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

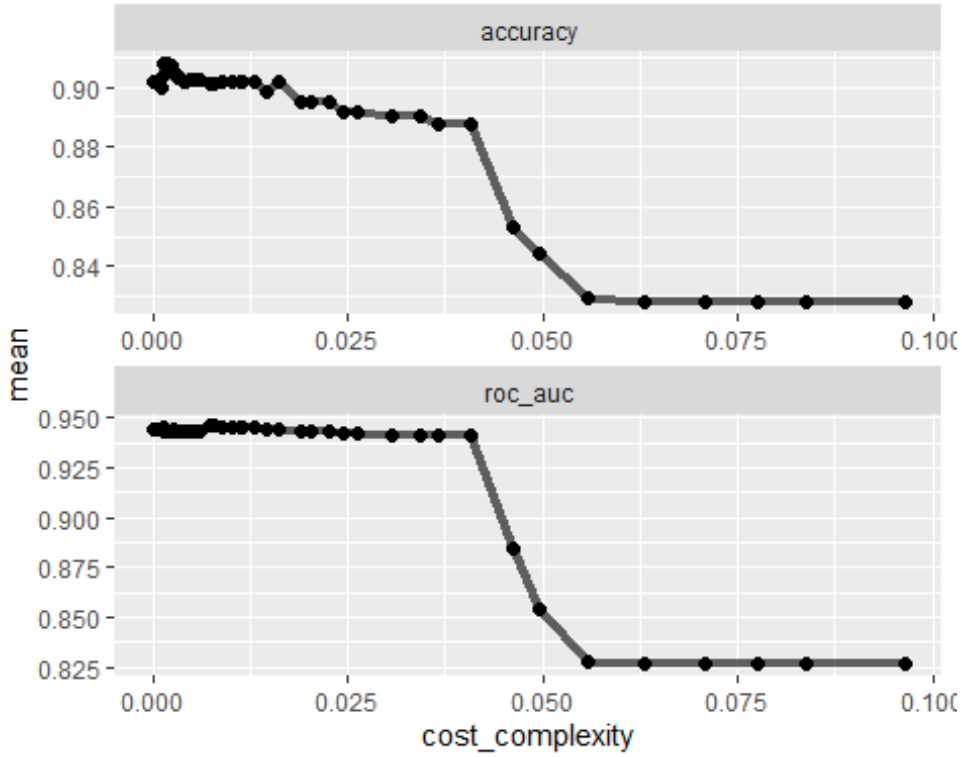
```
# tree_model = decision_tree(cost_complexity = tune()) %>%
#   set_engine("rpart", model = TRUE) %>%
#   set_mode("classification")
#
# tree_recipe = student_recipe
#
# tree_workflow = workflow() %>%
#   add_model(tree_model) %>%
#   add_recipe(tree_recipe)
#
# set.seed(123)
# tree_res =
#   tree_workflow %>%
#   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

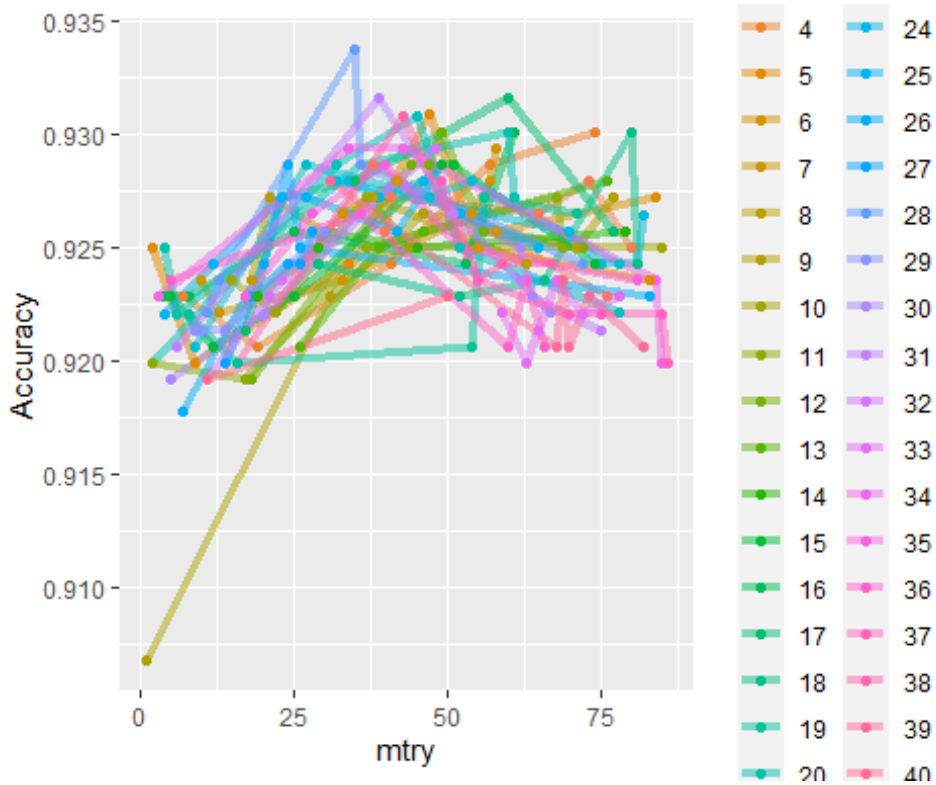
```
