is on the number of full bathrooms with a value greater or less than 1.5. This is perhaps because apartments with more than one full bathroom are generally very large or luxurious, as one is generally sufficient for an apartment-size living space. In apartments with less than 1 bathroom, the next split is on co-op or condo. Generally, co-ops offer lower prices up front in exchange for a good amount of control over tenants. The next two low-end splits are on square footage; living space is extremely important to low-cost apartments. In apartments with greater than 1 bathroom, the next split is on the square footage of the apartment. As this is likely the large or luxury apartment category, size likely begins to matter in much the same way number of bathrooms did. On the high end of that split, we see the next feature is the number of floors in the building. This may be because luxury apartments cost more than non-luxury large apartments and are often found in high-rise buildings. On the low end of that split is approximate year built, which likely is a result of advancements in technology and changes in the standards of architecture. After the split on the number of floors in the building, square footage asserts its importance; both the upper and lower splits are on that category. This makes sense, as an increase in square footage in a high-riser requires a tremendous amount of cost and resources.

```r
#REGRESSION TREE
options(java.parameters = "-Xmx4000m")
pacman::p_load(YARF)
```

```r
test_prop = 0.1

train_indices = sample(1 : nrow(imp_q_housing), round((1 - test_prop) * nrow(imp_q_housing)))
imp_q_train = imp_q_housing[train_indices, ]
y_train = imp_q_train$sale_price
X_train = imp_q_train
X_train$sale_price = NULL

test_indices = setdiff(1 : nrow(imp_q_train), train_indices)
imp_q_test = imp_q_housing[test_indices, ]
y_test = imp_q_test$sale_price
X_test = imp_q_test
X_test$sale_price = NULL

tree_mod = YARFCART(data.frame(X = X_train), y_train)
```

```
## YARF initializing with a fixed 1 trees...
## YARF factors created...
## YARF after data preprocessed... 24 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```r
tree_mod
```

```
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 2007 and p = 24
## Model construction completed within 0.01 minutes.
## No OOB results to show (no trees have been fit as of yet).
```

```r
get_tree_num_nodes_leaves_max_depths(tree_mod)
```

```
## $num_nodes
## [1] 1603
##
## $num_leaves
## [1] 802
##
## $max_depths
## [1] 24
```

```r
illustrate_trees(tree_mod, max_depth = 4, open_file = TRUE, length_in_px_per_half_split = 30)
```

