# Final Project - Part 2

## Underwood, Katie

library(tidymodels)

## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v broom 0.7.2 v recipes 0.1.15  
## v dials 0.0.9 v rsample 0.0.8   
## v dplyr 1.0.2 v tibble 3.0.4   
## v ggplot2 3.3.2 v tidyr 1.1.2   
## v infer 0.5.4 v tune 0.1.2   
## v modeldata 0.1.0 v workflows 0.2.1   
## v parsnip 0.1.5 v yardstick 0.0.7   
## v purrr 0.3.4

## -- Conflicts ----------------------------------------- tidymodels\_conflicts() --  
## x purrr::discard() masks scales::discard()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x recipes::step() masks stats::step()

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v readr 1.4.0 v forcats 0.5.0  
## v stringr 1.4.0

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x readr::col\_factor() masks scales::col\_factor()  
## x purrr::discard() masks scales::discard()  
## x dplyr::filter() masks stats::filter()  
## x stringr::fixed() masks recipes::fixed()  
## x dplyr::lag() masks stats::lag()  
## x readr::spec() masks yardstick::spec()

library(ROCR)  
library(rpart)

##   
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':  
##   
## prune

library(rpart.plot)  
library(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':  
##   
## precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':  
##   
## lift

library(RColorBrewer)  
library(ranger)

##   
## Attaching package: 'ranger'

## The following object is masked from 'package:rattle':  
##   
## importance

library(vip)

##   
## Attaching package: 'vip'

## The following object is masked from 'package:utils':  
##   
## vi

ames\_student <- read\_csv("ames\_student.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_character(),  
## X1 = col\_double(),  
## Lot\_Frontage = col\_double(),  
## Lot\_Area = col\_double(),  
## Year\_Built = col\_double(),  
## Year\_Remod\_Add = col\_double(),  
## Mas\_Vnr\_Area = col\_double(),  
## BsmtFin\_SF\_1 = col\_double(),  
## BsmtFin\_SF\_2 = col\_double(),  
## Bsmt\_Unf\_SF = col\_double(),  
## Total\_Bsmt\_SF = col\_double(),  
## First\_Flr\_SF = col\_double(),  
## Second\_Flr\_SF = col\_double(),  
## Low\_Qual\_Fin\_SF = col\_double(),  
## Gr\_Liv\_Area = col\_double(),  
## Bsmt\_Full\_Bath = col\_double(),  
## Bsmt\_Half\_Bath = col\_double(),  
## Full\_Bath = col\_double(),  
## Half\_Bath = col\_double(),  
## Bedroom\_AbvGr = col\_double(),  
## Kitchen\_AbvGr = col\_double()  
## # ... with 15 more columns  
## )  
## i Use `spec()` for the full column specifications.

