Above\_Median Project

Tabitha Hagen

2022-10-15

# Project for BAN 502 Predictive Analytics

# Read in the clean\_ames\_table which was created in “THagenBAN502Project\_Phase1.rmd”

ames\_df <- readRDS("new\_ames\_table.rds")

str (ames\_df)

## tibble [2,053 × 22] (S3: tbl\_df/tbl/data.frame)  
## $ Above\_Median : Factor w/ 2 levels "Yes","No": 1 2 1 1 1 1 1 1 1 1 ...  
## $ Lot\_Shape : Factor w/ 4 levels "Slightly\_Irregular",..: 1 2 1 2 1 1 1 1 1 2 ...  
## $ Overall\_Qual : Factor w/ 10 levels "Above\_Average",..: 1 2 1 3 2 1 4 4 1 3 ...  
## $ Mas\_Vnr\_Type : Factor w/ 5 levels "Stone","None",..: 1 2 3 2 2 3 2 2 2 2 ...  
## $ Exter\_Qual : Factor w/ 4 levels "Typical","Good",..: 1 1 1 2 1 1 2 2 1 1 ...  
## $ Foundation : Factor w/ 6 levels "CBlock","PConc",..: 1 1 1 1 2 2 2 2 2 2 ...  
## $ Bsmt\_Qual : Factor w/ 6 levels "Typical","Good",..: 1 1 1 1 2 1 2 2 2 2 ...  
## $ Heating\_QC : Factor w/ 5 levels "Fair","Typical",..: 1 2 2 3 4 3 3 3 3 4 ...  
## $ Kitchen\_Qual : Factor w/ 5 levels "Typical","Good",..: 1 1 2 3 1 2 2 2 1 2 ...  
## $ Fireplace\_Qu : Factor w/ 6 levels "Good","No\_Fireplace",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ Garage\_Type : Factor w/ 7 levels "Attchd","BuiltIn",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ Garage\_Finish : Factor w/ 4 levels "Fin","Unf","RFn",..: 1 2 2 1 1 1 3 3 1 2 ...  
## $ Mas\_Vnr\_Area : num [1:2053] 112 0 108 0 0 20 0 0 0 0 ...  
## $ Second\_Flr\_SF : num [1:2053] 0 0 0 0 701 678 0 0 0 0 ...  
## $ Low\_Qual\_Fin\_SF: num [1:2053] 0 0 0 0 0 0 0 0 0 0 ...  
## $ Half\_Bath : num [1:2053] 0 0 1 1 1 1 0 0 0 1 ...  
## $ Fireplaces : num [1:2053] 2 0 0 2 1 1 0 1 0 1 ...  
## $ Wood\_Deck\_SF : num [1:2053] 210 140 393 0 212 360 0 237 483 192 ...  
## $ Open\_Porch\_SF : num [1:2053] 62 0 36 0 34 36 82 152 21 0 ...  
## $ Enclosed\_Porch : num [1:2053] 0 0 0 0 0 0 0 0 0 0 ...  
## $ Screen\_Porch : num [1:2053] 0 120 0 0 0 0 144 0 0 0 ...  
## $ Neighborhood : Factor w/ 28 levels "North\_Ames","Gilbert",..: 1 1 1 1 2 2 3 3 2 2 ...

summary(ames\_df)

