Project Phase2

Engel, Alec

3/6/2021

Load in data set

# summary(student)  
# glimpse(student)  
# skim(student)

Data cleaning to adjust for bad values, convert characters into factors and remove unnecessary variables.

student = student %>% dplyr::select(-X1, -Utilities, -Roof\_Matl, -Electrical, -Longitude, -Latitude, -Misc\_Feature, -Misc\_Val, -MS\_Zoning, -Condition\_2, Roof\_Style, -Kitchen\_AbvGr, -Roof\_Style, -Mas\_Vnr\_Type, -Mas\_Vnr\_Area, -Bsmt\_Cond, -Heating, -Sale\_Type, -Lot\_Frontage, -Alley, -Condition\_1, Exterior\_2nd, -Bsmt\_Exposure, -BsmtFin\_Type\_2, -Low\_Qual\_Fin\_SF, -Bsmt\_Half\_Bath, -Exterior\_2nd, -Fireplace\_Qu, -Pool\_Area) %>%   
 mutate\_if(is.character,as\_factor) %>%   
 mutate(Mo\_Sold = as\_factor(Mo\_Sold)) %>%  
 mutate(Mo\_Sold = fct\_recode(Mo\_Sold, "Jan" = "1", "Feb" = "2", "Mar" = "3", "Apr" = "4", "May" = "5", "Jun" = "6",   
 "Jul" = "7", "Aug" = "8", "Sep" = "9", "Oct" = "10", "Nov" = "11", "Dec" = "12")) %>%  
 mutate(BsmtFin\_SF\_1 = Total\_Bsmt\_SF - BsmtFin\_SF\_2 - Bsmt\_Unf\_SF)

Eliminate outliers

student = student %>%  
 filter(Lot\_Area < 40000) %>%  
 filter(BsmtFin\_SF\_1 < 1600) %>%  
 filter(BsmtFin\_SF\_2 < 400) %>%  
 filter(Bsmt\_Unf\_SF < 2250) %>%  
 filter(Total\_Bsmt\_SF < 2750) %>%  
 filter(Full\_Bath > 0) %>%  
 filter(Half\_Bath < 1.1) %>%  
 filter(First\_Flr\_SF < 2750) %>%  
 filter(Second\_Flr\_SF < 1400) %>%  
 filter(Gr\_Liv\_Area < 3750) %>%  
 filter(Fireplaces < 4) %>%  
 filter(Garage\_Cars < 4) %>%  
 filter(Garage\_Area < 1250) %>%  
 filter(Wood\_Deck\_SF < 550) %>%  
 filter(Open\_Porch\_SF < 350) %>%  
 filter(Enclosed\_Porch < 300) %>%  
 filter(Three\_season\_porch < 240) %>%  
 filter(Screen\_Porch < 400)

Train and test split

set.seed(123)  
student\_split = initial\_split(student, prob = 0.75, strata = Above\_Median)  
train = training(student\_split)  
test = testing(student\_split)

5 K Folds

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

Basic recipe

student\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_other(Neighborhood,threshold = .02) %>%  
 step\_other(MS\_SubClass,threshold = .02) %>%  
 step\_other(Overall\_Qual,threshold = .02) %>%  
 step\_other(Overall\_Cond,threshold = .02) %>%  
 step\_other(Exterior\_1st,threshold = .02) %>%  
 step\_other(Functional,threshold = .02) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) %>%  
 step\_nzv(all\_predictors())  
  
ctrl\_grid = control\_stack\_grid()  
ctrl\_res = control\_stack\_resamples()

***Student Log Regression***

# student\_log\_model =   
# logistic\_reg(mode = "classification") %>%  
# set\_engine("glm")  
#   
# student\_log\_recipe = student\_recipe %>%  
# step\_dummy(all\_nominal(), -all\_outcomes())  
#   
# logreg\_wf = workflow() %>%  
# add\_recipe(student\_log\_recipe) %>%   
# add\_model(student\_log\_model)  
#   
# set.seed(123)  
# log\_res =  
# tune\_grid(  
# logreg\_wf,  
# resamples = folds,  
# grid = 200,  
# control = ctrl\_grid  
# )