# rf_recipe = student_recipe
#
# rf_model = rand_forest(mtry = tune(), min_n = tune(), trees = 200) %>%
#   set_engine("ranger", importance = "permutation") %>%
#   set_mode("classification")
#
# rf_wflow =
#   workflow() %>%
#   add_model(rf_model) %>%
#   add_recipe(rf_recipe)
#
# set.seed(123)
# 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)
```

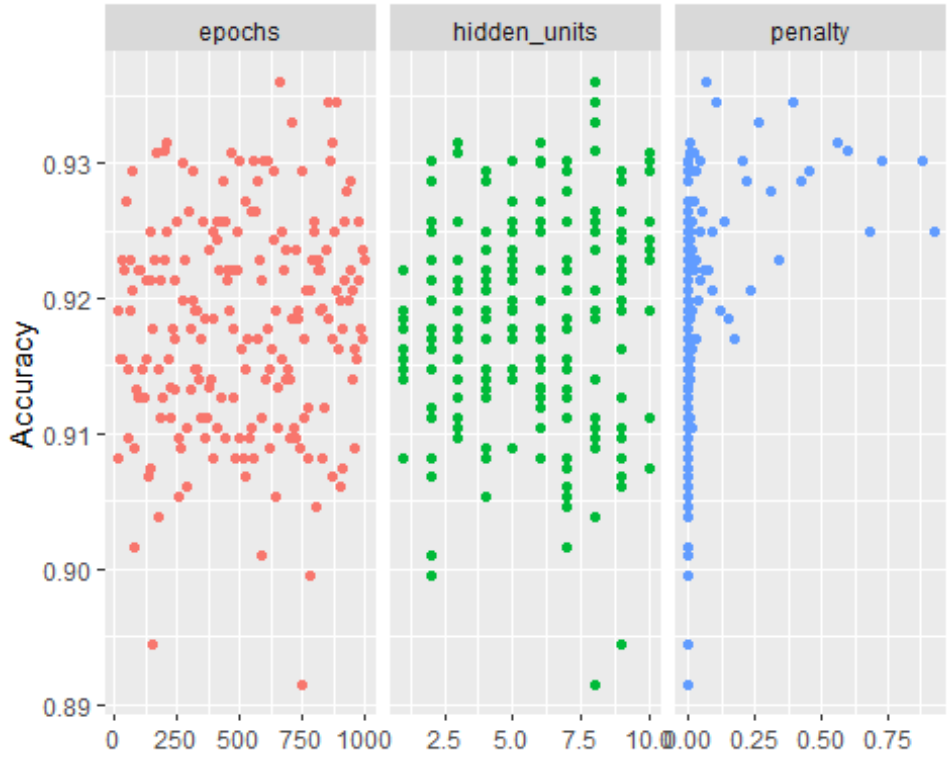
```
# neural_res <-
#   tune_grid(nn_workflow,
#             resamples = folds,
#             grid = 200,
#             control = ctrl_grid)