ames\_student\_clean = ames\_student %>%   
 mutate(Street = factor(Street)) %>%  
 mutate(Alley = factor(Alley)) %>%  
 mutate(Lot\_Shape = factor(Lot\_Shape)) %>%  
 mutate(Land\_Contour = factor(Land\_Contour)) %>%  
 mutate(Utilities = factor(Utilities)) %>%  
 mutate(Lot\_Config = factor(Lot\_Config)) %>%  
 mutate(Land\_Slope = factor(Land\_Slope)) %>%  
 mutate(Neighborhood = factor(Neighborhood)) %>%  
 mutate(MS\_Zoning = factor(MS\_Zoning)) %>%  
 mutate(MS\_SubClass = factor(MS\_SubClass)) %>%  
 mutate(Condition\_1 = factor(Condition\_1)) %>%  
 mutate(Condition\_2 = factor(Condition\_2)) %>%  
 mutate(Bldg\_Type = factor(Bldg\_Type)) %>%  
 mutate(House\_Style = factor(House\_Style)) %>%  
 mutate(Overall\_Qual = factor(Overall\_Qual)) %>%  
 mutate(Overall\_Cond = factor(Overall\_Cond)) %>%  
 mutate(Roof\_Style = factor(Roof\_Style)) %>%  
 mutate(Roof\_Matl = factor(Roof\_Matl)) %>%  
 mutate(Exterior\_1st = factor(Exterior\_1st)) %>%  
 mutate(Exterior\_2nd = factor(Exterior\_2nd)) %>%  
 mutate(Mas\_Vnr\_Type = factor(Mas\_Vnr\_Type)) %>%  
 mutate(Exter\_Qual = factor(Exter\_Qual)) %>%  
 mutate(Exter\_Cond = factor(Exter\_Cond)) %>%  
 mutate(Foundation = factor(Foundation)) %>%  
 mutate(Bsmt\_Qual = factor(Bsmt\_Qual)) %>%  
 mutate(Bsmt\_Cond = factor(Bsmt\_Cond)) %>%  
 mutate(Bsmt\_Exposure = factor(Bsmt\_Exposure)) %>%  
 mutate(BsmtFin\_Type\_1 = factor(BsmtFin\_Type\_1)) %>%  
 mutate(BsmtFin\_Type\_2 = factor(BsmtFin\_Type\_2)) %>%  
 mutate(Heating = factor(Heating)) %>%  
 mutate(Heating\_QC = factor(Heating\_QC)) %>%  
 mutate(Central\_Air = factor(Central\_Air)) %>%  
 mutate(Electrical = factor(Electrical)) %>%  
 mutate(Kitchen\_Qual = factor(Kitchen\_Qual)) %>%  
 mutate(Fireplace\_Qu = factor(Fireplace\_Qu)) %>%  
 mutate(Garage\_Type = factor(Garage\_Type)) %>%  
 mutate(Garage\_Finish = factor(Garage\_Finish)) %>%  
 mutate(Garage\_Qual = factor(Garage\_Qual)) %>%  
 mutate(Garage\_Cond = factor(Garage\_Cond)) %>%  
 mutate(Paved\_Drive = factor(Paved\_Drive)) %>%  
 mutate(Pool\_QC = factor(Pool\_QC)) %>%  
 mutate(Fence = factor(Fence)) %>%  
 mutate(Misc\_Feature = factor(Misc\_Feature)) %>%  
 mutate(Sale\_Type = factor(Sale\_Type)) %>%  
 mutate(Sale\_Condition = factor(Sale\_Condition)) %>%  
 mutate(Above\_Median = factor(Above\_Median)) %>%  
 mutate(Functional = factor(Functional)) %>%  
 mutate(Mo\_Sold = factor(Mo\_Sold)) %>%  
 mutate(Year\_Sold = factor(Year\_Sold))

ames\_split = initial\_split(ames\_student\_clean, prop = .7, strata = Above\_Median)  
train = training(ames\_split)  
test = testing(ames\_split)

### Logistic Regression Model

AIC 872

ames\_lr\_model =  
 logistic\_reg() %>%  
 set\_engine("glm")  
  
ames\_recipe = recipe(Above\_Median ~ Year\_Built + Gr\_Liv\_Area + Full\_Bath + Garage\_Cars + First\_Flr\_SF + Total\_Bsmt\_SF + Garage\_Area, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(ames\_recipe) %>%  
 add\_model(ames\_lr\_model)  
  
ames\_fit = fit(logreg\_wf, train)  
  
#summary(ames\_fit$fit$fit$fit)

AIC 592

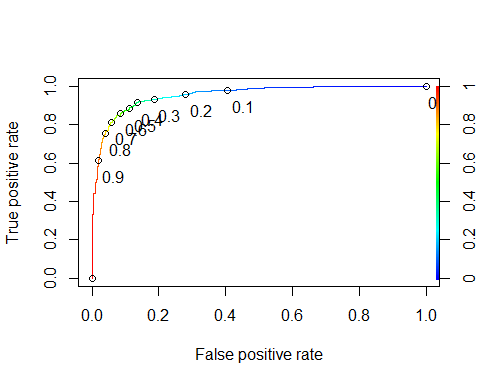
ames\_lr\_model =  
 logistic\_reg() %>%  
 set\_engine("glm")  
  
ames\_recipe2 = recipe(Above\_Median ~ ., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf2 = workflow() %>%  
 add\_recipe(ames\_recipe2) %>%  
 add\_model(ames\_lr\_model)  
  
ames\_fit2 = fit(logreg\_wf2, train)

