Final Project

Kurt Werber

MATH 390

2020/24/04

Abstract

A quick foray into predicting housing prices, comparing both linear and non-linear models: OLS, Regression Trees, and Random Forests. The aim is to beat the Zestimates of Zillow at predicting sale prices. It doesn't go well, but things do improve.

1 Introduction

In this paper I will be attempting to predict house selling prices based on 2016-2017 data.

A predictive model is something that tries to guess an outcome. There are good guesses and there are bad guesses, so all we can do is attempt to get the best guess possible. We will be examining the results of 3 such guesses, using Regression Tree, Linear, and Random Forest models. These models will be guessing the response, the selling price, measured in dollars, of each observation, the houses. Each has several variables associated with it, including categorical, numerical, and binary types. The resulting performance was depressingly bad, but there was improvement in performance with each model.

2 The Data

The raw data is the 2016-2017 housing data provided by MSLI and harvested via Amazon's MTurk. The raw data frame was 2230 observations of 55 variables. The data may not be entirely representative of the population as a whole since the median selling price $259,500, while Zillow claims the actual median selling price in Queens is $537,350. Because of this any predictions based on this data may be better for homes with lower value, and extrapolation is a potential trap.

2.1 Featurization

After wrangling the data I was left with 21 variables. Many of the original 55 were clearly clerical tags, so there was no meaning to be gotten from them, and many of them were the same for all observations. These were dropped. Of the 2230 observations, only 528 had the sale price reported, so all others were dropped. The variables used were:

- `WorkerId`: This was the only clerical tag kept. It is unclear who exactly the workers in question are, but it was reasonable to assume that if it was someone working on selling the houses such as a real estate agent, that might influence the final outcome. There are 21 unique worker IDs in total.

- `zip_code`: The ZIP code of the house sold. These were extracted from a variable in the raw data, `full_address_and_zip_code`. Real estate is all about "location, location, location" so it is expected for the ZIP code to have an appreciable effect.

- `approx_year_built`: The year the house was built. After checking for a possible parabolic shape due to the potential for historic homes to have more value, it was clear that it was just a slight positive correlation. The build year ranged from 1915 to 2016, with median year 1957.

- `maintenance_cost`: It was unclear exactly what this referred to, possibly either an estimate of the cost or HOA fees. It ranged from 155 to 4659 with median 734.

- `pets_allowed`: This was the combination of two variables from the raw data detailing whether dogs or cats individually allowed. In order to give a stronger influence to whether pet allowance is significant, these were combined into one binary variable. 283 apartments allowed pets, 245 did not.

- `garage_exists`: This variable was taken from the raw where it had multiple factors such as "yes" or "underground", but this was transformed to another binary variable, since most people presumably just care that they have a guaranteed spot, and not whether it is below or above ground. Of the 528 houses, 94 had garages.

- `num_bedrooms`: Number of bedrooms in each house. The median amount was 1, with 3 being the most.

- `num_full_bathrooms`: Number of bathrooms with shower/tub. Median amount was 1, highest was 3.

- `num_half_bathrooms`: Number of bathrooms without shower/tub. Median was 0, highest was 2.

- num_total_rooms: Total number of rooms in the house. Median was 4, ranging from 1 to 8.
- parking_charges: Cost of onsight parking.
- sq_footage: Size of apartment in square feet. Ranged from 375 to 6215, with median 874.
- total_taxes: Total amount of tax on the property in dollars. Ranged from 11 to 9300 with median 2148.
- is_coop: Binary variable representing if apartment is a coop. Taken from raw data where it was a factor variable with either coop or condo. 399 of the apartments were coops.
- community_district_num: The school district the apartment is in, which can be a big selling point for people with children. 15 districts were represented.
- dining_room_type: Whether there is a formal dining room, a combined dining area, or some other arrangement.
- fuel_type: Whether the house is heated with gas, oil, or another source.
- kitchen_type: Whether the kitchen has room for a table, is in a combined area, or is an "efficiency" kitchen which essentially means it has a small space only for food preparation.
- walk_score: A metric measuring how walkable the area around the house is. It ranged from 15 to 99 with median 85, so overall very high, as expected for Queens.

2.2 Errors and missingness

As stated before, all observations from the raw data that were missing the sale price were dropped. The only true error discovered in the data was a house which has its kitchen type listed as 1955, which is obviously nonsensical. This was fixed by deleting that cell so that it could be imputed later. Other than that, there were some issues with spelling and inconsistent capitalization that needed to be fixed.

For some variables, there was a clear meaning to the missingness.

- `num_half_bathrooms`: Most values were missing, and the only ones that had values reported were either 1 or 2. Due to the cramped nature of Queens meaning half-bathrooms aren't always feasible, all missing values in this column were set to zero.
- `parking_charges`: For a similar reason, and once again none of the non-missing values were 0, all missing values were set to zero.

