Predicting Housing Prices

Kyle Brewster

2022-05-30

01 - Briefly explain why being able to predict whether my model underpredicted a house’s price means that your model “beats” my model.

If I am able to predict if the model is under-predicting, then that means I would grouped the data in very similar groups as the model that predicted `undervalued`. Along the way, I will also gather an idea of how confident the models’ prediction of `undervalued` is. With this information, by the end of my modeling I will have (in theory) predicted whether an observation is `undervalued` by its defined characteristics, but also would have an idea of “how much” and “in which direction” the difference between the actual and suggested validation of an observation (i.e. 3>1)

02 - Use two different models to predict undervalued

```
housing = read.csv("final-data.csv")
```

Since we are attempting to predict a T/F value, we will be using methods of classification.

To get an overview of the data

```
# For handy viewing from environment
skimr::skim(housing) -> skim_df
skimr::skim(housing)
```

Data summary

Name	housing
Number of rows	1460
Number of columns	81
_______________________	
Column type frequency:	
character	43
logical	1
numeric	37
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ms_zoning	0	1.00	2	7	0	5	0
street	0	1.00	4	4	0	2	0
alley	1369	0.06	4	4	0	2	0
lot_shape	0	1.00	3	3	0	4	0
land_contour	0	1.00	3	3	0	4	0
utilities	0	1.00	6	6	0	2	0
lot_config	0	1.00	3	7	0	5	0
land_slope	0	1.00	3	3	0	3	0
neighborhood	0	1.00	5	7	0	25	0
condition1	0	1.00	4	6	0	9	0
condition2	0	1.00	4	6	0	8	0
bldg_type	0	1.00	4	6	0	5	0
house_style	0	1.00	4	6	0	8	0
roof_style	0	1.00	3	7	0	6	0
roof_matl	0	1.00	4	7	0	8	0
exterior1st	0	1.00	5	7	0	15	0
exterior2nd	0	1.00	5	7	0	16	0
mas_vnr_type	8	0.99	4	7	0	4	0
exter_qual	0	1.00	2	2	0	4	0
exter_cond	0	1.00	2	2	0	5	0
foundation	0	1.00	4	6	0	6	0
bsmt_qual	37	0.97	2	2	0	4	0
bsmt_cond	37	0.97	2	2	0	4	0
bsmt_exposure	38	0.97	2	2	0	4	0
bsmt_fin_type1	37	0.97	3	3	0	6	0
bsmt_fin_type2	38	0.97	3	3	0	6	0
heating	0	1.00	4	5	0	6	0
heating_qc	0	1.00	2	2	0	5	0
central_air	0	1.00	1	1	0	2	0
electrical	1	1.00	3	5	0	5	0
kitchen_qual	0	1.00	2	2	0	4	0
functional	0	1.00	3	4	0	7	0
fireplace_qu	690	0.53	2	2	0	5	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
garage_type	81	0.94	6	7	0	6	0
garage_finish	81	0.94	3	3	0	3	0
garage_qual	81	0.94	2	2	0	5	0
garage_cond	81	0.94	2	2	0	5	0
paved_drive	0	1.00	1	1	0	3	0
pool_qc	1453	0.00	2	2	0	3	0
fence	1179	0.19	4	5	0	4	0
misc_feature	1406	0.04	4	4	0	4	0
sale_type	0	1.00	2	5	0	9	0
sale_condition	0	1.00	6	7	0	6	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
undervalued	0	1	0.48	FAL: 754, TRU: 706

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1.00	730.50	421.61	1	365.75	730.5	1095.25	1460	
ms_sub_class	0	1.00	56.90	42.30	20	20.00	50.0	70.00	190	
lot_frontage	259	0.82	70.05	24.28	21	59.00	69.0	80.00	313	
lot_area	0	1.00	10516.83	9981.26	1300	7553.50	9478.5	11601.50	215245	
overall_qual	0	1.00	6.10	1.38	1	5.00	6.0	7.00	10	
overall_cond	0	1.00	5.58	1.11	1	5.00	5.0	6.00	9	
year_built	0	1.00	1971.27	30.20	1872	1954.00	1973.0	2000.00	2010	
year_remod_add	0	1.00	1984.87	20.65	1950	1967.00	1994.0	2004.00	2010	
mas_vnr_area	8	0.99	103.69	181.07	0	0.00	0.0	166.00	1600	
bsmt_fin_sf1	0	1.00	443.64	456.10	0	0.00	383.5	712.25	5644	
bsmt_fin_sf2	0	1.00	46.55	161.32	0	0.00	0.0	0.00	1474	
bsmt_unf_sf	0	1.00	567.24	441.87	0	223.00	477.5	808.00	2336	
total_bsmt_sf	0	1.00	1057.43	438.71	0	795.75	991.5	1298.25	6110	
x1st_flr_sf	0	1.00	1162.63	386.59	334	882.00	1087.0	1391.25	4692	
x2nd_flr_sf	0	1.00	346.99	436.53	0	0.00	0.0	728.00	2065	
low_qual_fin_sf	0	1.00	5.84	48.62	0	0.00	0.0	0.00	572	
gr_liv_area	0	1.00	1515.46	525.48	334	1129.50	1464.0	1776.75	5642	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
bsmt_full_bath	0	1.00	0.43	0.52	0	0.00	0.0	1.00	3	
bsmt_half_bath	0	1.00	0.06	0.24	0	0.00	0.0	0.00	2	
full_bath	0	1.00	1.57	0.55	0	1.00	2.0	2.00	3	
half_bath	0	1.00	0.38	0.50	0	0.00	0.0	1.00	2	
bedroom_abv_gr	0	1.00	2.87	0.82	0	2.00	3.0	3.00	8	
kitchen_abv_gr	0	1.00	1.05	0.22	0	1.00	1.0	1.00	3	
tot_rms_abv_grd	0	1.00	6.52	1.63	2	5.00	6.0	7.00	14	
fireplaces	0	1.00	0.61	0.64	0	0.00	1.0	1.00	3	
garage_yr_blt	81	0.94	1978.51	24.69	1900	1961.00	1980.0	2002.00	2010	
garage_cars	0	1.00	1.77	0.75	0	1.00	2.0	2.00	4	
garage_area	0	1.00	472.98	213.80	0	334.50	480.0	576.00	1418	
wood_deck_sf	0	1.00	94.24	125.34	0	0.00	0.0	168.00	857	
open_porch_sf	0	1.00	46.66	66.26	0	0.00	25.0	68.00	547	
enclosed_porch	0	1.00	21.95	61.12	0	0.00	0.0	0.00	552	
x3ssn_porch	0	1.00	3.41	29.32	0	0.00	0.0	0.00	508	
screen_porch	0	1.00	15.06	55.76	0	0.00	0.0	0.00	480	
pool_area	0	1.00	2.76	40.18	0	0.00	0.0	0.00	738	
misc_val	0	1.00	43.49	496.12	0	0.00	0.0	0.00	15500	
mo_sold	0	1.00	6.32	2.70	1	5.00	6.0	8.00	12	
yr_sold	0	1.00	2007.82	1.33	2006	2007.00	2008.0	2009.00	2010	

Looking at the description of the data, we can see that a `NA` value is assigned to observations that do not possess the amenity described by the variable. Thinking about `bsmt_qual` for example, a house without a basement would have a `NA` value for the variable and would not necessarily suggest a missing/incomplete response (although the `complete_rate` would need to be considered at the same time).

Only two numeric variables are missing values while several while several of the character variables are also missing data. We can formally call all variables with missing values with the following commands:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
skim_df %>% filter(skim_type=="numeric" & n_missing>0)  -> num_na
skim_df %>% filter(skim_type=="character" & n_missing>0)-> char_na
num_na
```

Data summary

Name	housing
Number of rows	1460
Number of columns	81
_______________________	
Column type frequency:	
numeric	3
________________________	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
lot_frontage	259	0.82	70.05	24.28	21	59	69	80	313	
mas_vnr_area	8	0.99	103.69	181.07	0	0	0	166	1600	
garage_yr_blt	81	0.94	1978.51	24.69	1900	1961	1980	2002	2010	

```
char_na
```

Data summary

Name	housing
Number of rows	1460
Number of columns	81
_______________________	
Column type frequency:	
character	16
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
alley	1369	0.06	4	4	0	2	0
mas_vnr_type	8	0.99	4	7	0	4	0
bsmt_qual	37	0.97	2	2	0	4	0
bsmt_cond	37	0.97	2	2	0	4	0
bsmt_exposure	38	0.97	2	2	0	4	0
bsmt_fin_type1	37	0.97	3	3	0	6	0
bsmt_fin_type2	38	0.97	3	3	0	6	0
electrical	1	1.00	3	5	0	5	0
fireplace_qu	690	0.53	2	2	0	5	0
garage_type	81	0.94	6	7	0	6	0
garage_finish	81	0.94	3	3	0	3	0
garage_qual	81	0.94	2	2	0	5	0
garage_cond	81	0.94	2	2	0	5	0
pool_qc	1453	0.00	2	2	0	3	0
fence	1179	0.19	4	5	0	4	0
misc_feature	1406	0.04	4	4	0	4	0

Even though the `complete_rate` for a few of these values is small enough to perhaps consider removing from the data in some contexts, we must remember that such is to be expected since not all houses having a pool, fencing, a fireplace, etc. We can also consider similar logic for the numeric variables. Not all houses have garages, streets connected to the property, or masonry veneer areas that can be quantified since they do not exist. Therefore we will assume that a missing value for the numeric variables with missing values indicates the observation does not possess the given feature and will replace those values with `zero`

Cleaning

After looking at the data, there are a few things we should change before modeling:

- Convert variables from character class to factors
- Replace `NA` values
- Add log transformation variables
- Normalizing data (with min/max scaling)