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

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

***Student Classification Tree Model***

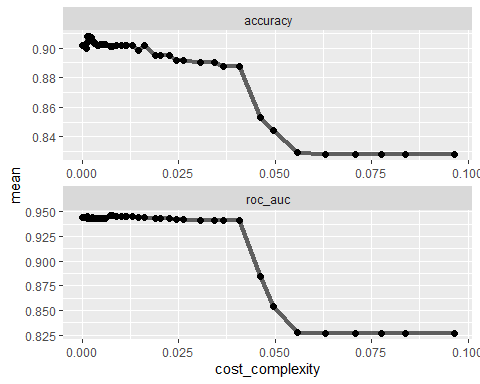
# tree\_model = decision\_tree(cost\_complexity = tune()) %>%  
# set\_engine("rpart", model = TRUE) %>%  
# set\_mode("classification")  
#   
# tree\_recipe = student\_recipe  
#   
# tree\_workflow = workflow() %>%  
# add\_model(tree\_model) %>%  
# add\_recipe(tree\_recipe)  
#   
# set.seed(123)  
# tree\_res =  
# tree\_workflow %>%  
# tune\_grid(  
# resamples = folds,  
# grid = 200,  
# control = ctrl\_grid  
# )

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

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

Classification Tree Model Accuracy Chart

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)



***Student Random Forest Model***

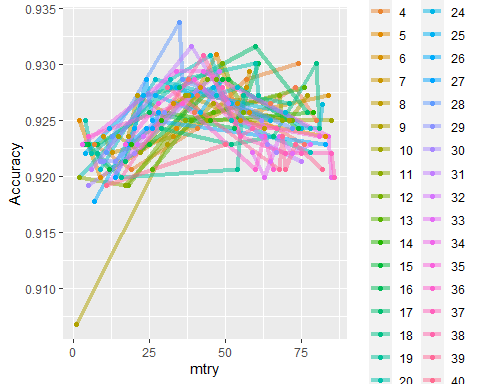
# rf\_recipe = student\_recipe  
#   
# rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 200) %>%  
# set\_engine("ranger", importance = "permutation") %>%  
# set\_mode("classification")  
#   
# rf\_wflow =  
# workflow() %>%  
# add\_model(rf\_model) %>%  
# add\_recipe(rf\_recipe)  
#   
# 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")

Random Forest Model Accuracy Chart

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")



***Student Neural Network Model***

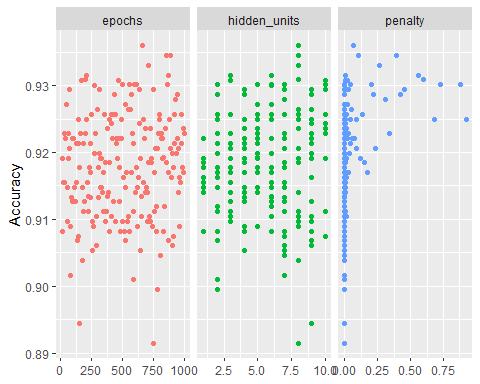
# nn\_recipe = student\_recipe %>%  
# step\_normalize(all\_predictors(), -all\_nominal()) %>%  
# step\_dummy(all\_nominal(), -all\_outcomes())  
#   
# nn\_model =  
# mlp(hidden\_units = tune(), penalty = tune(),  
# epochs = tune()) %>%  
# set\_mode("classification") %>%  
# set\_engine("nnet", verbose = 0)  
#   
# nn\_workflow <-  
# workflow() %>%  
# add\_recipe(nn\_recipe) %>%  
# add\_model(nn\_model)  
#   
# set.seed(123)  
# neural\_res <-  
# tune\_grid(nn\_workflow,  
# resamples = folds,  
# grid = 200,  
# control = ctrl\_grid)

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

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

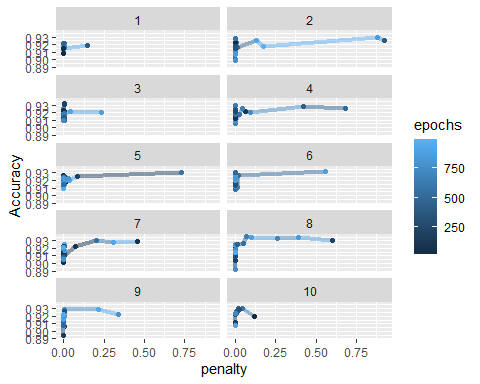
Neural Network Model Epochs/Hidden\_units/Penalties Charts #1

neural\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 dplyr::select(mean, hidden\_units, penalty, epochs) %>%  
 pivot\_longer(hidden\_units:epochs,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



Neural Network Model Epochs/Hidden\_units/Penalties Charts #2

neural\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(hidden\_units = factor(hidden\_units)) %>%  
 ggplot(aes(penalty, mean, color = epochs)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 facet\_wrap(~hidden\_units, ncol =2 ) +   
 labs(y = "Accuracy")