## Above\_Median Lot\_Shape Overall\_Qual Mas\_Vnr\_Type   
## Yes:1043 Slightly\_Irregular : 714 Average :587 Stone : 166   
## No :1010 Regular :1275 Above\_Average:518 None :1231   
## Moderately\_Irregular: 53 Good :411 BrkFace: 638   
## Irregular : 11 Very\_Good :237 BrkCmn : 17   
## Below\_Average:169 CBlock : 1   
## Excellent : 70   
## (Other) : 61   
## Exter\_Qual Foundation Bsmt\_Qual Heating\_QC   
## Typical :1272 CBlock:880 Typical :911 Fair : 61   
## Good : 682 PConc :911 Good :849 Typical : 618   
## Excellent: 78 Wood : 4 Excellent :178 Excellent:1040   
## Fair : 21 BrkTil:216 No\_Basement: 57 Good : 333   
## Slab : 36 Fair : 57 Poor : 1   
## Stone : 6 Poor : 1   
##   
## Kitchen\_Qual Fireplace\_Qu Garage\_Type Garage\_Finish  
## Typical :1070 Good :538 Attchd :1204 Fin :509   
## Good : 790 No\_Fireplace:993 BuiltIn : 127 Unf :872   
## Excellent: 142 Typical :409 Basment : 29 RFn :563   
## Fair : 50 Poor : 36 Detchd : 549 No\_Garage:109   
## Poor : 1 Excellent : 21 No\_Garage : 108   
## Fair : 56 CarPort : 15   
## More\_Than\_Two\_Types: 21   
## Mas\_Vnr\_Area Second\_Flr\_SF Low\_Qual\_Fin\_SF Half\_Bath   
## Min. : 0.0 Min. : 0.0 Min. : 0.000 Min. :0.0000   
## 1st Qu.: 0.0 1st Qu.: 0.0 1st Qu.: 0.000 1st Qu.:0.0000   
## Median : 0.0 Median : 0.0 Median : 0.000 Median :0.0000   
## Mean : 103.8 Mean : 326.1 Mean : 4.973 Mean :0.3751   
## 3rd Qu.: 164.0 3rd Qu.: 701.0 3rd Qu.: 0.000 3rd Qu.:1.0000   
## Max. :1600.0 Max. :1862.0 Max. :1064.000 Max. :2.0000   
##   
## Fireplaces Wood\_Deck\_SF Open\_Porch\_SF Enclosed\_Porch   
## Min. :0.000 Min. : 0.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.:0.000 1st Qu.: 0.00 1st Qu.: 0.00 1st Qu.: 0.00   
## Median :1.000 Median : 0.00 Median : 27.00 Median : 0.00   
## Mean :0.603 Mean : 93.52 Mean : 48.17 Mean : 23.02   
## 3rd Qu.:1.000 3rd Qu.: 168.00 3rd Qu.: 72.00 3rd Qu.: 0.00   
## Max. :4.000 Max. :1424.00 Max. :742.00 Max. :584.00   
##   
## Screen\_Porch Neighborhood   
## Min. : 0.00 North\_Ames : 327   
## 1st Qu.: 0.00 College\_Creek: 183   
## Median : 0.00 Old\_Town : 181   
## Mean : 16.68 Edwards : 129   
## 3rd Qu.: 0.00 Somerset : 119   
## Max. :576.00 Gilbert : 109   
## (Other) :1005

# Creating a Classification Tree

Splitting the data into a Training subset and a Testing subset.

set.seed(123)   
ames\_df\_split = initial\_split(ames\_df, prop = 0.7, strata = Above\_Median) #70% in training  
train = training(ames\_df\_split)   
test = testing(ames\_df\_split)

Now that we have the split data, let’s build a classification tree.

ames\_df\_recipe = recipe(Above\_Median ~ Overall\_Qual+Mas\_Vnr\_Type+Exter\_Qual+Foundation+Bsmt\_Qual+Heating\_QC+Kitchen\_Qual+Fireplace\_Qu+Garage\_Type+Garage\_Finish+Mas\_Vnr\_Area+Second\_Flr\_SF+Low\_Qual\_Fin\_SF+Half\_Bath+Fireplaces+Wood\_Deck\_SF+Open\_Porch\_SF+Enclosed\_Porch+Screen\_Porch+Neighborhood, train)  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% # don't forget the model = TRUE flag  
 set\_mode("classification")  
  
ames\_df\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames\_df\_recipe)  
  
ames\_df\_fit = fit(ames\_df\_wflow, train)

Let’s take a look at our classification tree (a couple of ways)

#look at the tree's fit  
ames\_df\_fit %>%  
 extract\_fit\_parsnip() %>%  
 #pull\_workflow\_fit() %>%  
 pluck("fit") #Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.