```r
# Including again to have single chunk to run for cleaning/wrangling
housing = read.csv("final-data.csv")
# For comparing original and modified data frames
clean_house <- housing
# Loading packages

pacman::p_load(  # package manager
   dplyr,        # for wrangling/cleaning/syntax
   magrittr,     # le pipe
   tidyverse,    # modeling
   caret         # also modeling
)

# Converting character variables to factors
clean_house[sapply(clean_house, is.character)] <- lapply(
   clean_house[sapply(clean_house, is.character)], as.factor)

# Removing NA's for numeric and factor variables
clean_house %<>% mutate(
       across(where(is.numeric),~replace_na(.,0)),
       across(where(is.factor),~fct_explicit_na(.,'none')))

# Define min-max normalization function
min_max_norm <- function(x) {
    (x - min(x)) / (max(x) - min(x))}

# Applying function to all non-factor variables
clean_house %<>% mutate_if(is.integer, min_max_norm) %>%
   mutate(id = seq(nrow(.)),
          undervalued = as.factor(undervalued))

# Splitting into training and testing sets by 80-20 splitting
set.seed(123)
train = clean_house %>% mutate(undervalued = as.factor(undervalued)) %>%
   sample_frac(0.8)
test  = anti_join(clean_house, train, by = 'id')

# Dropping ID variable
train %<>% select(-c("id"))
test %<>% select(-c("id"))
```

If desired, can also run the code below to get overview of data to check for other issues before moving forward (should have all numeric and factor variable classes at this point)

```r
str(train)
```

Modeling

Binary Logistic Regression

Let’s take a look at how our predictions would perform if only using regression methods are used to make predictions, first using all variables in the data.

```
# GLM model using all variables
glm_mod_all = glm(
   undervalued ~.,
   data = train,
   family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_mod_all)
```

```
##
## Call:
## glm(formula = undervalued ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.06390  -0.71949  -0.00022   0.71026   2.90806
##
## Coefficients: (10 not defined because of singularities)
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -3.276e+02  3.737e+04  -0.009 0.993007
## ms_sub_class           5.253e+00  2.139e+00   2.456 0.014061 *
## ms_zoningFV            1.166e+00  1.883e+00   0.619 0.535967
## ms_zoningRH            2.254e+00  1.982e+00   1.137 0.255343
## ms_zoningRL            6.449e-01  1.712e+00   0.377 0.706327
## ms_zoningRM            4.738e-02  1.642e+00   0.029 0.976973
## lot_frontage          -9.176e-01  9.887e-01  -0.928 0.353362
## lot_area               2.289e+01  4.800e+00   4.769 1.85e-06 ***
## streetPave             3.032e+01  1.195e+03   0.025 0.979763
## alleyPave              4.068e-01  9.098e-01   0.447 0.654779
## alleynone              8.510e-01  6.784e-01   1.254 0.209711
## lot_shapeIR2           7.342e-01  6.167e-01   1.190 0.233891
## lot_shapeIR3           1.206e+00  1.567e+00   0.770 0.441391
## lot_shapeReg           1.710e-01  2.190e-01   0.781 0.434747
## land_contourHLS       -2.906e-01  7.989e-01  -0.364 0.716011
## land_contourLow       -2.600e+00  9.512e-01  -2.734 0.006258 **
## land_contourLvl       -7.018e-01  5.663e-01  -1.239 0.215234
## utilitiesNoSeWa       -1.229e+01  3.956e+03  -0.003 0.997521
## lot_configCulDSac     -1.874e+00  4.819e-01  -3.888 0.000101 ***
## lot_configFR2         -1.510e+00  5.396e-01  -2.798 0.005148 **
## lot_configFR3         -1.709e+01  2.775e+03  -0.006 0.995087
## lot_configInside      -2.528e-01  2.439e-01  -1.036 0.300074
## land_slopeMod          1.256e+00  6.066e-01   2.071 0.038356 *
## land_slopeSev         -3.870e+00  2.184e+00  -1.772 0.076370 .
## neighborhoodBlueste   -1.635e+01  3.956e+03  -0.004 0.996702
## neighborhoodBrDale    -1.364e+00  1.422e+00  -0.960 0.337304
## neighborhoodBrkSide   -9.250e-01  1.292e+00  -0.716 0.474100
## neighborhoodClearCr   -3.335e+00  1.375e+00  -2.426 0.015286 *
## neighborhoodCollgCr   -9.551e-01  9.175e-01  -1.041 0.297885
## neighborhoodCrawfor    1.709e+00  1.186e+00   1.441 0.149633
## neighborhoodEdwards   -1.293e+00  1.052e+00  -1.229 0.219104
## neighborhoodGilbert   -4.685e-01  9.632e-01  -0.486 0.626675
## neighborhoodIDOTRR    -1.071e+00  1.473e+00  -0.727 0.467283
## neighborhoodMeadowV   -2.519e+00  1.461e+00  -1.724 0.084650 .
## neighborhoodMitchel   -3.493e+00  1.095e+00  -3.189 0.001428 **
## neighborhoodNAmes     -1.863e+00  1.012e+00  -1.841 0.065656 .
## neighborhoodNoRidge    2.980e+00  1.090e+00   2.733 0.006269 **
## neighborhoodNPkVill    1.625e+01  2.187e+03   0.007 0.994070
## neighborhoodNridgHt   -2.268e-01  9.525e-01  -0.238 0.811798
## neighborhoodNWAmes    -1.036e+00  1.020e+00  -1.015 0.309912
## neighborhoodOldTown   -2.986e-01  1.333e+00  -0.224 0.822754
## neighborhoodSawyer    -1.998e+00  1.043e+00  -1.916 0.055420 .
## neighborhoodSawyerW   -2.128e+00  1.029e+00  -2.069 0.038591 *
## neighborhoodSomerst    3.075e-01  1.100e+00   0.280 0.779826
## neighborhoodStoneBr    3.170e+00  1.115e+00   2.843 0.004475 **
## neighborhoodSWISU     -3.625e+00  1.627e+00  -2.227 0.025919 *
```