***Student Neural Network Model with Parameter Tuning***

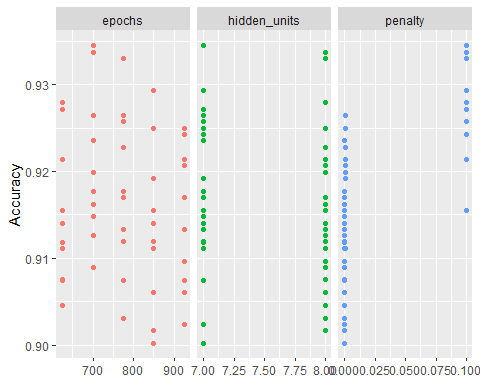
# neural\_grid = grid\_regular(  
# hidden\_units(range = c(7,8)),  
# penalty(range = c(-10,-1)),  
# epochs(range = c(625,925)),  
# levels = 5  
# )  
#   
# student\_nn\_recipe = student\_recipe %>%  
# step\_normalize(all\_predictors(), -all\_nominal()) %>%  
# step\_dummy(all\_nominal(), -all\_outcomes())  
#   
# student\_nn\_model =  
# mlp(hidden\_units = tune(), penalty = tune(),  
# epochs = tune()) %>%  
# set\_mode("classification") %>%  
# set\_engine("nnet", verbose = 0)  
#   
# student\_nn\_workflow <-  
# workflow() %>%  
# add\_recipe(student\_nn\_recipe) %>%  
# add\_model(student\_nn\_model)  
#   
# set.seed(123)  
# neural\_tune\_res <-  
# tune\_grid(student\_nn\_workflow, resamples = folds, grid = neural\_grid, control = ctrl\_grid)

# saveRDS(neural\_tune\_res,"neural\_tune\_res.rds")

neural\_tune\_res = readRDS("neural\_tune\_res.rds")

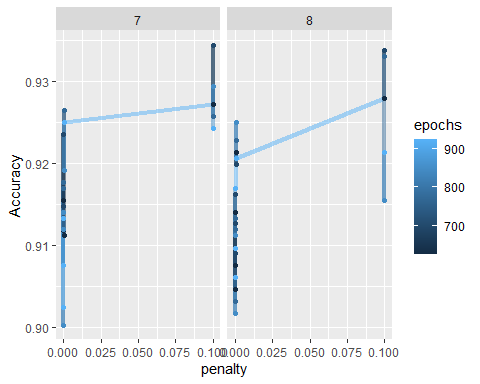
Neural Network Parameter Tuned Epochs/Hidden\_units/Penalties Charts

neural\_tune\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 dplyr::select(mean, hidden\_units, penalty, epochs) %>%  
 pivot\_longer(hidden\_units:epochs,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



Neural Network Parameter Tuned Model Epochs/Hidden\_units/Penalties Charts #2

neural\_tune\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(hidden\_units = factor(hidden\_units)) %>%  
 ggplot(aes(penalty, mean, color = epochs)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 facet\_wrap(~hidden\_units, ncol =2 ) +   
 labs(y = "Accuracy")



\***Student XGBOOST Model**

# xgboost\_recipe2 <- student\_recipe %>%  
# #step\_novel(all\_nominal(), -all\_outcomes()) %>%  
# step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%  
# step\_zv(all\_predictors())  
#   
# xgboost\_spec2 <-  
# 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\_workflow2 <-  
# workflow() %>%  
# add\_recipe(xgboost\_recipe2) %>%  
# add\_model(xgboost\_spec2)  
#   
# set.seed(123)  
# xgboost\_tune\_res <-  
# tune\_grid(xgboost\_workflow2, resamples = folds, grid = 200, control = ctrl\_grid)

# saveRDS(xgboost\_tune\_res,"xgboost\_tune\_res.rds")

xgboost\_tune\_res = readRDS("xgboost\_tune\_res.rds")

\***Student XGBOOST Model with Parameter Tuning**

# tgrid = expand.grid(  
# trees = 100,  
# min\_n = 1,  
# tree\_depth = c(1,2,3,4),  
# learn\_rate = c(0.01, 0.1, 0.2, 0.3, 0.4),  
# loss\_reduction = 0,  
# sample\_size = c(0.5, 0.8, 1))  
#   
# xgboost\_recipe <-  
# student\_recipe %>%  
# 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(123)  
# xgb\_res <-  
# tune\_grid(xgboost\_workflow,  
# resamples = folds,  
# grid = tgrid,  
# control = ctrl\_grid)

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

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

***Stacking*** Building Stack

student\_stacks = stacks() %>%  
 add\_candidates(tree\_res) %>%  
 add\_candidates(rf\_res) %>%   
 add\_candidates(xgb\_res) %>%  
 add\_candidates(neural\_tune\_res) %>%  
 add\_candidates(log\_res)  
 # add\_candidates(neural\_res)  
 # add\_candidates(xgboost\_tune\_res)

fitting a Lasso model to the stack.

student\_blend =   
 student\_stacks %>%   
 blend\_predictions(metric = metric\_set(accuracy))

