# Elizabeth Taylor BAN 502 Predictive Analytics

library(tidyverse) # read in the packages I need  
library(tidymodels)  
library(GGally) #ggcorr and ggpairs  
library(ggcorrplot) #correlation plot alternative  
library(gridExtra) #create grids of plots  
library(rpart)  
library(rpart.plot)  
library(RColorBrewer)  
library(rattle)  
library(caret)  
library(esquisse)

library(readr) #read in the data  
ames\_student <- read\_csv("ames\_student.csv")

str(ames\_student) #review the data

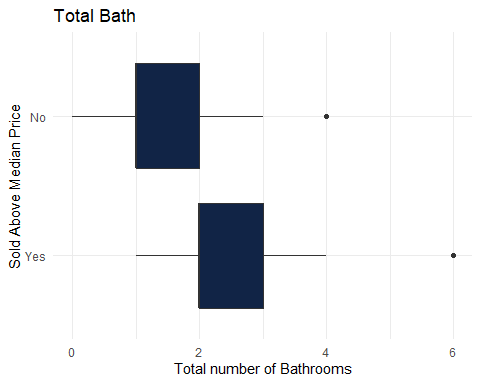
#esquisser(ames\_student) Helped me visualize which variables might be strong predictors of Above Average

ames = ames\_student%>%dplyr::select(Above\_Median,Year\_Built,Gr\_Liv\_Area, Lot\_Area, Full\_Bath,Half\_Bath, TotRms\_AbvGrd, Year\_Built,Year\_Sold, Three\_season\_porch,Screen\_Porch )  
#These were the variables that I found that seemed to be good predictions of the variable Above\_Median

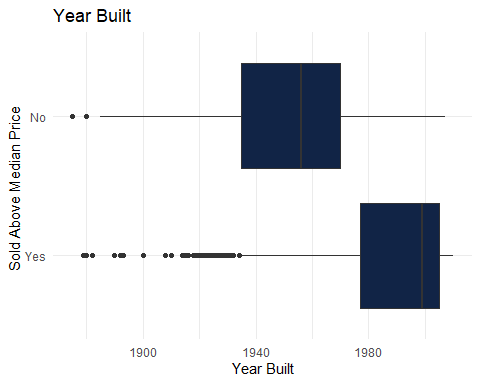
ames = ames %>% mutate(Above\_Median = as\_factor(Above\_Median))   
   
ames$TotBath <- ames$Full\_Bath+ ames$Half\_Bath  
ames$Porch <- ames$Three\_season\_porch+ames$Screen\_Porch  
ames <- ames %>% select(-Full\_Bath,-Half\_Bath,-Three\_season\_porch,-Screen\_Porch)  
#I decide to add both half bath and full bath together to create total bath as one variable.   
#I decided that a three season porch and screen porch were similar enough to add them together as one variable  
#Once I created the new variables I subtracted the old variables out of the data set.

#ggpairs(ames) #So I can visualize my variables on my data set

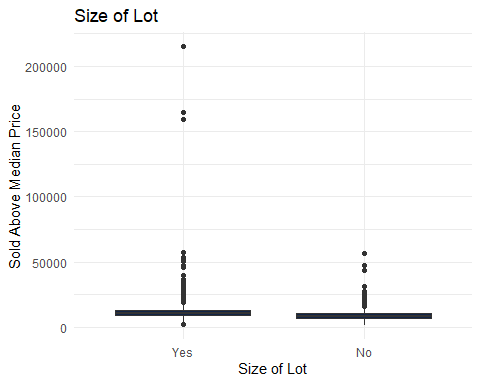
ggplot(ames) +  
 aes(x = TotBath, y = Above\_Median) +  
 geom\_boxplot(fill = "#112446") +  
 labs(  
 x = "Total number of Bathrooms",  
 y = "Sold Above Median Price",  
 title = "Total Bath"  
 ) +  
 theme\_minimal()



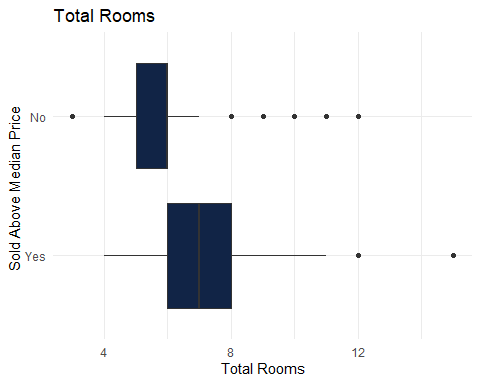
ggplot(ames) +  
 aes(x = Year\_Built, y = Above\_Median) +  
 geom\_boxplot(fill = "#112446") +  
 labs(  
 x = "Year Built",  
 y = "Sold Above Median Price ",  
 title = "Year Built"  
 ) +  
 theme\_minimal()