```
## neighborhoodTimber    -1.623e+00  1.075e+00  -1.510 0.131005
## neighborhoodVeenker   -7.997e-01  1.349e+00  -0.593 0.553431
## condition1Feedr        2.357e-01  7.799e-01   0.302 0.762505
## condition1Norm        -7.175e-01  6.748e-01  -1.063 0.287658
## condition1PosA        -1.344e-01  1.556e+00  -0.086 0.931182
## condition1PosN         1.004e+00  1.069e+00   0.939 0.347820
## condition1RRAe        -2.279e+00  1.317e+00  -1.731 0.083539 .
## condition1RRAn         9.407e-01  1.027e+00   0.916 0.359891
## condition1RRNe         1.491e+00  2.063e+00   0.722 0.470070
## condition1RRNn         2.535e-01  2.048e+00   0.124 0.901462
## condition2Feedr        2.226e-01  3.958e+00   0.056 0.955150
## condition2Norm         1.554e+00  3.284e+00   0.473 0.636014
## condition2PosA         5.329e+00  5.595e+03   0.001 0.999240
## condition2PosN        -2.184e+01  2.277e+03  -0.010 0.992345
## condition2RRAn        -1.937e+01  3.956e+03  -0.005 0.996093
## condition2RRNn         1.762e+01  3.956e+03   0.004 0.996445
## bldg_type2fmCon       -1.759e+00  1.980e+00  -0.889 0.374164
## bldg_typeDuplex       -3.454e+00  1.324e+00  -2.608 0.009102 **
## bldg_typeTwnhs        -3.685e+00  1.486e+00  -2.480 0.013151 *
## bldg_typeTwnhsE       -2.461e+00  1.308e+00  -1.881 0.059965 .
## house_style1.5Unf      5.821e-01  1.435e+00   0.406 0.684966
## house_style1Story      2.872e-01  6.433e-01   0.446 0.655328
## house_style2.5Fin     -1.784e+01  1.431e+03  -0.012 0.990058
## house_style2.5Unf      4.718e-01  1.332e+00   0.354 0.723233
## house_style2Story     -6.294e-02  5.072e-01  -0.124 0.901241
## house_styleSFoyer     -8.474e-02  9.172e-01  -0.092 0.926388
## house_styleSLvl        4.004e-01  7.911e-01   0.506 0.612758
## overall_qual          -1.099e+01  1.487e+00  -7.388 1.49e-13 ***
## overall_cond           1.134e-01  1.025e+00   0.111 0.911959
## year_built             3.340e+00  1.727e+00   1.934 0.053156 .
## year_remod_add         4.541e-01  4.697e-01   0.967 0.333713
## roof_styleGable       -1.558e+01  3.956e+03  -0.004 0.996858
## roof_styleGambrel     -1.601e+01  3.956e+03  -0.004 0.996770
## roof_styleHip         -1.601e+01  3.956e+03  -0.004 0.996772
## roof_styleMansard     -9.735e+00  3.956e+03  -0.002 0.998037
## roof_styleShed         1.247e+01  5.595e+03   0.002 0.998221
## roof_matlCompShg      -9.804e+01  3.405e+04  -0.003 0.997703
## roof_matlMetal        -1.245e+02  3.451e+04  -0.004 0.997122
## roof_matlRoll         -1.110e+02  3.428e+04  -0.003 0.997415
## roof_matlTar&Grv      -1.135e+02  3.428e+04  -0.003 0.997357
## roof_matlWdShake      -1.009e+02  3.405e+04  -0.003 0.997636
## roof_matlWdShngl      -8.174e+01  3.409e+04  -0.002 0.998087
## exterior1stAsphShn    -1.644e+01  3.956e+03  -0.004 0.996685
## exterior1stBrkComm     3.191e+01  2.938e+03   0.011 0.991335
## exterior1stBrkFace     1.696e+01  1.962e+03   0.009 0.993102
## exterior1stCBlock     -2.042e+01  3.956e+03  -0.005 0.995882
## exterior1stCemntBd     1.106e+01  1.962e+03   0.006 0.995502
## exterior1stHdBoard     1.466e+01  1.962e+03   0.007 0.994039
## exterior1stImStucc     2.967e+01  4.416e+03   0.007 0.994640
## exterior1stMetalSd     1.431e+01  1.962e+03   0.007 0.994178
## exterior1stPlywood     1.451e+01  1.962e+03   0.007 0.994100
## exterior1stStone       2.773e+01  4.416e+03   0.006 0.994990
## exterior1stStucco      1.411e+01  1.962e+03   0.007 0.994261
## exterior1stVinylSd     1.499e+01  1.962e+03   0.008 0.993904
## exterior1stWd Sdng     1.316e+01  1.962e+03   0.007 0.994646
## exterior1stWdShing     1.534e+01  1.962e+03   0.008 0.993761
## exterior2ndAsphShn            NA         NA      NA       NA
```

```
## exterior2ndBrk Cmn    -3.124e+01  2.938e+03  -0.011 0.991516
## exterior2ndBrkFace    -1.526e+01  1.962e+03  -0.008 0.993793
## exterior2ndCBlock             NA         NA      NA       NA
## exterior2ndCmentBd    -1.025e+01  1.962e+03  -0.005 0.995831
## exterior2ndHdBoard    -1.444e+01  1.962e+03  -0.007 0.994129
## exterior2ndImStucc    -1.484e+01  1.962e+03  -0.008 0.993966
## exterior2ndMetalSd    -1.388e+01  1.962e+03  -0.007 0.994354
## exterior2ndPlywood    -1.360e+01  1.962e+03  -0.007 0.994468
## exterior2ndStone      -1.479e+01  1.962e+03  -0.008 0.993986
## exterior2ndStucco     -1.211e+01  1.962e+03  -0.006 0.995075
## exterior2ndVinylSd    -1.432e+01  1.962e+03  -0.007 0.994176
## exterior2ndWd Sdng    -1.257e+01  1.962e+03  -0.006 0.994887
## exterior2ndWd Shng    -1.538e+01  1.962e+03  -0.008 0.993743
## mas_vnr_typeBrkFace    1.441e+00  8.968e-01   1.607 0.108056
## mas_vnr_typeNone       5.383e-01  9.010e-01   0.597 0.550210
## mas_vnr_typeStone      1.202e+00  9.522e-01   1.263 0.206660
## mas_vnr_typenone      -1.649e+00  1.577e+00  -1.046 0.295709
## mas_vnr_area          -4.149e+00  1.316e+00  -3.152 0.001622 **
## exter_qualFa           2.368e-01  2.034e+00   0.116 0.907293
## exter_qualGd          -1.055e+00  6.232e-01  -1.694 0.090338 .
## exter_qualTA           2.654e-01  6.941e-01   0.382 0.702170
## exter_condFa          -1.562e+01  3.956e+03  -0.004 0.996849
## exter_condGd          -1.583e+01  3.956e+03  -0.004 0.996807
## exter_condPo          -3.429e+01  5.595e+03  -0.006 0.995110
## exter_condTA          -1.544e+01  3.956e+03  -0.004 0.996886
## foundationCBlock       9.155e-01  4.982e-01   1.837 0.066148 .
## foundationPConc        1.709e-01  5.492e-01   0.311 0.755658
## foundationSlab        -2.828e-01  1.819e+00  -0.155 0.876432
## foundationStone       -1.053e+00  1.864e+00  -0.565 0.572063
## foundationWood        -1.652e+00  1.838e+00  -0.899 0.368855
## bsmt_qualFa           -1.722e+00  9.085e-01  -1.896 0.058018 .
## bsmt_qualGd           -5.530e-01  4.549e-01  -1.216 0.224116
## bsmt_qualTA           -4.060e-01  5.667e-01  -0.716 0.473815
## bsmt_qualnone          1.968e+01  3.956e+03   0.005 0.996031
## bsmt_condGd           -9.703e-01  8.049e-01  -1.205 0.228048
## bsmt_condPo            2.265e+01  3.956e+03   0.006 0.995433
## bsmt_condTA           -3.868e-01  6.282e-01  -0.616 0.538124
## bsmt_condnone                 NA         NA      NA       NA
## bsmt_exposureGd       -1.057e+00  4.345e-01  -2.432 0.015011 *
## bsmt_exposureMn       -2.288e-01  4.081e-01  -0.561 0.575013
## bsmt_exposureNo        2.101e-01  2.947e-01   0.713 0.475865
## bsmt_exposurenone     -1.752e+01  3.956e+03  -0.004 0.996467
## bsmt_fin_type1BLQ     -3.960e-01  3.818e-01  -1.037 0.299610
## bsmt_fin_type1GLQ      9.821e-02  3.504e-01   0.280 0.779277
## bsmt_fin_type1LwQ     -9.037e-02  5.341e-01  -0.169 0.865654
## bsmt_fin_type1Rec      1.707e-01  4.075e-01   0.419 0.675236
## bsmt_fin_type1Unf      1.702e+00  4.291e-01   3.967 7.28e-05 ***
## bsmt_fin_type1none            NA         NA      NA       NA
## bsmt_fin_sf1           3.108e+01  4.880e+00   6.368 1.91e-10 ***
## bsmt_fin_type2BLQ     -2.654e+00  1.115e+00  -2.380 0.017320 *
## bsmt_fin_type2GLQ      1.227e+00  1.387e+00   0.884 0.376441
## bsmt_fin_type2LwQ     -2.066e+00  1.069e+00  -1.931 0.053429 .
## bsmt_fin_type2Rec     -1.823e+00  1.050e+00  -1.735 0.082653 .
## bsmt_fin_type2Unf     -1.877e+00  1.126e+00  -1.667 0.095534 .
## bsmt_fin_type2none            NA         NA      NA       NA
## bsmt_fin_sf2           5.648e+00  1.927e+00   2.931 0.003380 **
## bsmt_unf_sf            6.918e+00  1.858e+00   3.723 0.000197 ***
```