# saveRDS(neural_res,"neural_res.rds")

neural_res = readRDS("neural_res.rds")
```

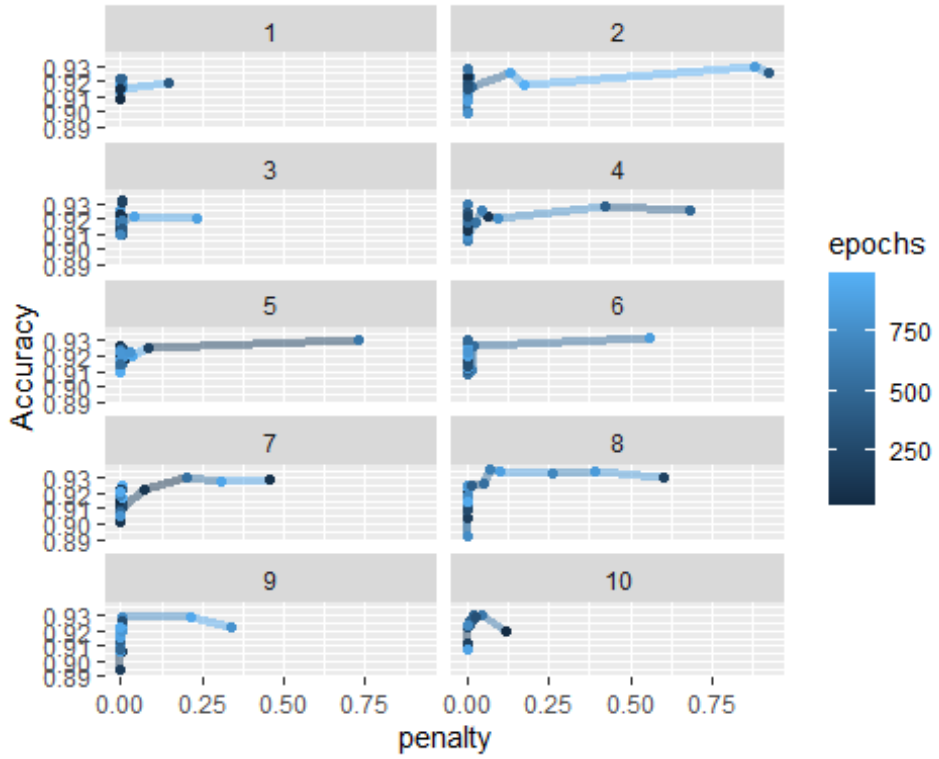
Neural Network Model Epochs/Hidden_units/Penalties Charts #1

```
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") %>%
```

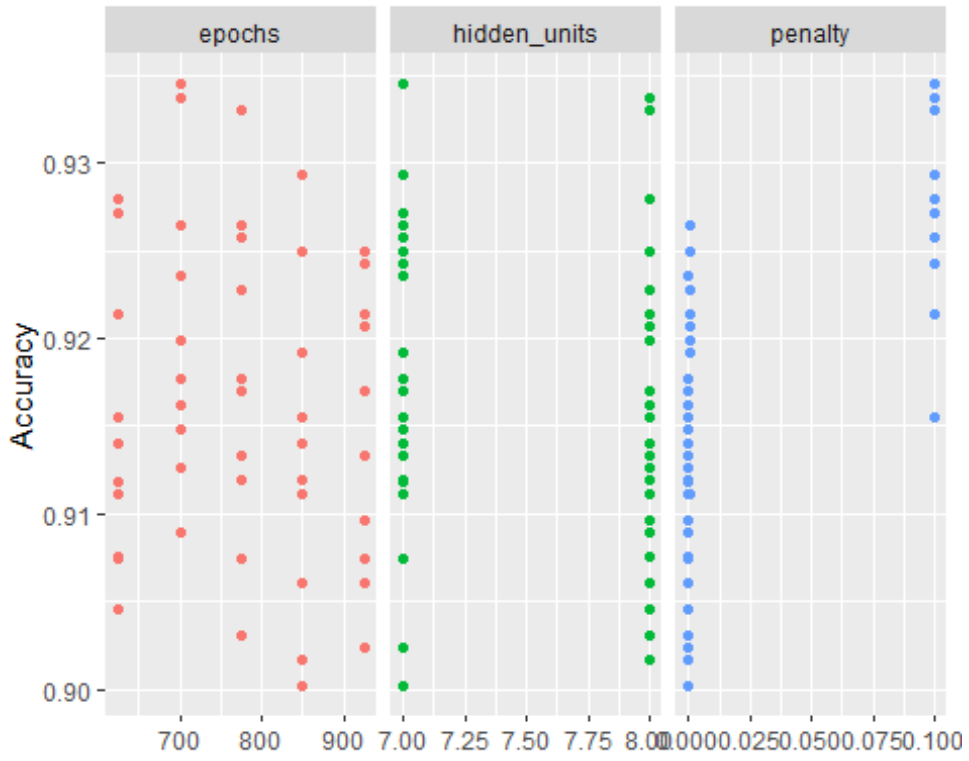
```
#   set_engine("nnet", verbose = 0)
#
# student_nn_workflow <-
#   workflow() %>%
#   add_recipe(student_nn_recipe) %>%
#   add_model(student_nn_model)
#
# set.seed(123)
# neural_tune_res <-
#   tune_grid(student_nn_workflow, resamples = folds, grid = neural_grid,
control = ctrl_grid)