Imputation was done in 2 steps. The first was done by partitioning the numerical data so that it could be imputed using a `missForest`. The categorical data was then added back and imputed using Bayesian polytomous regression via the `MICE` package.

3 Modeling

In this section are the results of modeling with Regression Trees, Linear Regression, and Random Forests

3.1 Regression Tree Modeling

The results of modeling sale price via CART are displayed in Figure 1. As is shown, the total number of bathrooms, square-footage, and "coop-ness" were found to be the most important for modeling sale price. RMSE was 92,324.

The variables used in the tree are shown below.

```
Variables actually used in tree construction:
[1] approx_year_built      community_district_num is_coop
num_full_bathrooms     sq_footage             walk_score
[7] zip_code
```

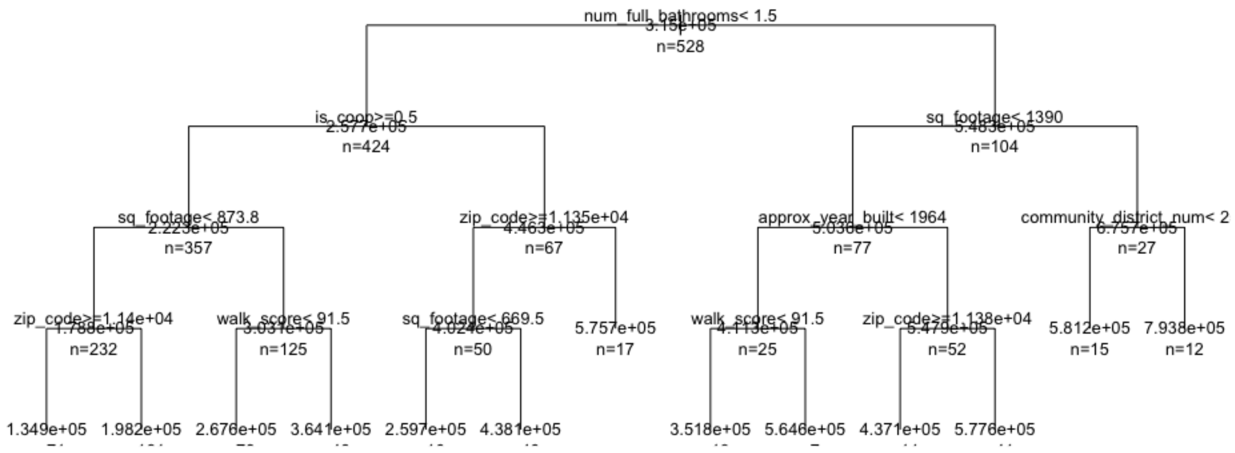

Figure 1: Regression Tree

3.2 Linear Modeling

The OLS estimates and metrics are included in Listing 1. The out-of-sample RMSE was 98001, and out-of-sample R^2 is 0.68. Due to the large number of factors for `zip_code` and `community_district_num`, they had to be omitted because some of the factors did not wind up in the training data.

It is clear that the tree model and linear model disagree on the signifcance of the number of bedrooms. The tree did not split on `num_bedrooms` while the coefficient estimate for it is quite significant in the linear model. The tree split on `sq_footage` while OLS found it insignificant, so both models found the overall size signifciant, but in different variables.

In addition, the dummy variable for the "efficiency" factor of `kitchen_type` has quite a strong negative sway. The base category for this category is "combo", which may reflect modern tastes for open plan spaces in houses. In both models coops are expected to have lower sale prices, with `is_coop` being significant both times.

Overall, the size and location of the apartments, as expected, have the most sway in both

$oobR^2$	0.776
$oobRMSE$	84816
test $RMSE$	72758

models.

3.3 Random Forest Modeling

Random Forests are a good choice for this scenario. For one, it can handle all the different types of data, including categorical, numeric, and binary. This is invaluable in something as complex as an entire house. In addition, they handle high-dimension data very well, and doesn't care about linearity. If sale prices as a function of time was actually parabolic as previously speculated, the forest would be able to handle it.

However, the issue of interpretability here is evident. When selling houses, it can be useful to know what exactly is important, so a random forest's man-behind-the-curtain approach can be a detriment when trying to understand the strength of the relationship between a measurement and its associated response.

4 Random Forest Performance

As shown in the table, this model is far from the best. One can expect to be off by more than 70,000 dollars on average, which even for a house is extraordinarily large.

5 Discussion

As we have seen, the models presented in this paper have been varying degrees of bad, the metrics speak for themselves.