```
## total_bsmt_sf                NA         NA      NA       NA
## heatingGasA          -1.254e+01  3.956e+03  -0.003 0.997470
## heatingGasW          -1.436e+01  3.956e+03  -0.004 0.997103
## heatingGrav          -1.209e+01  3.956e+03  -0.003 0.997563
## heatingOthW          -2.547e+01  4.803e+03  -0.005 0.995769
## heatingWall          -1.191e+01  3.956e+03  -0.003 0.997598
## heating_qcFa          2.795e-01  6.874e-01   0.407 0.684318
## heating_qcGd         -6.900e-01  2.864e-01  -2.409 0.015979 *
## heating_qcPo         -1.458e+01  3.956e+03  -0.004 0.997059
## heating_qcTA          2.524e-03  2.827e-01   0.009 0.992875
## central_airY         -2.904e-01  6.529e-01  -0.445 0.656425
## electricalFuseF      -8.145e-01  9.155e-01  -0.890 0.373686
## electricalFuseP      -2.378e+01  2.477e+03  -0.010 0.992340
## electricalSBrkr      -2.521e-01  4.465e-01  -0.565 0.572308
## electricalnone        1.600e+01  3.956e+03   0.004 0.996773
## x1st_flr_sf          -5.360e+00  3.748e+00  -1.430 0.152702
## x2nd_flr_sf           6.689e-01  1.627e+00   0.411 0.681047
## low_qual_fin_sf      -2.248e+00  1.628e+00  -1.381 0.167259
## gr_liv_area                  NA         NA      NA       NA
## bsmt_full_bath       -2.936e+00  8.218e-01  -3.573 0.000353 ***
## bsmt_half_bath       -1.642e+00  8.679e-01  -1.892 0.058428 .
## full_bath            -2.507e+00  9.462e-01  -2.649 0.008071 **
## half_bath            -1.116e-01  5.714e-01  -0.195 0.845161
## bedroom_abv_gr       -1.915e+00  1.600e+00  -1.197 0.231310
## kitchen_abv_gr       -1.435e+00  3.517e+00  -0.408 0.683275
## kitchen_qualFa       -2.508e+00  9.562e-01  -2.623 0.008718 **
## kitchen_qualGd       -1.025e+00  4.742e-01  -2.161 0.030685 *
## kitchen_qualTA       -1.422e+00  5.317e-01  -2.675 0.007475 **
## tot_rms_abv_grd       9.037e-01  1.593e+00   0.567 0.570591
## functionalMaj2       -1.763e+01  1.629e+03  -0.011 0.991364
## functionalMin1        1.549e-01  1.274e+00   0.122 0.903216
## functionalMin2        1.048e+00  1.314e+00   0.797 0.425478
## functionalMod        -7.276e-01  1.476e+00  -0.493 0.622005
## functionalSev        -1.558e+01  3.956e+03  -0.004 0.996858
## functionalTyp         1.181e-01  1.106e+00   0.107 0.914961
## fireplaces           -1.353e+00  1.118e+00  -1.211 0.226080
## fireplace_quFa        1.280e-01  9.171e-01   0.140 0.888981
## fireplace_quGd       -1.235e-01  6.787e-01  -0.182 0.855593
## fireplace_quPo        5.784e-01  9.969e-01   0.580 0.561765
## fireplace_quTA       -4.158e-01  7.180e-01  -0.579 0.562538
## fireplace_qunone     -4.572e-02  8.390e-01  -0.055 0.956536
## garage_typeAttchd     1.681e+00  1.744e+00   0.964 0.335223
## garage_typeBasment    1.345e+00  1.977e+00   0.681 0.496126
## garage_typeBuiltIn    1.469e+00  1.788e+00   0.822 0.411041
## garage_typeCarPort    8.613e-01  2.311e+00   0.373 0.709342
## garage_typeDetchd     1.712e+00  1.736e+00   0.986 0.324233
## garage_typenone       9.011e+00  2.700e+03   0.003 0.997337
## garage_yr_blt         2.409e+01  1.950e+01   1.235 0.216823
## garage_finishRFn      2.580e-01  2.699e-01   0.956 0.339213
## garage_finishUnf      1.517e-01  3.383e-01   0.448 0.653846
## garage_finishnone            NA         NA      NA       NA
## garage_cars          -3.817e+00  1.233e+00  -3.096 0.001961 **
## garage_area           3.412e+00  1.576e+00   2.165 0.030399 *
## garage_qualFa        -2.550e+01  4.541e+03  -0.006 0.995520
## garage_qualGd        -2.261e+01  4.541e+03  -0.005 0.996028
## garage_qualPo        -1.489e+01  4.925e+03  -0.003 0.997589
## garage_qualTA        -2.378e+01  4.541e+03  -0.005 0.995821
```

```
## garage_qualnone              NA         NA      NA       NA
## garage_condFa         5.994e+00  5.283e+03   0.001 0.999095
## garage_condGd        -1.292e+01  5.396e+03  -0.002 0.998089
## garage_condPo         8.946e+00  5.283e+03   0.002 0.998649
## garage_condTA         6.596e+00  5.283e+03   0.001 0.999004
## garage_condnone              NA         NA      NA       NA
## paved_driveP         -2.476e-01  8.771e-01  -0.282 0.777760
## paved_driveY         -2.333e-01  5.221e-01  -0.447 0.655037
## wood_deck_sf         -1.002e-01  7.091e-01  -0.141 0.887613
## open_porch_sf         1.556e+00  8.841e-01   1.760 0.078452 .
## enclosed_porch        1.718e+00  1.065e+00   1.614 0.106462
## x3ssn_porch           2.481e+00  1.484e+00   1.672 0.094607 .
## screen_porch          2.758e+00  9.139e-01   3.018 0.002547 **
## pool_area             5.202e+02  9.602e+04   0.005 0.995677
## pool_qcFa            -6.444e+01  1.567e+04  -0.004 0.996718
## pool_qcGd            -1.069e+02  2.707e+04  -0.004 0.996849
## pool_qcnone           3.744e+02  6.947e+04   0.005 0.995701
## fenceGdWo             1.729e+00  7.408e-01   2.334 0.019590 *
## fenceMnPrv            9.479e-01  6.579e-01   1.441 0.149609
## fenceMnWw            -2.829e-01  1.072e+00  -0.264 0.791854
## fencenone             9.841e-01  6.075e-01   1.620 0.105268
## misc_featureOthr      3.048e+01  4.359e+03   0.007 0.994421
## misc_featureShed      5.611e+01  3.956e+03   0.014 0.988684
## misc_featureTenC      1.381e+02  1.813e+04   0.008 0.993924
## misc_featurenone      5.768e+01  3.956e+03   0.015 0.988368
## misc_val              4.284e+01  1.844e+01   2.324 0.020142 *
## mo_sold              -8.836e-02  3.613e-01  -0.245 0.806805
## yr_sold              -1.877e-01  2.801e-01  -0.670 0.502690
## sale_typeCon          1.966e+01  3.956e+03   0.005 0.996034
## sale_typeConLD       -1.824e-01  1.468e+00  -0.124 0.901088
## sale_typeConLI       -9.632e-02  1.476e+00  -0.065 0.947960
## sale_typeConLw       -5.276e-01  1.585e+00  -0.333 0.739176
## sale_typeCWD          3.597e+00  1.876e+00   1.917 0.055244 .
## sale_typeNew         -2.999e+00  1.956e+00  -1.533 0.125252
## sale_typeOth          2.478e+00  1.593e+00   1.556 0.119737
## sale_typeWD           7.296e-02  5.553e-01   0.131 0.895463
## sale_conditionAdjLand  2.148e+01  1.995e+03   0.011 0.991412
## sale_conditionAlloca   9.700e-01  1.407e+00   0.690 0.490465
## sale_conditionFamily   1.874e+00  7.691e-01   2.436 0.014839 *
## sale_conditionNormal   1.490e+00  3.942e-01   3.779 0.000157 ***
## sale_conditionPartial  3.555e+00  1.879e+00   1.892 0.058538 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1614.99  on 1167  degrees of freedom
## Residual deviance:  999.38  on  920  degrees of freedom
## AIC: 1495.4
##
## Number of Fisher Scoring iterations: 16
```

Now let’s narrow down the number of variables used to include only those considered above to be statistically significant.

```r
# Same modeling but only with significant variables
glm_mod_sig = glm(
   undervalued ~
      ms_sub_class+lot_area+land_contour+lot_config+
      bldg_type+overall_qual+mas_vnr_area+bsmt_fin_type1+
      bsmt_fin_sf1+bsmt_fin_sf2+bsmt_unf_sf+bsmt_full_bath+
      kitchen_qual,
   data = train,
   family = "binomial"
)
summary(glm_mod_sig)
```

```
##
## Call:
## glm(formula = undervalued ~ ms_sub_class + lot_area + land_contour +
##     lot_config + bldg_type + overall_qual + mas_vnr_area + bsmt_fin_type1 +
##     bsmt_fin_sf1 + bsmt_fin_sf2 + bsmt_unf_sf + bsmt_full_bath +
##     kitchen_qual, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3715  -1.0735  -0.7088   1.1288   1.8968
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.24396    0.67286   3.335 0.000853 ***
## ms_sub_class          0.84666    0.59610   1.420 0.155510
## lot_area             -1.08436    1.90857  -0.568 0.569930
## land_contourHLS       0.78456    0.44776   1.752 0.079742 .
## land_contourLow       0.27721    0.51219   0.541 0.588350
## land_contourLvl       0.26330    0.31670   0.831 0.405753
## lot_configCulDSac    -0.47828    0.29690  -1.611 0.107204
## lot_configFR2        -0.73615    0.38435  -1.915 0.055455 .
## lot_configFR3       -13.17837  374.94743  -0.035 0.971962
## lot_configInside     -0.14764    0.16256  -0.908 0.363764
## bldg_type2fmCon      -0.21554    0.68297  -0.316 0.752316
## bldg_typeDuplex      -1.45946    0.44556  -3.276 0.001054 **
## bldg_typeTwnhs       -1.17756    0.59442  -1.981 0.047590 *
## bldg_typeTwnhsE      -0.27117    0.38680  -0.701 0.483276
## overall_qual         -4.40082    0.70662  -6.228 4.72e-10 ***
## mas_vnr_area         -0.95832    0.65646  -1.460 0.144334
## bsmt_fin_type1BLQ    -0.05452    0.25160  -0.217 0.828447
## bsmt_fin_type1GLQ    -0.08703    0.22180  -0.392 0.694763
## bsmt_fin_type1LwQ    -0.21270    0.32220  -0.660 0.509148
## bsmt_fin_type1Rec     0.08247    0.25726   0.321 0.748541
## bsmt_fin_type1Unf     0.31652    0.25462   1.243 0.213826
## bsmt_fin_type1none    0.66984    0.54496   1.229 0.219018
## bsmt_fin_sf1          9.01509    1.85870   4.850 1.23e-06 ***
## bsmt_fin_sf2          1.76016    0.68550   2.568 0.010238 *
## bsmt_unf_sf           1.49910    0.64875   2.311 0.020846 *
## bsmt_full_bath       -1.19286    0.52282  -2.282 0.022512 *
## kitchen_qualFa       -2.03516    0.55195  -3.687 0.000227 ***
## kitchen_qualGd       -0.89918    0.28263  -3.181 0.001466 **
## kitchen_qualTA       -1.15107    0.32581  -3.533 0.000411 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1615.0  on 1167  degrees of freedom
## Residual deviance: 1514.1  on 1139  degrees of freedom
## AIC: 1572.1
##
## Number of Fisher Scoring iterations: 12
```

Now let’s adjust the GLM model once more to remove some extra noise