## Warning: glm.fit: algorithm did not converge

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

#summary(ames\_fit2$fit$fit$fit)

predictions\_lr = predict(ames\_fit, train, type= "prob")  
ROCRpred = prediction(predictions\_lr[,2], train$Above\_Median)  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],  
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.8919289  
## specificity 0.8868458  
## cutoff 0.4735307

Accuracy is 88.1% on training set for Logistic Regression model

t1 = table(train$Above\_Median, predictions\_lr[,2] > 0.5344833)  
t1

##   
## FALSE TRUE  
## No 639 68  
## Yes 95 636

(t1[1,1]+t1[2,2])/nrow(train)

## [1] 0.8866481

Accuracy remains at 88.45% on testing set for Logistic Regression model

predictions\_lr\_test = predict(ames\_fit, test, type= "prob")  
t2 = table(test$Above\_Median, predictions\_lr\_test[,2] > 0.5344833)  
t2

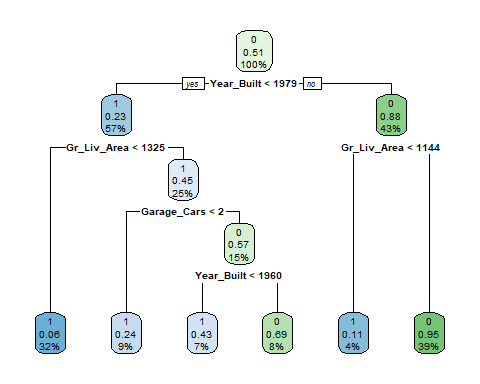
##   
## FALSE TRUE  
## No 265 38  
## Yes 44 268

(t2[1,1]+t2[2,2])/nrow(test)

## [1] 0.8666667

### Classification Trees model

ames\_student\_clean2 = ames\_student\_clean %>% mutate(Above\_Median = as\_factor(Above\_Median)) %>%  
 mutate(Above\_Median = fct\_recode(Above\_Median, "1" = "No", "0" = "Yes"))  
ames\_split2 = initial\_split(ames\_student\_clean2, prop = .7, strata = Above\_Median)  
train2 = training(ames\_split2)  
test2 = testing(ames\_split2)  
  
tree\_model = decision\_tree() %>%  
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
classification\_wflow =   
 workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(ames\_recipe)  
  
ames\_fit\_classification = fit(classification\_wflow, train2)  
  
tree = ames\_fit\_classification %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
  
rpart.plot(tree)



folds = vfold\_cv(train2, v = 5)  
  
ames\_classification\_model2 = decision\_tree(cost\_complexity = tune()) %>%  
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
ames\_grid = grid\_regular(cost\_complexity(), levels = 25)  
  
ames\_classification\_wflow2 =  
 workflow() %>%  
 add\_model(ames\_classification\_model2) %>%  
 add\_recipe(ames\_recipe)  
  
tree\_res =   
 ames\_classification\_wflow2 %>%  
 tune\_grid(  
 resamples = folds,  
 grid = ames\_grid  
 )

##   
## Attaching package: 'rlang'

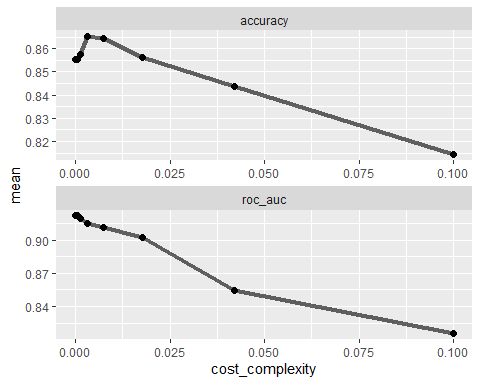
## The following objects are masked from 'package:purrr':  
##   
## %@%, as\_function, flatten, flatten\_chr, flatten\_dbl, flatten\_int,  
## flatten\_lgl, flatten\_raw, invoke, list\_along, modify, prepend,  
## splice