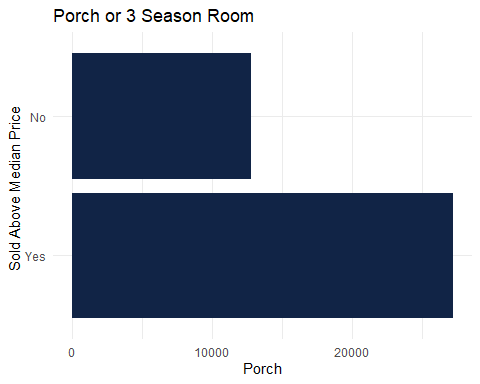
ggplot(ames) +  
 aes(x = Above\_Median, y = Lot\_Area) +  
 geom\_boxplot(fill = "#112446") +  
 labs(  
 x = "Size of Lot",  
 y = "Sold Above Median Price",  
 title = "Size of Lot"  
 ) +  
 theme\_minimal()



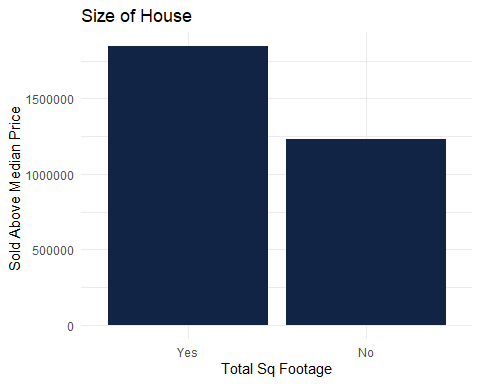
ggplot(ames) +  
 aes(x = TotRms\_AbvGrd, y = Above\_Median) +  
 geom\_boxplot(fill = "#112446") +  
 labs(  
 x = "Total Rooms",  
 y = "Sold Above Median Price ",  
 title = "Total Rooms"  
 ) +  
 theme\_minimal()



ggplot(ames) +aes(x = Porch, y = Above\_Median) +  
 geom\_col(fill = "#112446") +  
 labs(  
 x = "Porch",  
 y = "Sold Above Median Price ",  
 title = "Porch or 3 Season Room"  
 ) +  
 theme\_minimal()



theme\_minimal()  
ggplot(ames) +  
 aes(x = Above\_Median, y = Gr\_Liv\_Area, ) +  
 geom\_col(fill = "#112446") +  
 labs(  
 x = "Total Sq Footage",  
 y = "Sold Above Median Price ",  
 title = "Size of House"  
 ) +  
 theme\_minimal()

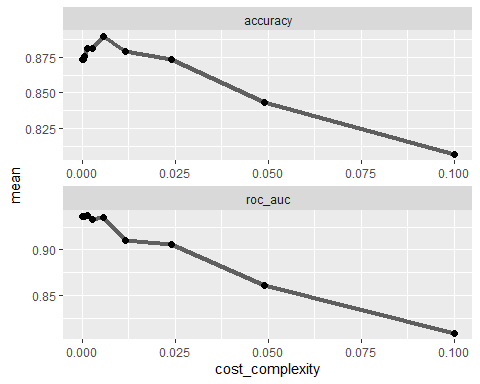


set.seed(123) #spliting in to the testing and training datasets  
#set up the vfolds   
ames\_split = initial\_split(ames, prop = 0.7, strata = Above\_Median)   
train = training(ames\_split)   
test = testing(ames\_split)  
folds = vfold\_cv(train, v = 10)

#creating my classification tree recipe with a complexity model to see the accuracy of the model   
ames\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 30)   
  
ames\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames\_recipe)  
  
tree\_res =   
 ames\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res

## # Tuning results  
## # 10-fold cross-validation   
## # A tibble: 10 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [1293/144]> Fold01 <tibble [60 × 5]> <tibble [0 × 3]>  
## 2 <split [1293/144]> Fold02 <tibble [60 × 5]> <tibble [0 × 3]>  
## 3 <split [1293/144]> Fold03 <tibble [60 × 5]> <tibble [0 × 3]>  
## 4 <split [1293/144]> Fold04 <tibble [60 × 5]> <tibble [0 × 3]>  
## 5 <split [1293/144]> Fold05 <tibble [60 × 5]> <tibble [0 × 3]>  
## 6 <split [1293/144]> Fold06 <tibble [60 × 5]> <tibble [0 × 3]>  
## 7 <split [1293/144]> Fold07 <tibble [60 × 5]> <tibble [0 × 3]>  
## 8 <split [1294/143]> Fold08 <tibble [60 × 5]> <tibble [0 × 3]>  
## 9 <split [1294/143]> Fold09 <tibble [60 × 5]> <tibble [0 × 3]>  
## 10 <split [1294/143]> Fold10 <tibble [60 × 5]> <tibble [0 × 3]>

#the graph that shows the accuracy and ROC\_AUC  
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)