For starters, better data will go a long way. The raw data was plagued with missingness, and despite the imputations, it would have performed better if some of the variables weren't more than half missing. Because of this, three quarters of the data was immediately lost

Listing 1: Linear model

```
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -5.914e+05  7.569e+05  -0.781 0.435075
approx_year_built                3.565e+02  3.801e+02   0.938 0.348829
garage_exists                    5.030e+03  1.351e+04   0.372 0.709847
maintenance_cost                 1.657e+02  2.036e+01   8.136 5.43e-15 ***
num_bedrooms                     4.371e+04  1.092e+04   4.004 7.45e-05 ***
num_full_bathrooms               8.984e+04  1.631e+04   5.508 6.57e-08 ***
num_half_bathrooms               4.215e+04  2.123e+04   1.986 0.047744 *
num_total_rooms                 -6.509e+03  7.405e+03  -0.879 0.379905
parking_charges                  8.010e+01  9.840e+01   0.814 0.416142
sq_footage                      -1.237e+01  1.772e+01  -0.698 0.485642
total_taxes                      6.903e+00  5.113e+00   1.350 0.177808
walk_score                       1.641e+03  3.773e+02   4.348 1.75e-05 ***
is_coop                         -2.483e+05  4.378e+04  -5.672 2.74e-08 ***
pets_allowed                     1.104e+04  9.759e+03   1.131 0.258751
dining_room_typeformal           9.674e+03  1.173e+04   0.825 0.409909
dining_room_typeother            2.838e+04  1.431e+04   1.984 0.047981 *
fuel_typegas                    -1.210e+03  3.155e+04  -0.038 0.969438
fuel_typeoil                    -2.467e+03  3.252e+04  -0.076 0.939588
fuel_typeother                   2.941e+04  4.371e+04   0.673 0.501447
kitchen_typeeat in              -2.703e+04  1.414e+04  -1.911 0.056685 .
kitchen_typeefficiency          -5.131e+04  1.427e+04  -3.596 0.000365 ***
is_missing_approx_year_built     5.470e+04  4.165e+04   1.314 0.189768
is_missing_community_district_num  1.464e+05  9.617e+04   1.523 0.128676
is_missing_dining_room_type      8.434e+03  1.156e+04   0.730 0.466071
is_missing_fuel_type             1.584e+04  2.101e+04   0.754 0.451364
is_missing_kitchen_type         -8.306e+04  3.872e+04  -2.145 0.032551 *
is_missing_maintenance_cost     -4.268e+04  2.168e+04  -1.968 0.049743 *
is_missing_sq_footage           -7.065e+03  9.980e+03  -0.708 0.479424
is_missing_total_taxes           1.560e+03  3.717e+04   0.042 0.966539
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

Residual standard error: 93770 on 393 degrees of freedom
Multiple R-squared:  0.7493,   Adjusted R-squared:  0.7314
F-statistic: 41.95 on 28 and 393 DF,  p-value: < 2.2e-16
```

due to it lacking the sale price. The attempt to use so many categorical variables may have impeded progress rather than helped. In addition, fine tuning the hyperparameters for the random forest algorithm will help get the variance down and improve the accuracy. So for now, I won't be beating Zillow.