##   
## 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

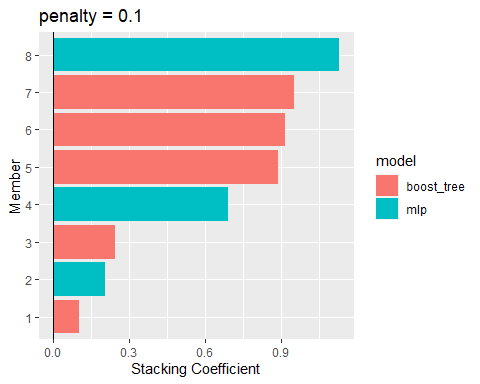
##   
## Attaching package: 'vctrs'

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

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

Strongest Resulting Models

autoplot(student\_blend, type = "weights")



Fitting the stack to training data

#student\_blend <- # student\_blend %>% # fit\_members()

Training data predictions

trainpredstack = predict(student\_blend, train)  
head(trainpredstack)

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

Training data confusion matrix

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 676 3  
## No 4 690  
##   
## Accuracy : 0.9949   
## 95% CI : (0.9895, 0.9979)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9898   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9941   
## Specificity : 0.9957   
## Pos Pred Value : 0.9956   
## Neg Pred Value : 0.9942   
## Prevalence : 0.4953   
## Detection Rate : 0.4924   
## Detection Prevalence : 0.4945   
## Balanced Accuracy : 0.9949   
##   
## 'Positive' Class : Yes   
##

Test data predictions

testpredstack = predict(student\_blend, test)  
head(testpredstack)

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

Test data confusion matrix

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 204 18  
## No 22 213  
##   
## Accuracy : 0.9125   
## 95% CI : (0.8827, 0.9367)  
## No Information Rate : 0.5055   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8249   
##   
## Mcnemar's Test P-Value : 0.6353   
##   
## Sensitivity : 0.9027   
## Specificity : 0.9221   
## Pos Pred Value : 0.9189   
## Neg Pred Value : 0.9064   
## Prevalence : 0.4945   
## Detection Rate : 0.4464   
## Detection Prevalence : 0.4858   
## Balanced Accuracy : 0.9124   
##   
## 'Positive' Class : Yes   
##

Comparison of model performance on test set

test = test %>% bind\_cols(predict(student\_blend,.))

Stacked model compared to constituent models

member\_testpreds =   
 test %>%  
 dplyr::select(Above\_Median) %>%  
 bind\_cols(predict(student\_blend, test, members = TRUE))

map\_dfr(member\_testpreds, accuracy, truth = Above\_Median, data = member\_testpreds) %>%  
 mutate(member = colnames(member\_testpreds))

## # A tibble: 10 x 4  
## .metric .estimator .estimate member   
## <chr> <chr> <dbl> <chr>   
## 1 accuracy binary 1 Above\_Median   
## 2 accuracy binary 0.912 .pred\_class   
## 3 accuracy binary 0.906 .pred\_class\_xgb\_res\_1\_11   
## 4 accuracy binary 0.906 .pred\_class\_xgb\_res\_1\_15   
## 5 accuracy binary 0.908 .pred\_class\_xgb\_res\_1\_24   
## 6 accuracy binary 0.899 .pred\_class\_xgb\_res\_1\_27   
## 7 accuracy binary 0.902 .pred\_class\_xgb\_res\_1\_53   
## 8 accuracy binary 0.893 .pred\_class\_neural\_tune\_res\_1\_13  
## 9 accuracy binary 0.906 .pred\_class\_neural\_tune\_res\_1\_27  
## 10 accuracy binary 0.904 .pred\_class\_neural\_tune\_res\_1\_20

Implementation of stack on competition set

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

# competition = competition %>%  
# mutate\_if(is.character,as\_factor) %>%   
# mutate(Mo\_Sold = as\_factor(Mo\_Sold)) %>%  
# mutate(Mo\_Sold = fct\_recode(Mo\_Sold, "Jan" = "1", "Feb" = "2", "Mar" = "3", "Apr" = "4", "May" = "5", "Jun" = "6",   
# "Jul" = "7", "Aug" = "8", "Sep" = "9", "Oct" = "10", "Nov" = "11", "Dec" = "12")) %>%  
# mutate(BsmtFin\_SF\_1 = Total\_Bsmt\_SF - BsmtFin\_SF\_2 - Bsmt\_Unf\_SF)

# competitionpredstack = predict(student\_blend, competition)  
# head(competitionpredstack)

# kaggle = competition %>% dplyr::select(X1)  
#   
# kaggle = bind\_cols(kaggle, competitionpredstack)  
#   
# kaggle

# write.csv(kaggle, "kaggle\_submit4.csv", row.names=FALSE)