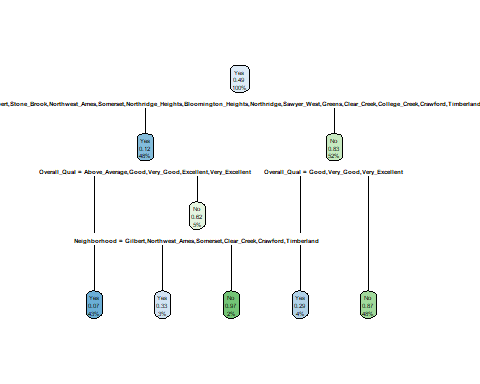
## n= 1437   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1437 707 Yes (0.50800278 0.49199722)   
## 2) Neighborhood=Gilbert,Stone\_Brook,Northwest\_Ames,Somerset,Northridge\_Heights,Bloomington\_Heights,Northridge,Sawyer\_West,Greens,Clear\_Creek,College\_Creek,Crawford,Timberland,Veenker,Green\_Hills 687 85 Yes (0.87627365 0.12372635)   
## 4) Overall\_Qual=Above\_Average,Good,Very\_Good,Excellent,Very\_Excellent 621 44 Yes (0.92914654 0.07085346) \*  
## 5) Overall\_Qual=Average,Below\_Average 66 25 No (0.37878788 0.62121212)   
## 10) Neighborhood=Gilbert,Northwest\_Ames,Somerset,Clear\_Creek,Crawford,Timberland 36 12 Yes (0.66666667 0.33333333) \*  
## 11) Neighborhood=Sawyer\_West,College\_Creek 30 1 No (0.03333333 0.96666667) \*  
## 3) Neighborhood=North\_Ames,Briardale,Northpark\_Villa,Sawyer,Old\_Town,Brookside,Iowa\_DOT\_and\_Rail\_Road,South\_and\_West\_of\_Iowa\_State\_University,Edwards,Mitchell,Meadow\_Village,Blueste,Landmark 750 128 No (0.17066667 0.82933333)   
## 6) Overall\_Qual=Good,Very\_Good,Very\_Excellent 55 16 Yes (0.70909091 0.29090909) \*  
## 7) Overall\_Qual=Above\_Average,Average,Below\_Average,Fair,Poor,Very\_Poor 695 89 No (0.12805755 0.87194245) \*

#Please use `extract\_fit\_parsnip()` instead

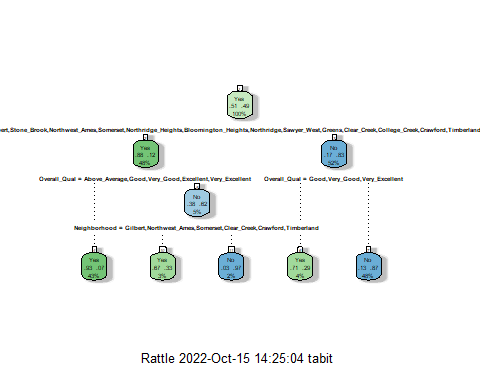
#extract the tree's fit from the fit object  
tree = ames\_df\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

#plot the tree  
rpart.plot(tree)



#alternative  
fancyRpartPlot(tree)



Looking at the “rpart” complexity parameter “cp”.  
A lower Cp value will allow more splits to occur, but at a greater risk of overfitting. A higher Cp value may not provide enough splits. It won’t split if the improvement is not good enough to wort the split.

ames\_df\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.69872702 0 1.0000000 1.0381895 0.02680266  
## 2 0.03253182 1 0.3012730 0.3111740 0.01930673  
## 3 0.02263083 2 0.2687412 0.2857143 0.01863637  
## 4 0.01697313 3 0.2461103 0.2659123 0.01808058  
## 5 0.01000000 4 0.2291372 0.2432815 0.01740451

I started with only the “Overall\_Qual” variable to start my tree which had 1 level. From my prior tesing using a glm model, this was the strongest predictor.

Then based on my previous models, I added “Lot\_Shape” and I gained a 2nd and 3rd tree level. I kept those and added “Kitchen\_Qual” and gained a 4th tree level.

Then I started added in all the other variables to see how this model could perform. The Classification Tree model decided that ““Overall\_Qual” was still the best predictor of Above\_Median.”Neighborhood” was the strongest predictor variable and that “Overall\_Qual” Interestingly, it placed some of the neighborhoods above the “Overall\_Qual” level and others below it. It decided that there should be an optimal complexity parameter value of .01000000 with a value xerror of 0.2659123.

# Now I will look at XGBoost Models to further develop which predictor variables would be best:

#use\_xgboost(Survived ~., train) #comment me out before knitting

We pick seeds to make sure that all sets of data get a fair amount of all the data.

set.seed(123)  
folds = vfold\_cv(train, v = 5)

Copy and paste the model from the use\_xgboost function. Modify a few elements. We’ll let R tune the parameters by looking at 25 plausible combinations of parameters.