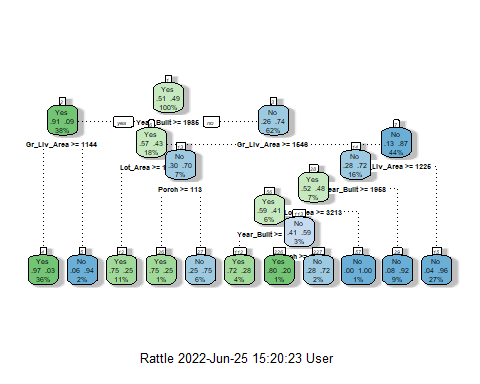


#choosing the tree with the best accuracy   
best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 × 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00574 Preprocessor1\_Model26

#finalizing the workflow  
final\_wf =   
 ames\_wflow %>%   
 finalize\_workflow(best\_tree)

#fitting to the training set  
final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 extract\_fit\_parsnip() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 2.2)



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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 672 75  
## No 58 632  
##   
## Accuracy : 0.9074   
## 95% CI : (0.8913, 0.9219)  
## No Information Rate : 0.508   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8148   
##   
## Mcnemar's Test P-Value : 0.1653   
##   
## Sensitivity : 0.9205   
## Specificity : 0.8939   
## Pos Pred Value : 0.8996   
## Neg Pred Value : 0.9159   
## Prevalence : 0.5080   
## Detection Rate : 0.4676   
## Detection Prevalence : 0.5198   
## Balanced Accuracy : 0.9072   
##   
## 'Positive' Class : Yes   
##

#making predictions on tje testing set   
treepred\_test = predict(final\_fit, test, type = "class")  
head(treepred\_test)

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

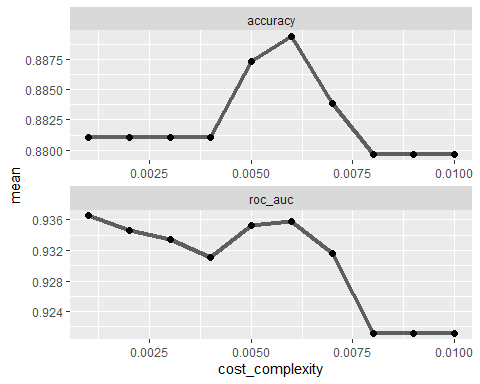
#To see the Confusion Matrix with accuracy sensitivity and Specificity  
confusionMatrix(treepred\_test$.pred\_class,test$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 271 40  
## No 42 263  
##   
## Accuracy : 0.8669   
## 95% CI : (0.8375, 0.8927)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7337   
##   
## Mcnemar's Test P-Value : 0.9121   
##   
## Sensitivity : 0.8658   
## Specificity : 0.8680   
## Pos Pred Value : 0.8714   
## Neg Pred Value : 0.8623   
## Prevalence : 0.5081   
## Detection Rate : 0.4399   
## Detection Prevalence : 0.5049   
## Balanced Accuracy : 0.8669   
##   
## 'Positive' Class : Yes   
##

#tuning  
  
ames\_recipe = recipe(Above\_Median ~., train) %>%   
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = expand.grid(cost\_complexity = seq(0.001,0.01,by=0.001))  
  
ames\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames\_recipe)  
  
tree\_res =   
 ames\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res

## # Tuning results  
## # 10-fold cross-validation   
## # A tibble: 10 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [1293/144]> Fold01 <tibble [20 × 5]> <tibble [0 × 3]>  
## 2 <split [1293/144]> Fold02 <tibble [20 × 5]> <tibble [0 × 3]>  
## 3 <split [1293/144]> Fold03 <tibble [20 × 5]> <tibble [0 × 3]>  
## 4 <split [1293/144]> Fold04 <tibble [20 × 5]> <tibble [0 × 3]>  
## 5 <split [1293/144]> Fold05 <tibble [20 × 5]> <tibble [0 × 3]>  
## 6 <split [1293/144]> Fold06 <tibble [20 × 5]> <tibble [0 × 3]>  
## 7 <split [1293/144]> Fold07 <tibble [20 × 5]> <tibble [0 × 3]>  
## 8 <split [1294/143]> Fold08 <tibble [20 × 5]> <tibble [0 × 3]>  
## 9 <split [1294/143]> Fold09 <tibble [20 × 5]> <tibble [0 × 3]>  
## 10 <split [1294/143]> Fold10 <tibble [20 × 5]> <tibble [0 × 3]>

#after tuning find the tree with the best accuracy   
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)

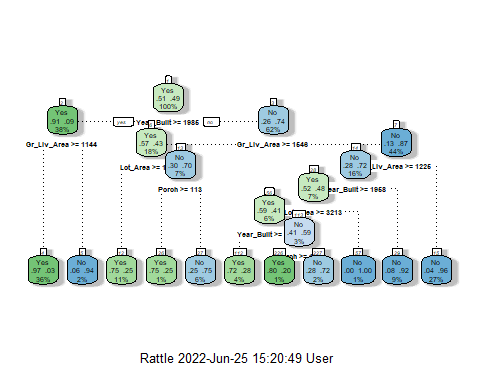


best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 × 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.006 Preprocessor1\_Model06

final\_wf =   
 ames\_wflow %>%   
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 extract\_fit\_parsnip() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 2)



`