```
glm_mod_fin = glm(
   undervalued ~
     overall_qual + bsmt_fin_sf1 + kitchen_qual + bldg_type,
   family = "binomial",
   data = train)

cbind(
   coef(glm_mod_fin),
   odds_ratio=exp(coef(glm_mod_fin)),
   exp(confint(glm_mod_fin))
)
```

```
##                              odds_ratio       2.5 %      97.5 %
## (Intercept)      2.83164697 16.97339248 6.242159919 47.44852835
## overall_qual    -3.56801540  0.02821179 0.009176981  0.08452687
## bsmt_fin_sf1     2.47021650 11.82500672 2.536111574 56.96379760
## kitchen_qualFa  -2.09642922  0.12289448 0.041739934  0.34609383
## kitchen_qualGd  -1.02566034  0.35855962 0.208902716  0.60830650
## kitchen_qualTA  -1.23559638  0.29066137 0.156725395  0.53070272
## bldg_type2fmCon  0.44222182  1.55616088 0.653857113  3.94345726
## bldg_typeDuplex -1.04574754  0.35142901 0.172328382  0.67928675
## bldg_typeTwnhs  -0.89237553  0.40968139 0.164711039  0.92662089
## bldg_typeTwnhsE  0.06949113  1.07196256 0.696911460  1.64488364
```

To determine what we should have for the cutoff value, we can create multiple classification tables and look at the differences between accuracy, sensitivity, and specificity and then choose a value that minimizes the marginal differences between these measures. We want to minimize the error that the final version of this model has when applied to new data, lest we sacrifice generalization of the model for higher performance in training (i.e. overfitting and the bias-variance tradeoff).

```
# Cutoff Value set to 0.3
train$predprob <- round(fitted(glm_mod_fin),2)
class.table = table(Actual = train$undervalued,
                    Predicted = train$predprob>0.3)
class.table
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE    82  537
##   TRUE     36  513
```

```
# Cutoff Value set to 0.5
train$predprob <- round(fitted(glm_mod_fin),2)
class.table = table(Actual = train$undervalued,
                    Predicted = train$predprob>0.5)
class.table
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE   437  182
##   TRUE    273  276
```

```
# Cutoff Value set to 0.7
train$predprob <- round(fitted(glm_mod_fin),2)
class.table = table(Actual = train$undervalued,
                    Predicted = train$predprob>0.7)
class.table
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE   610    9
##   TRUE    529   20
```

We can see a significant variation in model predictions by adjusting these values. Let's now take a look at some metrics to judge the strength of our model.

Accuracy can be calculated by taking the sum of the correct predictions divided by the total number of observations (i.e. sum(top-left+bottom-right)/n.total.obs)

```
m1_0.3 = (82+513)/1168
m2_0.5 = (437+276)/1168
m3_0.7 = (610+20)/1168
metric = "accuracy"
accuracy_metrics = cbind.data.frame(metric, m1_0.3, m2_0.5, m3_0.7)
```

Sensitivity can be calculated by the number of correct positive assessments of `undervalued` observations by the total number of values predicted to be `undervalued` (i.e. bottom-right/sum(across-bottom))

```
513/(36+513) -> m1_0.3
276/(273+276) -> m2_0.5
20/(529+20) -> m3_0.7
metric = "senstivity"
accuracy_metrics2 = cbind.data.frame(metric, m1_0.3, m2_0.5, m3_0.7)
```

Specificity can be calculated by dividing the accurately predicted observations that were *not* `undervalued` by the total number of values predicted not to be `undervalued` (i.e. top-left/sum(across-top))

```
82/(537+82) -> m1_0.3
437/619 -> m2_0.5
610/619 -> m3_0.7
metric = "specificity"
accuracy_metrics3 = cbind.data.frame(metric, m1_0.3, m2_0.5, m3_0.7)

# Overview of measurement metrics
rbind(accuracy_metrics, accuracy_metrics2, accuracy_metrics3)
```

```
##        metric    m1_0.3    m2_0.5     m3_0.7
## 1    accuracy 0.5094178 0.6104452 0.53938356
## 2  senstivity 0.9344262 0.5027322 0.03642987
## 3 specificity 0.1324717 0.7059774 0.98546042
```

It is important to consider the cutoff value that we use in the modeling because we face a tradeoff between bias and variance when applying the models to the real world or on a testing set. We can see that resulting variation in the table presenting an overview of the metrics above and note a few things:

- The cutoff value of 0.5 has the greatest accuracy of our selected models

- Sensitivity increases significantly for smaller cutoff values
- Specificity increases with higher cutoff values

To save some extra scripting time and potential risk of typos, we can also save this as a template (https://www.lexjansen.com/nesug/nesug10/hl/hl07.pdf) we can use for future calculations, where:

- `TN` = Number of true negative assessments
- `TP` = Number of true positive assessments
- `FP` = Number of false positives
- `FN` = Number of false negatives
- `(TP + FP)` = Total observations with positive assessments
- `(FN + TN)` = Total observations wit negative assessments
- `N = (TP + TN + FP + FN)` = Total number of observations

```
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
```

We could also run the create a model similar to the coding shown below if we wanted to methodologically determine the best variables to select including in our regression models.

```
# GLM modeling using step-wise selection of variables
glm.step <- MASS::stepAIC(glm_mod_all, trace = FALSE,
                          data = train, steps = 1000) # 1000 by default
glm.step$annova
```

Since we know from the results of the preceding chunks, however, we can see that regression might not be the best attempt to make predictions considering the tradeoffs we would have to make and can opt to save the computational time/effort for other models we expect to perform better.

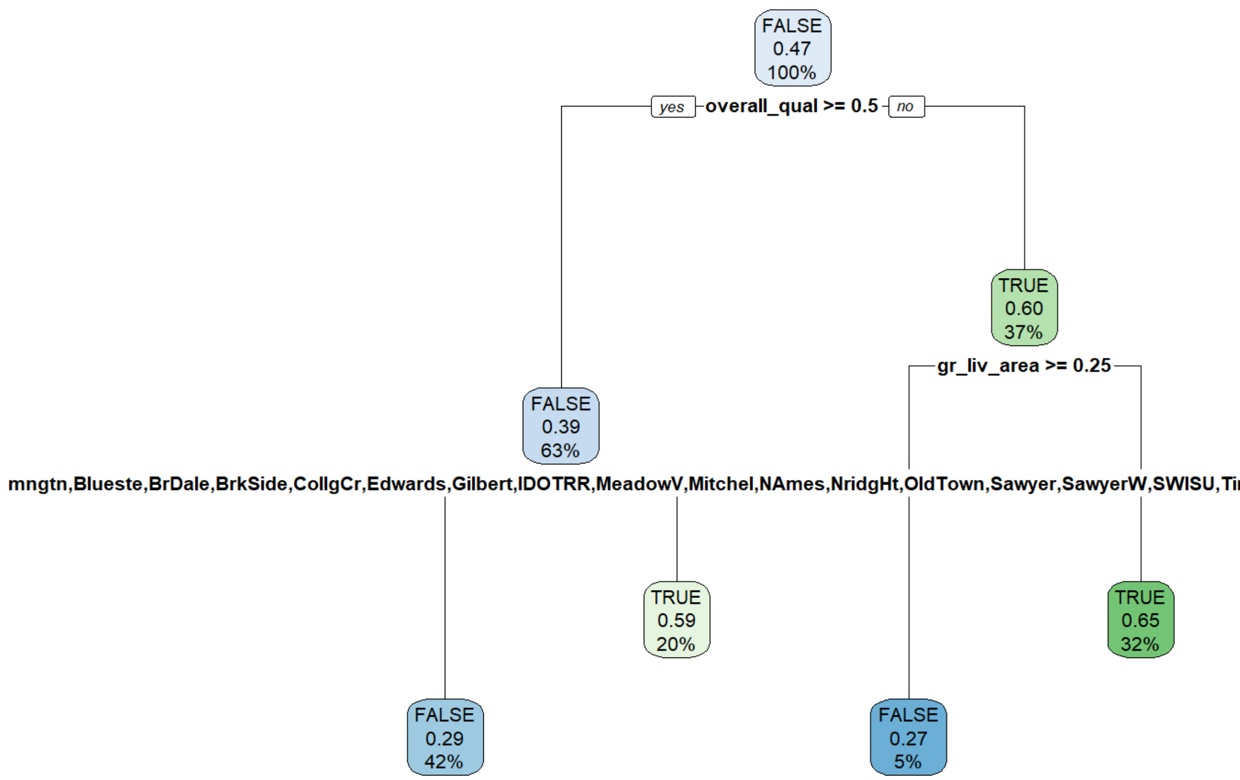
Even though we know that regression might not be the best type of model to use in this context, we can still gain some insight from our calculations. For example, when determining which variables we were going to include, several of the factor variables had individual levels that were considered statistically significant but not the variable as a whole. Perhaps this indicates that there are specific features that carry more weight in home valuation than other feature. If so, then perhaps decision trees can help model out data with greater accuracy

Decision Trees and Random Forest

Let’s see what happens if we algorithmically build a model, this time using decision trees to predict our outcome. algorithmically