##   
## Attaching package: 'vctrs'

## The following object is masked from 'package:tibble':  
##   
## data\_frame

## The following object is masked from 'package:dplyr':  
##   
## data\_frame

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00316 Preprocessor1\_Model21

final\_wf =   
 ames\_classification\_wflow2 %>%  
 finalize\_workflow(best\_tree)  
  
final\_fit = fit(final\_wf, train2)  
  
tree2 = final\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

Accuracy is 88.0% on the training set for Classification trees

treepred = predict(final\_fit, train2, type = "class")  
head(treepred)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 0   
## 2 1   
## 3 1   
## 4 0   
## 5 0   
## 6 0

confusionMatrix(treepred$.pred\_class,train2$Above\_Median,positive="1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 0  
## 1 664 95  
## 0 43 636  
##   
## Accuracy : 0.904   
## 95% CI : (0.8876, 0.9188)  
## No Information Rate : 0.5083   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8082   
##   
## Mcnemar's Test P-Value : 1.416e-05   
##   
## Sensitivity : 0.9392   
## Specificity : 0.8700   
## Pos Pred Value : 0.8748   
## Neg Pred Value : 0.9367   
## Prevalence : 0.4917   
## Detection Rate : 0.4618   
## Detection Prevalence : 0.5278   
## Balanced Accuracy : 0.9046   
##   
## 'Positive' Class : 1   
##

Accuracy is 87.3% on testing set

treepred2 = predict(final\_fit, test2, type = "class")  
head(treepred2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 0   
## 2 0   
## 3 0   
## 4 0   
## 5 1   
## 6 1

confusionMatrix(treepred2$.pred\_class,test2$Above\_Median,positive="1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 0  
## 1 274 54  
## 0 29 258  
##   
## Accuracy : 0.865   
## 95% CI : (0.8355, 0.8911)  
## No Information Rate : 0.5073   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.7303   
##   
## Mcnemar's Test P-Value : 0.00843   
##   
## Sensitivity : 0.9043   
## Specificity : 0.8269   
## Pos Pred Value : 0.8354   
## Neg Pred Value : 0.8990   
## Prevalence : 0.4927   
## Detection Rate : 0.4455   
## Detection Prevalence : 0.5333   
## Balanced Accuracy : 0.8656   
##   
## 'Positive' Class : 1   
##

### Random Forest

rf\_folds = vfold\_cv(train, v =5)  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>%  
 set\_engine("ranger", importance = "permutation") %>%  
 set\_mode("classification")  
  
ames\_wflow =  
 workflow() %>%  
 add\_model(rf\_model) %>%  
 add\_recipe(ames\_recipe)  
  
rf\_grid = grid\_regular(  
 mtry(range = c(2, 8)),  
 min\_n(range = c(5, 20)), levels = 10  
)  
  
rf\_res\_tuned = tune\_grid(ames\_wflow,   
 resamples = rf\_folds,  
 grid = rf\_grid)

## ! Fold1: preprocessor 1/1, model 7/70: 8 columns were requested but there were 7 ...

## ! Fold1: preprocessor 1/1, model 14/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 21/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 28/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 35/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 42/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 49/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 56/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 63/70: 8 columns were requested but there were 7...

## ! Fold1: preprocessor 1/1, model 70/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 7/70: 8 columns were requested but there were 7 ...

## ! Fold2: preprocessor 1/1, model 14/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 21/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 28/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 35/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 42/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 49/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 56/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 63/70: 8 columns were requested but there were 7...

## ! Fold2: preprocessor 1/1, model 70/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 7/70: 8 columns were requested but there were 7 ...

## ! Fold3: preprocessor 1/1, model 14/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 21/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 28/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 35/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 42/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 49/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 56/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 63/70: 8 columns were requested but there were 7...

## ! Fold3: preprocessor 1/1, model 70/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 7/70: 8 columns were requested but there were 7 ...

## ! Fold4: preprocessor 1/1, model 14/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 21/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 28/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 35/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 42/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 49/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 56/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 63/70: 8 columns were requested but there were 7...

## ! Fold4: preprocessor 1/1, model 70/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 7/70: 8 columns were requested but there were 7 ...

## ! Fold5: preprocessor 1/1, model 14/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 21/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 28/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 35/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 42/70: 8 columns were requested but there were 7...