Random Forest modeling provides incredible predictive mileage out of our data. This algorithm is non-parametric, because as we observe more data (in the present case, apartment sales) we can increase the number of splits freely. We begin with our dataset in a supervised learning environment. We then augment our dataset with a process called bootstrapping, in which we sample from our dataset with replacement. We iteratively create trees, where each tree is split on a random selection of the features. Each individual tree has high variance and idiosyncrasy but taking an average on the splits of all tree diminishes our variance. We've bootstrapped, so the increase in our bias is negligible. Random Forests lets us squeeze our data to maximize the predictive juice our model can give us. The model fit the data well, as is evident by the out of bag (OOB) metrics that are described shortly. Ultimately, I believe the features that were most significantly decisive in price are the same regardless of the model. There could be a truly causal relationship between the number of bathrooms, the number of bedrooms, and/or the square footage on sale price, though we would need more data to truly come to that conclusion.

```r
#RANDOM FOREST

y = imp_q_housing$sale_price
X = imp_q_housing
X$sale_price= NULL

mod_rf = YARF(X, y, num_trees = 300)
```

```
## YARF initializing with a fixed 300 trees...
## YARF factors created...
## YARF after data preprocessed... 24 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```r
mod_rf
```

```
## YARF v1.0 for regression
## Missing data feature ON.
## 300 trees, training data n = 2230 and p = 24
## Model construction completed within 0.42 minutes.
## OOB results on all observations:
##   R^2: 0.95952
##   RMSE: 32210.98
##   MAE: 14114.39
##   L2: 2.31373e+12
##   L1: 31475084
```

```r
illustrate_trees(mod_rf, max_depth = 4, open_file = TRUE, length_in_px_per_half_split = 30)

#mlr random forest
# modeling_task = makeRegrTask(data = imp_q_housing, target = "sale_price")
# algorithm = makeLearner("regr.randomForest")
# validation = makeResampleDesc("CV", iters = 5) #instantiate the 5-fold CV
# pred = resample(algorithm, modeling_task, validation, measures = list(rmse))
# mean(pred$measures.test$rmse)
# sd(pred$measures.test$rmse)

test_prop = 0.1

train_indices = sample(1 : nrow(imp_q_housing), round((1 - test_prop) * nrow(imp_q_housing)))
imp_q_train = imp_q_housing[train_indices, ]
y_train = imp_q_train$sale_price
X_train = imp_q_train
X_train$sale_price = NULL

test_indices = setdiff(1 : nrow(imp_q_train), train_indices)
imp_q_test = imp_q_housing[test_indices, ]
y_test = imp_q_test$sale_price
X_test = imp_q_test
X_test$sale_price = NULL

holdout_mod_rf = YARF(X_train, y_train, num_trees = 300)
```

```
## YARF initializing with a fixed 300 trees...
## YARF factors created...
## YARF after data preprocessed... 24 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```
holdout_mod_rf
```

```
## YARF v1.0 for regression
## Missing data feature ON.
## 300 trees, training data n = 2007 and p = 24
## Model construction completed within 0.46 minutes.
## OOB results on all observations:
##   R^2: 0.95985
##   RMSE: 32210.38
##   MAE: 14512.78
##   L2: 2.08228e+12
##   L1: 29127144
```

```
y_hat_test = predict(holdout_mod_rf, X_test)
holdout_rmse = sqrt(mean((y_test - y_hat_test)^2))
holdout_rmse
```

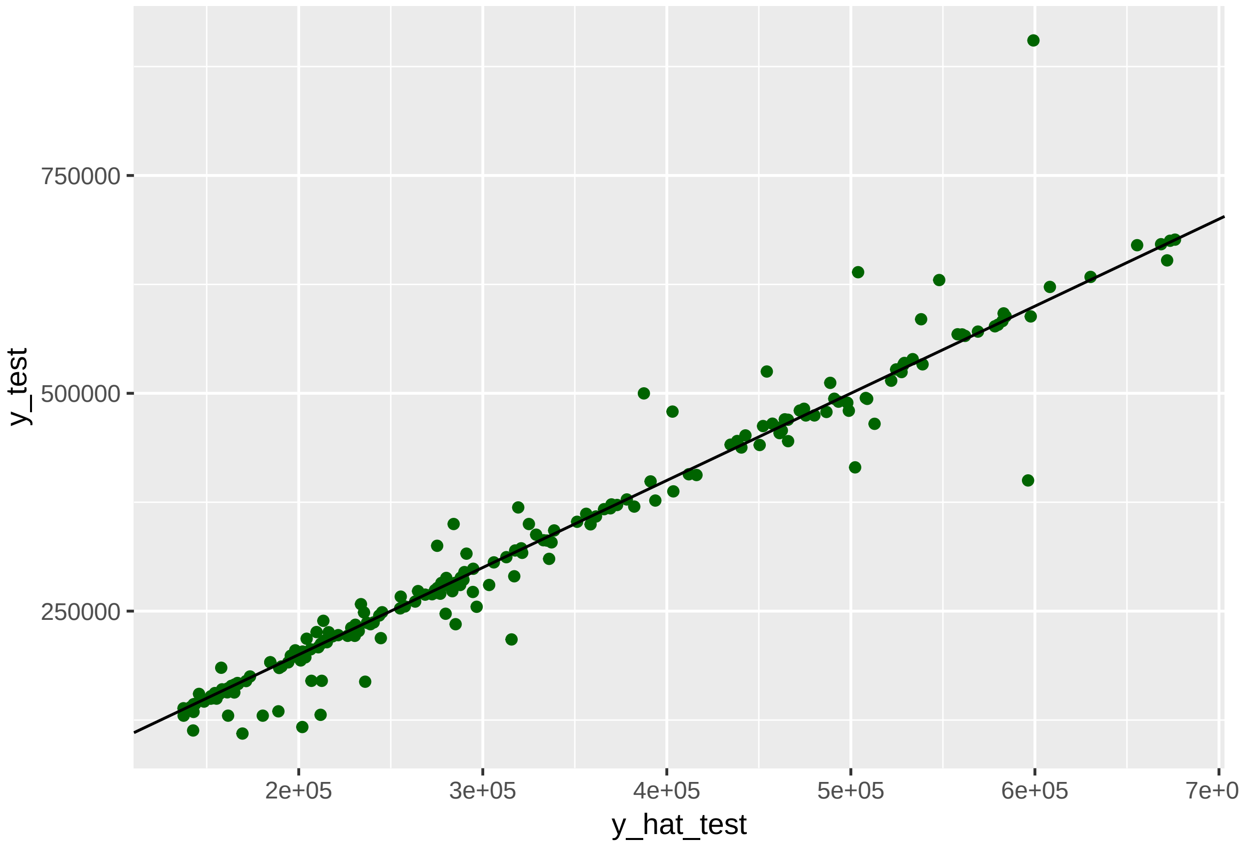
```
## [1] 36441.72
```

```
holdout_rsq = 1 - sum((y_test - y_hat_test)^2)/sum((y_test - mean(y))^2)
holdout_rsq
```

```
## [1] 0.945767
```