```
# Refreshing our training and testing sets
set.seed(123)
train = clean_house %>% mutate(undervalued = as.factor(undervalued)) %>%
   sample_frac(0.8)
test  = anti_join(clean_house, train, by = 'id')

# Dropping ID variable
train %<>% select(-c("id"))
test %<>% select(-c("id"))
```

```r
pacman::p_load(tidymodels,
               rpart.plot)

# Chunk takes ~5 minutes to execute

default_cv = train %>% vfold_cv(v =5)
default_tree = decision_tree(mode ="classification",
                             cost_complexity = tune(),
                             tree_depth = tune()) %>%
               set_engine("rpart")

# Defining recipe
default_recipe = recipe(undervalued ~., data = train)
# Defining workflow
default_flow = workflow() %>%
  add_model(default_tree) %>%
  add_recipe(default_recipe)
# Tuning
default_cv_fit = default_flow %>%
  tune_grid(
    default_cv,
    grid = expand_grid(
      cost_complexity = seq(0, 0.15, by = 0.01),
      tree_depth = c(1,2,5,10),
    ),
    metrics = metric_set(accuracy, roc_auc))
# Fitting and selecting best model
best_flow = default_flow %>%
  finalize_workflow(select_best(default_cv_fit, metric = "accuracy")) %>%
  fit(data = train)
best_tree = best_flow %>% extract_fit_parsnip()
best_tree$fit %>% rpart.plot::rpart.plot(roundint=F)
```




```
# Summary statistics and plotting
printcp(best_tree$fit)
```

```
##
## Classification tree:
## rpart::rpart(formula = ..y ~ ., data = data, cp = ~0, maxdepth = ~2)
##
## Variables actually used in tree construction:
## [1] gr_liv_area  neighborhood overall_qual
##
## Root node error: 549/1168 = 0.47003
##
## n= 1168
##
##         CP nsplit rel error  xerror     xstd
## 1 0.165756      0   1.00000 1.00000 0.031070
## 2 0.078324      1   0.83424 0.90346 0.030770
## 3 0.047359      2   0.75592 0.86885 0.030599
## 4 0.000000      3   0.70856 0.82332 0.030320
```

```
best_tree$fit$variable.importance
```

```
##    neighborhood    overall_qual     gr_liv_area      year_built       bsmt_qual
##       38.579931       25.102234       24.271346        9.810069        9.636950
##     garage_cars tot_rms_abv_grd       ms_zoning     x2nd_flr_sf     x1st_flr_sf
##        9.579243        6.247153        6.040665        5.726557        5.205961
##  bedroom_abv_gr   total_bsmt_sf     exterior2nd     exterior1st      land_slope
##        2.863278        2.082384        1.895110        1.776666        1.421333
##           alley
##        1.184444
```

```
# Predicting values
as.data.frame(predict(best_tree, new_data=train)) -> df1
```

Looking at our first decision tree, there are several things worth considering:

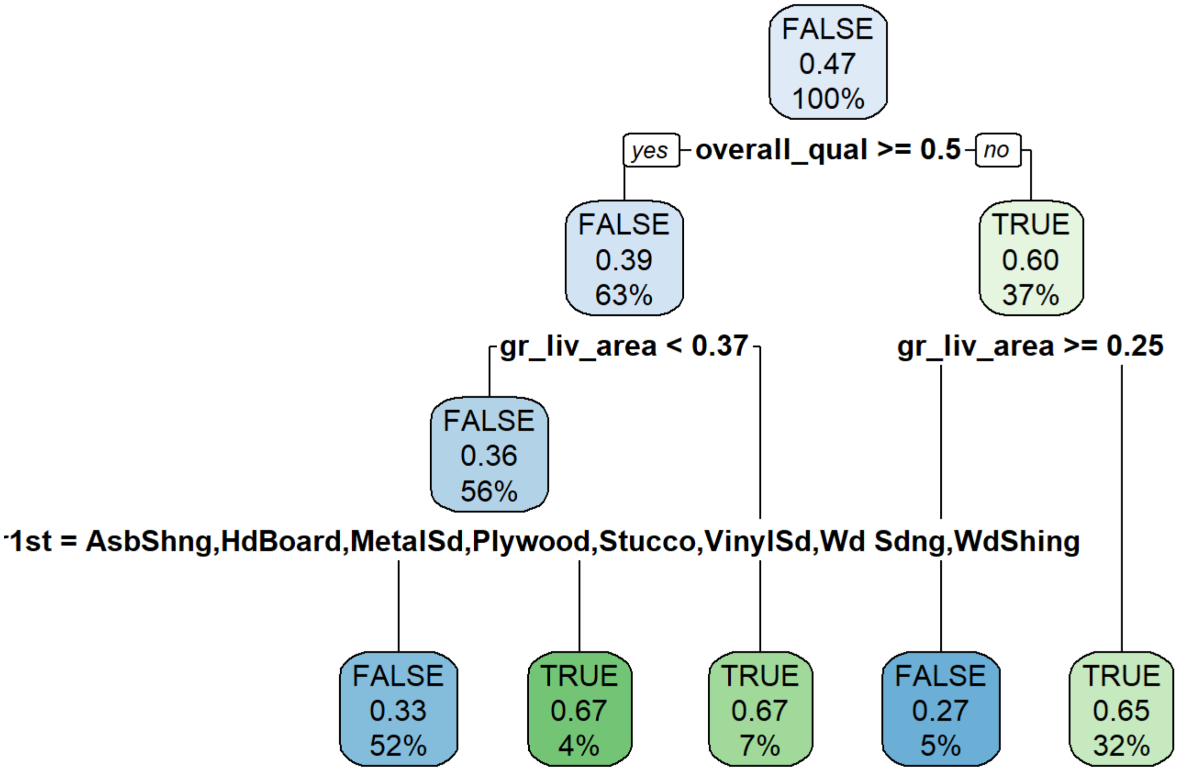
- The `neighborhood` variables was ranked as the greatest variable of importance.
  - This might be good to know if we use this model on future data on housing from the same selection of neighborhoods, but wouldn't perform well on data from other areas where the distinction between neighborhood is less explanatory
- The significant variables are different than those suggested and used in our previous model

For the next trees, we will omit the `neighborhood` variable in an effort to produce a model that is more generalizable. While we could likely develop a way to preserve the information that this variable contributes to the data (like by creating an additional variable(s) with metrics pertaining to the given neighborhood such as crime-rate, median income, average education, neighborhood demographics, etc.), it could provide insight if create a model that cannot look to the neighborhood of an observation.