# Commented out because there s a saved model below named : "final\_xgb\_fit"  
#start\_time = Sys.time() #for timing  
  
#xgboost\_recipe <-   
# recipe(formula = Above\_Median ~ ., data = train) %>%   
# #step\_novel(all\_nominal(), -all\_outcomes()) %>%   
# step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%   
# step\_zv(all\_predictors())   
  
#xgboost\_spec <-   
# boost\_tree(trees = tune(), min\_n = tune(), tree\_depth = tune(), learn\_rate = tune(),   
# loss\_reduction = tune(), sample\_size = tune()) %>%   
# set\_mode("classification") %>%   
# set\_engine("xgboost")   
  
#xgboost\_workflow <-   
# workflow() %>%   
# add\_recipe(xgboost\_recipe) %>%   
# add\_model(xgboost\_spec)   
  
#set.seed(77680)  
#xgboost\_tune <-  
# tune\_grid(xgboost\_workflow, resamples = folds, grid = 25)  
  
#end\_time = Sys.time()  
#end\_time - start\_time

# Commented out because there s a saved model below named : "final\_xgb\_fit"  
#best\_xgb = select\_best(xgboost\_tune, "accuracy")  
  
#final\_xgb = finalize\_workflow(  
# xgboost\_workflow,  
# best\_xgb  
#)  
  
#final\_xgb

# Commented out because there s a saved model below named : "final\_xgb\_fit"  
#fit the finalized workflow to our training data  
#final\_xgb\_fit = fit(final\_xgb, train)

Saving

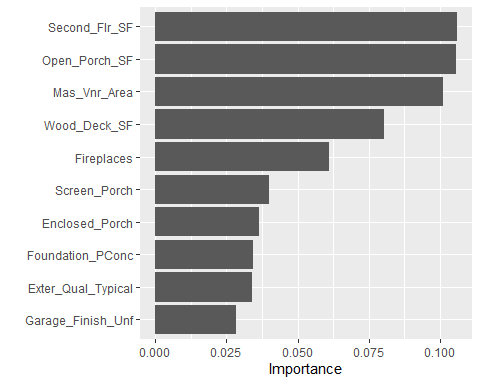
#saveRDS(final\_xgb\_fit,"AMES\_xgb\_fit.rds")

Opening saved XGB Model

final\_xgb\_fit = readRDS("AMES\_xgb\_fit.rds")

Let’s take a look at variable importance before proceeding to SHAP values. We first extract the fit and then feed it to the “vip” function.

xg\_mod = extract\_fit\_parsnip(final\_xgb\_fit)  
vip(xg\_mod$fit)



# Random Forest Model

This model takes awhile, so I’ve commented it out and saved the model “rf\_res-1.rds” to an RDS.

Set-up our folds

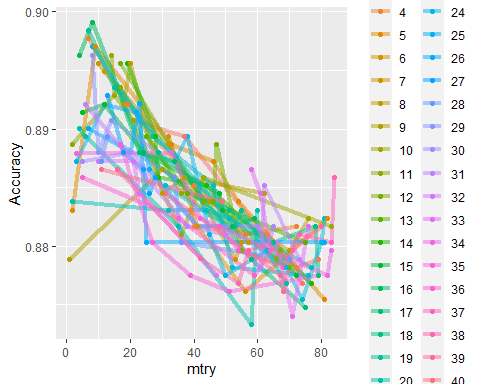
set.seed(123)  
folds = vfold\_cv(train, v = 5)

# rf\_recipe = ames\_df\_recipe %>% #Create basic recipe  
# step\_dummy(all\_nominal(), -all\_outcomes())  
   
#ctrl\_grid = control\_stack\_grid() #necessary for working with tuning grids in the stacks package  
#ctrl\_res = control\_stack\_resamples() #necessary for working with the stacks package  
  
# rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
# set\_engine("ranger", importance = "permutation") %>% #added importance metric  
# set\_mode("classification")  
   
# rf\_wflow =   
# workflow() %>%   
# add\_model(rf\_model) %>%   
# add\_recipe(rf\_recipe)  
   
# set.seed(123)  
# rf\_res = tune\_grid(  
# rf\_wflow,  
# resamples = folds,  
# grid = 200,   
# control = ctrl\_grid  
#)

#saveRDS(rf\_res,"rf\_res.rds")

rf\_res = readRDS("rf\_res.rds")

rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



As min (the colors) gets bigger, the accuracy gets worse. mtry gets to maximize around 8. Towards the right gets a rapidly decline in accuracy.

# CONCLUSION

The four models did not have agreement which variables were strong predictors of a house being “Above\_Median” price. We saw “Overall\_Qual” start strong only to be replaced by “Neighborhood” in our second model. The third model tossed both of these aside and said that “Second\_Flr\_SF” and “Open\_Porch\_SF” mattered more. Overall, I have to say that all of these together would probably make a pretty good model.