6 Code appendix

```
attach(housingdata)

skim(sale_price)
summary(sale_price)

housingdata %<>%
  filter(!is.na(sale_price))


housingdata$sale_price %<>%
  parse_number()

n <- nrow(housingdata)

skim(housingdata)

housingdata[, which(colSums(is.na.data.frame(housingdata)) == n)] <- NULL

housingdata$WorkerId %<>%
  as.factor()

housingdata$community_district_num %<>%
  as.factor()

housingdata$is_coop <- coop_condo == "co-op"

cats_allowed <- cats_allowed == "yes"

dogs_allowed <- dogs_allowed == "yes"

housingdata$pets_allowed <- cats_allowed | dogs_allowed

housingdata$cats_allowed <- NULL
housingdata$dogs_allowed <- NULL

colnames(housingdata)

skim(housingdata)

housingdata[, 1:12] <-  NULL
```

```
housingdata[, 2:11] <- NULL

housingdata$zip_code <- stringr::str_extract(housingdata$full_address_or_zip_code,
                                                        pattern = "11[0-9]{3}") %>%
                                                as.factor()

housingdata$full_address_or_zip_code <- NULL

garage_exists <-  !is.na(garage_exists)
housingdata$garage_exists <- garage_exists

housingdata$date_of_sale %<>%
  as.Date(format = "%m/%d/%Y")
class(housingdata$date_of_sale)

housingdata$coop_condo <- NULL

housingdata$model_type <- NULL

housingdata$num_floors_in_building <- NULL

housingdata$pct_tax_deductibl <- NULL

housingdata$common_charges <- NULL

apply(housingdata$dining_room_type, FUN = function(x){if(x == "dining area"){x <- "other"}})

housingdata$dining_room_type[housingdata$dining_room_type == "dining area"] <- "other"

housingdata$fuel_type[housingdata$fuel_type %in% c("none", "other", "Other")] <- "other"

housingdata$kitchen_type[kitchen_type %in% c("eat in", "Eat in", "Eat In")] <- "eat in"

housingdata$kitchen_type[kitchen_type %in% c("combo", "Combo")] <- "combo"

housingdata$kitchen_type[kitchen_type == "1955"] <- NA

housingdata$kitchen_type <- droplevels.factor(x = housingdata$kitchen_type)

housingdata$num_half_bathrooms[is.na(num_half_bathrooms)] <- 0
housingdata$parking_charges[is.na(parking_charges)] <- 0

housingdata$maintenance_cost %<>%
```

```
  parse_number()

housingdata$parking_charges %<>%
  parse_number()

housingdata$common_charges %<>%
  parse_number()

housingdata$total_taxes %<>%
  parse_number()

housingdata$dining_room_type %<>%
  as.factor()

housingdata$fuel_type %<>%
  as.factor()

housingdata$kitchen_type %<>%
  as.factor()

housingdata$zip_code %<>%
  as.factor()

housingdata$community_district_num %<>%
  as.factor()

housingdata$WorkerId %<>%
  as.factor()

housingdata$approx_year_built %<>%
  as.integer()

M = tbl_df(apply(is.na(housingdata), 2, as.numeric))
colnames(M) = paste("is_missing_", colnames(housingdata), sep = "")

M = tbl_df(t(unique(t(M))))
M %<>%
  select_if(function(x){sum(x) > 0})

housingdata %>%
  select_if(function(x){is.logical(x) | is.numeric(x)}) %>%
  select(-sale_price) %>%
  as.matrix() -> Xnum
```

```r
Xnum_imp <- missForest::missForest(Xnum)

cbind.data.frame(Xnum_imp$ximp, housingdata %>%
                     select_if(function(x){!(is.logical(x) | is.numeric(x))})) -> housing_imp_1


Xcat <- housingdata %>%
          select_if(is.factor)

ggplot(data = housingdata, aes(approx_year_built, sale_price)) +
  geom_point()

ggplot(data = housingdata, aes(date_of_sale, sale_price)) +
  geom_point()

housingdata$date_of_sale <- NULL

housing_imp_final <- mice::mice(housing_imp_1, method = "polyreg")

X <- complete(housing_imp_final, sample(1:5, size = 1)) %>%
  cbind(M)



D <- cbind(X, sale_price)

X_reg <- D %>%
  select(-WorkerId, -community_district_num, -zip_code, -sale_price)

n <- nrow(D)

train_indices <- sample(1:n, size = 0.8 * n, replace = FALSE)
test_indices <- setdiff(1:n, train_indices)

testthat::expect_equal(sort(c(train_indices, test_indices)), 1:n)

X_train <- X_reg[train_indices, ]
y_train <- sale_price[train_indices]
X_test <- X_reg[test_indices, ]
y_test <- sale_price[test_indices]

D_train <- cbind(X_train, y_train)
```

```
D_test <- cbind(X_test, y_test)

reg <- lm(y_train ~ ., data = D_train)
summary(reg)

lin_oos_rmse <- sqrt(mean((y_test - predict(reg, X_test))^2))
oos_rsq <- 1 - sum((y_test - predict(reg, X_test))^2) / sum((y_test - mean(y_test))^2)

options(java.parameters = "-Xmx4000m")
tree_reg <- YARFCART(X_train,
                     y_train,
                     bootstrap_indices = 1:nrow(X_train))

part_reg <- rpart(sale_price ~ ., data = D, method = "anova")

printcp(part_reg)
plot(part_reg, uniform = TRUE)
text(part_reg, use.n=TRUE, all=TRUE, cex=.8)

illustrate_trees(tree_reg, max_depth = 4)

tree_rmse <- sqrt(mean((y_test - predict(tree_reg, X_test))^2))

rf_mod <- YARF(X_train, y_train, num_trees = 1000)

print(c("Random Forest RMSE:", rf_mod$rmse_oob))
print(c("Random Forest Rsq:" , rf_mod$pseudo_rsq_oob))

sqrt(mean((y_test - predict(rf_mod, X_test))^2))

predict(rf_mod, X_test[1, ])
y_test[1]

tree_reg$rmse_oob
rdresids <- cbind.data.frame(y_test, resids <- y_test - predict(rf_mod, X_test))

ggplot(data = rdresids, aes(y_test, resids)) +
  geom_point()

library(mlr3)
```