```
set.seed(123)
train_hold = clean_house %>%
   select(-c("neighborhood")) %>%
   mutate(undervalued = as.factor(undervalued)) %>%
   sample_frac(0.8)
test_hold = anti_join(clean_house, train_hold, by = 'id')

train_hold %<>% select(-c("id"))
test_hold %<>% select(-c("id"))
```

And now to plant another tree

```
default_recipe = recipe(undervalued ~., data = train_hold)
default_flow = workflow() %>%
  add_model(default_tree) %>%
  add_recipe(default_recipe)
default_cv_fit = default_flow %>%
  tune_grid(
    default_cv,
    grid = expand_grid(
      cost_complexity = seq(0, 0.15, by = 0.01),
      tree_depth = c(1,2,5,10),),
    metrics = metric_set(accuracy, roc_auc))
best_flow = default_flow %>%
  finalize_workflow(select_best(default_cv_fit, metric = "accuracy")) %>%
  fit(data = train_hold)
best_tree2 = best_flow %>% extract_fit_parsnip()
best_tree2$fit %>% rpart.plot::rpart.plot(roundint=F)
```




```r
# Summary and plotting
printcp(best_tree2$fit)
```

```
##
## Classification tree:
## rpart::rpart(formula = ..y ~ ., data = data, cp = ~0.02, maxdepth = ~5)
##
## Variables actually used in tree construction:
## [1] exterior1st  gr_liv_area  overall_qual
##
## Root node error: 549/1168 = 0.47003
##
## n= 1168
##
##         CP nsplit rel error  xerror     xstd
## 1 0.165756      0   1.00000 1.00000 0.031070
## 2 0.049180      1   0.83424 0.83424 0.030393
## 3 0.047359      2   0.78506 0.85428 0.030516
## 4 0.029144      3   0.73770 0.81421 0.030257
## 5 0.020000      4   0.70856 0.79417 0.030110
```

```r
best_tree2$fit$variable.importance
```

```
##     gr_liv_area    overall_qual     x2nd_flr_sf tot_rms_abv_grd       full_bath
##      38.1941946      25.5376667      12.4301505      11.4037635      11.2919544
##     exterior1st      year_built       bsmt_qual     garage_cars     exterior2nd
##      10.0149479       9.8100685       9.6369497       9.5792434       6.9669203
##     x1st_flr_sf   total_bsmt_sf  bedroom_abv_gr      condition2      functional
##       6.9248310       3.6293675       2.8632784       0.2177163       0.2177163
##       sale_type
##       0.2177163
```

```
# Predicting values
as.data.frame(predict(best_tree2, new_data=train_hold)) -> df2
```

By removing the `neighborhood` variable, we can note several differences in the resulting model construction. The variable `gr_liv_area` (above-ground living area in squared feet) was used instead of `neighborhood` . We can also see removing the one-column also changed the listed variables of importance that were generated. For example, in the second tree we see `functional` and `sale_type` that do not appear in the first tree and see variables like `year_built` and `ms_zoning` that appear in the first tree but not the second.

From this seemingly small change, we can gather some additional insight into the story that is hiding in our data. Obviously the `neighboorhood` variable must hold some importance since its commission changes the resulting tree, but I believe there is additional information that the `neighboorhood` could carry.

While the name of a neighborhood is nothing more than a logical arrangement of letters and spaces, geographic boundaries could imply several characteristics that would apply to a home being sold within those boundaries. Even if the floor-plan and characteristics of two given homes are 100% similar apart from the neighborhood, there could still be differences that affect sale prices

- *Crime rates*
  - A quick online search could determine reports of crimes for a given geographic area, knowledge of which could affect the price one is willing to pay for a home if house is located in crime-heavy area
- *Public education jurisdiction*
  - Results in distinct differences determined by specific boundaries. Even the side of the street a home is on could determine if their children attend an A-rated school versus a D-rated school
- *Community*
  - Not as black-and-white as education or crime, but could potentially provide some explanatory effect
  - Examples: Willingness to pay more than would otherwise in order to live close to family, friends, cultural significance (e.g. Spanish-speaking), class-status/“prestige”, etc
  - Aforementioned examples could potentially result in consumer willingness to pay a price either higher or lower than they would otherwise accept
- *Presences of Homeowners Association*
  - Because who the hell wants to be told what color they are allowed to paint their house, or if they are able to have a garden on their property that is visible from the street, or how long their grass is “allowed” to grow? This is America! - (unless we are looking at data from another country)

Before moving on to another model, let’s first consider the predictions from the previous trees.

```
df1 %<>% rename(pred_tree = names(.)[1])
# Creating confusion matrix
table(Actual = train$undervalued,
      Predicted = df1$pred_tree)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE   390  229
##   TRUE    160  389
```

```
TN = 390
TP = 389
FP = 229
FN = 160
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_tree1 = rbind(Sensitivity,Specificity,Accuracy)
metrics_tree1
```

```
##                  [,1]
## Sensitivity 0.7085610
## Specificity 0.6300485
## Accuracy    0.6669521
```

```
df2 %<>% rename(pred_tree = names(.)[1])
table(Actual = train_hold$undervalued,
      Predicted = df2$pred_tree)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE   446  173
##   TRUE    216  333
```

```
TN = 446
TP = 333
FP = 173
FN = 216
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_tree2 = rbind(Sensitivity,Specificity,Accuracy)
metrics_tree2
```

```
##                  [,1]
## Sensitivity 0.6065574
## Specificity 0.7205170
## Accuracy    0.6669521
```

Looks like we were able to produce more stable models with decent accuracy (on the training set at least, we will have to see how it performs on testing). If the single decision trees produced these different results, let's see how an ensemble of trees (i.e. random forest) will perform

Random Forest

I saw the values of the training/testing sets when `unique(test$neighborhood)` were both the same, so I figured that it would be good practice to keep the `neighborhood` variable for the random forest modeling. If there was an uneven distribution of neighborhoods in both sets, then we would need to consider a different resampling method or removing the variable.

```
# Refreshing our training and testing sets
set.seed(123)
train = clean_house %>% mutate(undervalued = as.factor(undervalued)) %>%
   sample_frac(0.8)
test  = anti_join(clean_house, train, by = 'id')
train %<>% select(-c("id"))
test %<>% select(-c("id"))

library(randomForest)
mod_rf = randomForest(formula = undervalued ~ .,
                      data = train,
                      importance = T,
                      ntree = 100)
importance(mod_rf)
```

```
##                     FALSE        TRUE MeanDecreaseAccuracy MeanDecreaseGini
## ms_sub_class     2.37110510 -0.59289756          2.230481757       6.25671639
## ms_zoning        2.96050446 -0.39285711          2.660849202       2.98502094
## lot_frontage     1.31352137 -0.75593306          0.525960886      14.47293022
## lot_area         0.14130204  2.97691136          2.481027539      21.74603953
## street           0.00000000  1.00503782          1.005037815       0.14684310
## alley            2.08679934 -0.93761809          1.052651776       1.55984791
## lot_shape        0.53106960 -0.86906473         -0.280213261       4.27020668
## land_contour     1.39066020  0.96429231          1.761896764       3.48457769
## utilities        0.00000000  0.00000000          0.000000000       0.01714286
## lot_config       0.84471029  1.31603075          1.538559412       5.68282684
## land_slope       0.47421537  1.33882923          1.771745823       1.46127417
## neighborhood     9.15263439 -1.41384978          7.012149979      49.00285106
## condition1       1.52000381  0.91315354          1.974712262       6.14739613
## condition2       0.00000000 -1.00503782         -1.005037815       0.19665702
## bldg_type        1.12996782  0.12753681          0.942682077       3.37526159
## house_style      1.38336800  0.66013400          1.800959183       5.70180504
## overall_qual     1.68452586  6.57037197          6.717004190      16.11463389
## overall_cond     2.11388375 -0.11103405          1.758216763       6.25304209
## year_built       3.16800540  0.88441877          3.404948037      14.99962580
## year_remod_add   2.88187116  1.79992658          4.174840176      14.38416237
## roof_style       1.49534911 -1.44243954         -0.011418427       3.19500972
## roof_matl       -0.01075588  0.33854578          0.254120308       0.59345575
## exterior1st      4.30936617 -1.75965058          2.426072343      17.01934930
## exterior2nd      4.91475920 -1.17302712          2.957139085      18.57794048
## mas_vnr_type     1.29290657 -0.07143388          0.953393308       3.65983121
## mas_vnr_area     2.40413643 -1.79750713          0.688785035      11.17923784
## exter_qual       3.28861788  2.58876793          4.100838893       4.85561014
## exter_cond       3.03014026 -0.62215309          2.054099581       2.12103282
## foundation       1.80653485  2.53050396          3.088919431       4.08226693
## bsmt_qual        0.21747787  1.18317604          1.048360612       3.49740634
## bsmt_cond        1.34455606  0.58695672          1.156748477       1.93102645
## bsmt_exposure    0.39992031  3.24006287          2.459550251       8.83336503
## bsmt_fin_type1  -0.75334871  1.62290860          0.644859625      11.48029368
## bsmt_fin_sf1     0.25873156  3.55411626          2.947305142      16.96627929
## bsmt_fin_type2   1.96117137 -0.71964105          1.023802693       5.58847525
## bsmt_fin_sf2     2.45084614 -0.68022909          1.576533782       3.83102914
## bsmt_unf_sf      2.45604685  1.09652020          3.087944915      18.86575270
## total_bsmt_sf    2.91840785  1.87437567          3.844247593      19.86606691
## heating         -0.85114868 -0.46607137         -1.289622553       0.41671989
## heating_qc      -0.89839832 -0.04319601         -0.783790997       6.03820039
## central_air      0.31879951 -0.37323215         -0.001255129       0.76931049
## electrical      -0.95169657  0.68866317         -0.255548189       1.40753503
## x1st_flr_sf      3.07262823  2.52068801          4.954511150      21.21923469
## x2nd_flr_sf      2.00559147  1.62843691          3.094340721      10.54007485
## low_qual_fin_sf -0.73691411 -0.47976615         -0.672646514       0.60291609
## gr_liv_area      1.05040323  5.21595445          5.685237227      26.44762641
## bsmt_full_bath   0.27016929  0.42142287          0.459635305       3.18937797
## bsmt_half_bath  -1.18006922  1.85579676          0.673713255       1.30181844
## full_bath        3.19254213  0.47346541          2.732841615       3.06827333
## half_bath       -0.21995308  1.99493077          1.456179460       3.04303319
## bedroom_abv_gr   1.55899450 -0.23970178          0.970044209       5.69155134
## kitchen_abv_gr   1.16430043  1.15335008          1.754335069       0.95336992
## kitchen_qual     0.69174921 -0.16207846          0.567714859       4.03953453
## tot_rms_abv_grd  1.41720781  0.97973296          1.787462605      11.03212578
## functional       0.52206798  0.22259156          0.671310737       2.68818438
```