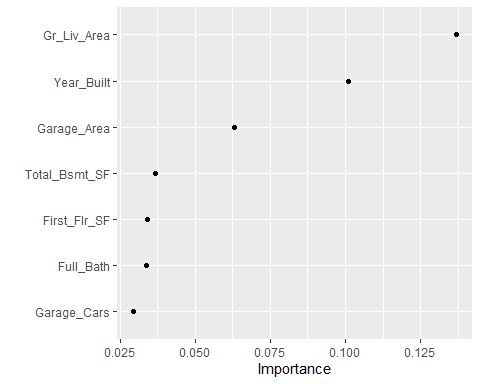
## ! Fold5: preprocessor 1/1, model 49/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 56/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 63/70: 8 columns were requested but there were 7...

## ! Fold5: preprocessor 1/1, model 70/70: 8 columns were requested but there were 7...

best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 ames\_wflow,  
 best\_rf  
)  
  
final\_rf\_fit = fit(final\_rf, train)  
  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



predRF = predict(final\_rf\_fit, train)  
head(predRF)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 No   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

Accuracy is 98.9% on training set for random forest model

confusionMatrix(predRF$.pred\_class, train$Above\_Median, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 702 9  
## Yes 5 722  
##   
## Accuracy : 0.9903   
## 95% CI : (0.9837, 0.9947)  
## No Information Rate : 0.5083   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9805   
##   
## Mcnemar's Test P-Value : 0.4227   
##   
## Sensitivity : 0.9877   
## Specificity : 0.9929   
## Pos Pred Value : 0.9931   
## Neg Pred Value : 0.9873   
## Prevalence : 0.5083   
## Detection Rate : 0.5021   
## Detection Prevalence : 0.5056   
## Balanced Accuracy : 0.9903   
##   
## 'Positive' Class : Yes   
##

Accuracy is 89.7% on testing set for the random forest model

predRF2 = predict(final\_rf\_fit, test)  
head(predRF2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 No   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(predRF2$.pred\_class, test$Above\_Median, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 271 46  
## Yes 32 266  
##   
## Accuracy : 0.8732   
## 95% CI : (0.8443, 0.8984)  
## No Information Rate : 0.5073   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7465   
##   
## Mcnemar's Test P-Value : 0.141   
##   
## Sensitivity : 0.8526   
## Specificity : 0.8944   
## Pos Pred Value : 0.8926   
## Neg Pred Value : 0.8549   
## Prevalence : 0.5073   
## Detection Rate : 0.4325   
## Detection Prevalence : 0.4846   
## Balanced Accuracy : 0.8735   
##   
## 'Positive' Class : Yes   
##

Model I am going to move forward with is the Random Forest model for the competition set.

### Create competition set

competition = read\_csv("ames\_competition.csv")

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_character(),  
## ID = col\_double(),  
## Lot\_Frontage = col\_double(),  
## Lot\_Area = col\_double(),  
## Year\_Built = col\_double(),  
## Year\_Remod\_Add = col\_double(),  
## Mas\_Vnr\_Area = col\_double(),  
## BsmtFin\_SF\_1 = col\_double(),  
## BsmtFin\_SF\_2 = col\_double(),  
## Bsmt\_Unf\_SF = col\_double(),  
## Total\_Bsmt\_SF = col\_double(),  
## First\_Flr\_SF = col\_double(),  
## Second\_Flr\_SF = col\_double(),  
## Low\_Qual\_Fin\_SF = col\_double(),  
## Gr\_Liv\_Area = col\_double(),  
## Bsmt\_Full\_Bath = col\_double(),  
## Bsmt\_Half\_Bath = col\_double(),  
## Full\_Bath = col\_double(),  
## Half\_Bath = col\_double(),  
## Bedroom\_AbvGr = col\_double(),  
## Kitchen\_AbvGr = col\_double()  
## # ... with 15 more columns  
## )  
## i Use `spec()` for the full column specifications.

predRF3 = predict(final\_rf\_fit, competition)  
head(predRF3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 No   
## 6 Yes

kaggle = competition %>% select(ID)  
  
kaggle = bind\_cols(kaggle, predRF3)  
  
kaggle

## # A tibble: 877 x 2  
## ID .pred\_class  
## <dbl> <fct>   
## 1 1 Yes   
## 2 2 Yes   
## 3 3 Yes   
## 4 4 Yes   
## 5 5 No   
## 6 6 Yes   
## 7 7 No   
## 8 8 No   
## 9 9 No   
## 10 10 No   
## # ... with 867 more rows

write.csv(kaggle, "kaggle\_submit.csv", row.names=FALSE)