OOB R2 for this model is 0.9601. OOB RMSE is 31,990.03. These are calculated by predicting on the entries randomly left out of the individual trees that compose our Random Forest. The former indicates our RF performs 96% better than the null model, and our error is, on average + $31,990.03. This is our estimate of generalization error, and we know this is a valid estimate of performance because of the nature of out of bag sampling. We are able to use the data that was randomly left out in each tree to generate out of sample metrics, and assuming statisticity, we are left with a model capable of predicting future sale prices. Running a hold-out test yields similar results, with an OOS R2 of 0.9672 and an OOS RMSE of 27869.95. While these out of sample metrics are better, they are on a model informed by less data, and as such are less representative.

```
ggplot(aes(y_hat_test, y_test), data = data.frame(y_test, y_hat_test)) +
  geom_point(col = "darkgreen")+
  geom_abline(col = "black")
```



This graphic compares y_test, our partition of the data reserved for testing, to y_hat_test, our predictions from the model generated by the training data. Overall, we can see a strong tendency toward a slope of 1, which indicates our predictions are very similar to the data.

Discussion Our goal was to create an accurate model for predicting housing prices. With the use of the 2016-2017 housing data and Linear, Tree, and Random Forest modelling, we created three models that succeeded in doing so. The largest point of contention I have with our project is the way in which we handled data after imputation. The appropriate protocol would have been to drop data points that did not have recorded sale prices. Had we done that, we would have had only 80% of our data left, which would have decreased our predictive power. Our OOB R2 in the Random Forest would have dropped to 83%. While this is better than the R2 of the zestimates discussed in class, much improvement could be made. To appropriately combat this, I would have liked to reduce the number of NA's in the data set by manually seeking out their entries, improving our imputation. I would also increase the amount of data and the feature space by trawling the web and researching real estate's impacting factors. To increase the number of observations, I would expand our sources from Multiple Listing Services alone. The data accumulated from other locations would likely not include the same features as our current data set, and we would need modify, clean, and supplement our new, larger data set. Time since last renovation and apartment quality would greatly affect sale price and would possibly serve as a better point of reference for sale price than a feature such as approximate year built. Additionally, rent regulation would affect sale price: the more limitation on how much the prospective buyer could rent the apartment for, the lower its potential sale price. This could be encoded into a single binary variable, a continuous variable (maximum rental price), or a categorical variable. Regression trees seemed to consistently split on 1970 in the year built feature, and then on fuel type. After researching this, Joseph discovered that there was an oil crisis in the early 70's, and this may have forced developers to resort to natural gas as apartment fuel. This is a fascinating example of how unknown occurrences can affect features that closely relate to our prediction, and with the use of modelling techniques and data science, we can capture trends despite lack of knowledge.