```
## fireplaces       2.51830238 -1.52092312         0.985649842       3.91226395
## fireplace_qu     0.69102897 -0.91224663        -0.070888822       7.15109410
## garage_type     -1.57165483  0.25323769        -1.101717772       4.37105256
## garage_yr_blt    1.69176319 -0.03213763         1.368718929      14.56753377
## garage_finish    0.64110738 -1.20190809        -0.512738270       5.79121789
## garage_cars      1.43213918  3.53908494         4.114635413       5.03218419
## garage_area      1.68224003  3.72926157         4.470294774      22.20102103
## garage_qual      1.40040682  1.13201326         1.843622225       2.69165645
## garage_cond      1.27111869  1.59375679         1.956286436       1.68613947
## paved_drive      1.01075020 -2.26737841        -1.005071285       1.35452269
## wood_deck_sf     1.57367433  0.33060731         1.733572642      11.20831386
## open_porch_sf    0.70978466  1.53498262         1.850285953      14.98678612
## enclosed_porch  -2.33560242  0.35132891        -1.775893268       3.88019036
## x3ssn_porch      0.05564202 -1.01153916        -0.599054567       1.26608555
## screen_porch     0.55810559  0.05994389         0.480398720       4.11634414
## pool_area        0.00000000 -1.00503782        -1.005037815       0.09971477
## pool_qc          0.00000000  0.03414999         0.031273130       0.10317498
## fence            2.17475352  0.02116753         1.877720232       4.87639051
## misc_feature     1.36030512 -1.48585604        -0.136666022       1.17537986
## misc_val         1.90386644 -0.17248438         1.208188046       1.93359543
## mo_sold          1.49376084  0.49456602         1.248362677      13.38199354
## yr_sold         -2.26464306 -0.93846176        -2.191942636       8.68738217
## sale_type       -0.91609319 -0.18198627        -0.890808478       2.18995595
## sale_condition   0.89141467  0.57870719         1.058412573       5.50860524
```

```
train$pred_rf = predict(mod_rf, type="response", newdata = train)
# Creating confusion matrix
table(Actual = train$undervalued,
      Predicted = train$pred_rf)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE   619    0
##   TRUE      0  549
```

Looks like the neighborhood ended up being a significant variable according to the random forest model. Good thing we kept it!

Testing

First with the most-recently constructed random forest model

```
test$pred_rf = predict(mod_rf, type="response", newdata = test)
table(Actual = test$undervalued,
      Predicted = test$pred_rf)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE    93   42
##   TRUE     68   89
```

```
TN = 93
TP = 89
FP = 42
FN = 68
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_rf = rbind(Sensitivity,Specificity,Accuracy)
metrics_rf
```

```
##                  [,1]
## Sensitivity 0.5668790
## Specificity 0.6888889
## Accuracy    0.6232877
```

Not quite the same performance as with training, but we can compare these results to the test predictions of the other models as well.

GLM model with cutoff value of 0.5

```
# refreshing sets
set.seed(123)
train = clean_house %>% mutate(undervalued = as.factor(undervalued)) %>%
   sample_frac(0.8)
test  = anti_join(clean_house, train, by = 'id')
train %<>% select(-c("id"))
test %<>% select(-c("id"))

glm_mod_fin = glm(
   undervalued ~
     overall_qual + bsmt_fin_sf1 + kitchen_qual + bldg_type,
   family = "binomial",
   data = test)

# Cutoff Value set to 0.5
test$predprob <- round(fitted(glm_mod_fin),2)
table(Actual = test$undervalued,
    Predicted = test$predprob>0.5)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE    60   75
##   TRUE     35  122
```

```r
TN = 60
TP = 122
FP = 75
FN = 35
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_glm = rbind(Sensitivity,Specificity,Accuracy)
metrics_glm
```

```
##                  [,1]
## Sensitivity 0.7770701
## Specificity 0.4444444
## Accuracy    0.6232877
```

Single tree (with Neighborhood)

```r
# Predicting values
as.data.frame(predict(best_tree, new_data=test)) -> df1

df1 %<>% rename(pred_tree = names(.)[1])
# Creating confusion matrix
table(Actual = test$undervalued,
      Predicted = df1$pred_tree)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE    84   51
##   TRUE     63   94
```

```r
TN = 84
TP = 94
FP = 51
FN = 63
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_tree1 = rbind(Sensitivity,Specificity,Accuracy)
metrics_tree1
```

```
##                  [,1]
## Sensitivity 0.5987261
## Specificity 0.6222222
## Accuracy    0.6095890
```

Single tree (without Neighborhood)

```
# Predicting values
as.data.frame(predict(best_tree2, new_data=test_hold)) -> df2
df2 %<>% rename(pred_tree = names(.)[1])
# Creating confusion matrix
table(Actual = test_hold$undervalued,
      Predicted = df2$pred_tree)
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE    97   38
##   TRUE     75   82
```

```
TN = 97
TP = 82
FP = 38
FN = 75
Sensitivity = TP/(TP + FN)
# (Number of true positive assessment)/(Number of
#  all positive assessment)
Specificity = TN/(TN + FP)
# (Number of true negative assessment)/(Number of
#  all negative assessment)
Accuracy   = (TN + TP)/(TN+TP+FN+FP)
# (Number of correct assessments)/Number of
#  all assessments)
metrics_tree2 = rbind(Sensitivity,Specificity,Accuracy)
metrics_tree2
```

```
##                  [,1]
## Sensitivity 0.5222930
## Specificity 0.7185185
## Accuracy    0.6130137
```

03 - How did you do? Compare your models’ levels of accuracy to the null classifier?

```
as.data.frame(cbind(metrics_glm,
   metrics_tree1,metrics_tree2,metrics_rf)) -> comp_df
comp_df %>% rename("GLM" = 1,
                   "Tree #1" = 2,
                   "Tree #3" = 3,
                   "Random Forest"=4)
```

```
##                   GLM   Tree #1   Tree #3 Random Forest
## Sensitivity 0.7770701 0.5987261 0.5222930     0.5668790
## Specificity 0.4444444 0.6222222 0.7185185     0.6888889
## Accuracy    0.6232877 0.6095890 0.6130137     0.6232877
```

As mentioned above, the random forest model performed the best on the training data, but not as well as we would have hoped for on the testing data. Something we could consider doing would be to create bins when attempting binary logistic regression. Doing so might reveal a linear trend that helps reduce the outlier effect. It would also be interesting to see the results of some different models that automatically choose the variables to be included. We can already see the variation that arises with slight changes to the model or data, so perhaps there is a model we did not try is the one used to construct the `undervaluded` variable

04 - Are all errors equal in this setting? Briefly explain your answer.

All errors are NOT equal. For example, consider two different observation that have a `TRUE` value of `undervalued` variable. Even though both of these observations are considered undervalued, suppose that observation 1 is undervalued by 500% whereas observation 2 is only undervalued by 1%. If our model was to predict a `TRUE` value for the `undervalued` variable for observation 1 and 2, even thought the marginal cost (in terms of additional model uncertainty from errors) would be relatively similar (or the exact same), an incorrect classification of observation 1 would have a greater impact of the *true* performance strength of out model.

05 - Why would it be a bad idea to use a linear model here (for example plain OLS or lasso)?

Since the outcome we are predicting is a binary categorical variable rather than a variable with a continuous value. When there are potential outliers, regression models like these can also be more susceptible to the influence of outliers. Even if our data had optimal characteristics for using a regression model, there still might not be a linear trend that the model is able to properly fit. Decision trees, on the other hand, are much better at fitting non-linear trends that exist in the model.

In addition, another assumption in linear models is that the variables included in the model are independent. While we could attempt to navigate around this by using interaction variables in the model, but it would be more worth our while to utilize a different method better suited for classification problems.

```
test -> testingtest
as.data.frame(cbind(metrics_glm,
   metrics_tree1,metrics_tree2,metrics_rf)) -> comp_df
comp_df %>% rename("GLM" = 1,
                   "Tree #1" = 2,
                   "Tree #3" = 3,
                   "Random Forest"=4)
```


```
##                   GLM   Tree #1   Tree #3 Random Forest
## Sensitivity 0.7770701 0.5987261 0.5222930     0.5668790
## Specificity 0.4444444 0.6222222 0.7185185     0.6888889
## Accuracy    0.6232877 0.6095890 0.6130137     0.6232877
```