# saveRDS(neural_tune_res,"neural_tune_res.rds")

neural_tune_res = readRDS("neural_tune_res.rds")
```

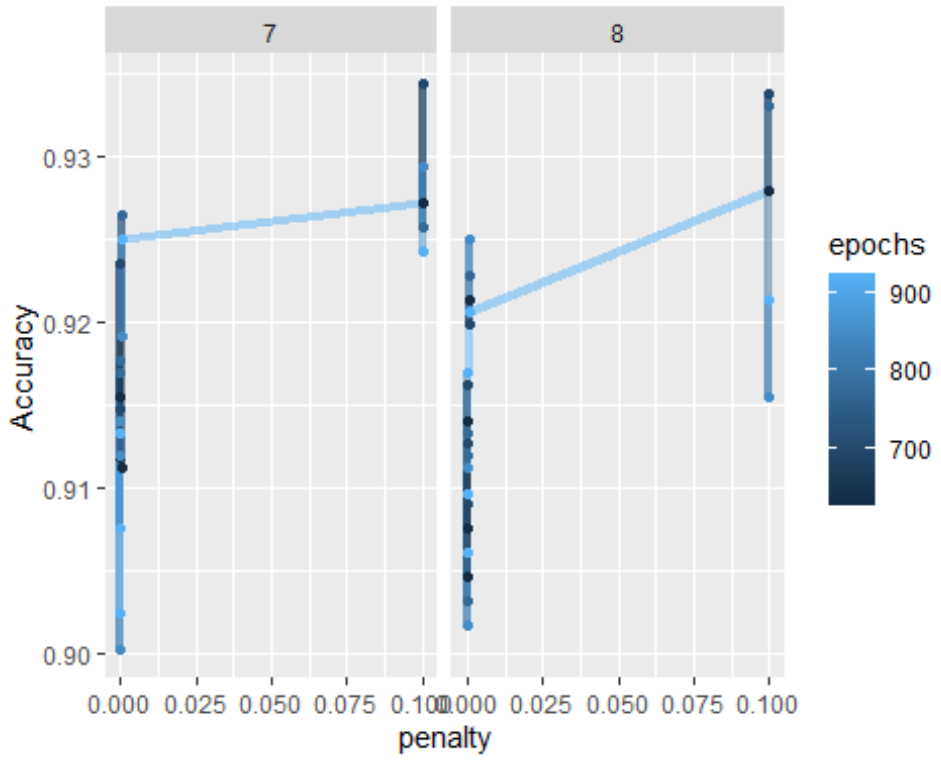
Neural Network Parameter Tuned Epochs/Hidden_units/Penalties Charts

```
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
```

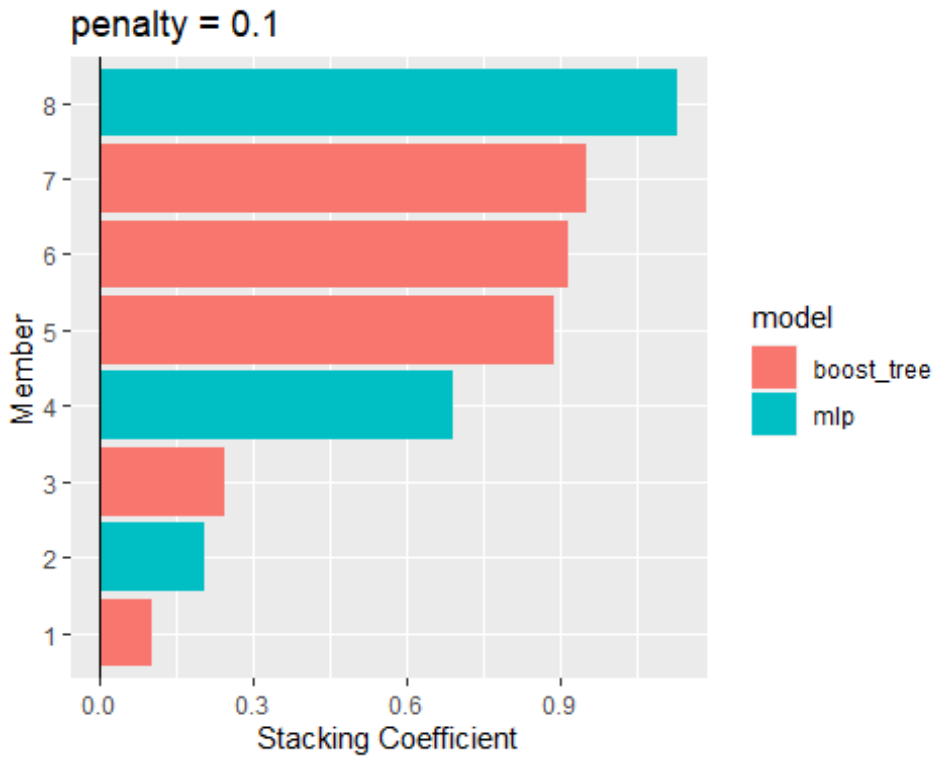
```
##
## Attaching package: 'vctrs'
```

```
## 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)
```