Competition Final

Alex Federspiel and Evan Miu

3/30/2021

```r
retrain_model <- FALSE
if(!retrain_model) {
  load("competition.Rdata")
}
```

Loading necessary packages, plus initiating x10 core parallel compute to speed up training of each model.

Reading in initial training data set, which includes descriptors alongside outcomes.

Preprocessing data using centering and scaling, as well as Box-Cox transformation if deemed appropropriate by the preProcess function.

```r
df_competition <- data.frame(competition_train)
predictors_comp_only <- df_competition %>% select(-outcome)
preprocess_fit <- preProcess(predictors_comp_only,
                             method = c("BoxCox", "center", "scale"))
preprocess_train <- predict(preprocess_fit, predictors_comp_only)
```

Retaining only those predictors that adhere to a cutoff Pearson correlation of 0.90.

```r
predictors_only_correlation_matrix <- cor(preprocess_train)
cutoff <- 0.90

names_of_predictors_to_remove <- findCorrelation(predictors_only_correlation_matrix,
                                                 names = TRUE, cutoff = cutoff)
removed_predictors <- preprocess_train %>%
select(-all_of(names_of_predictors_to_remove))
```

Setting up both data and folding for cross validation and model training.

```r
x <- data.frame(removed_predictors)
sum(is.na(x))
```

```
## [1] 0
```

```r
y <- competition_train$outcome
sum(is.na(y))
```

```
## [1] 0
```

```r
ctrl <- trainControl(method = "cv", number = 10)
```

Trying a simple non-regularized linear regression model.

```r
if (retrain_model) {
  set.seed(42)
  linear_regression <- train(x = x, y = y, method = "lm",
                             trControl = ctrl)
```

```
}
min(linear_regression$results$RMSE)
```

```
## [1] 10.43507
```

Maybe some regularization would help. Trying an L1-norm regularized linear regression model.

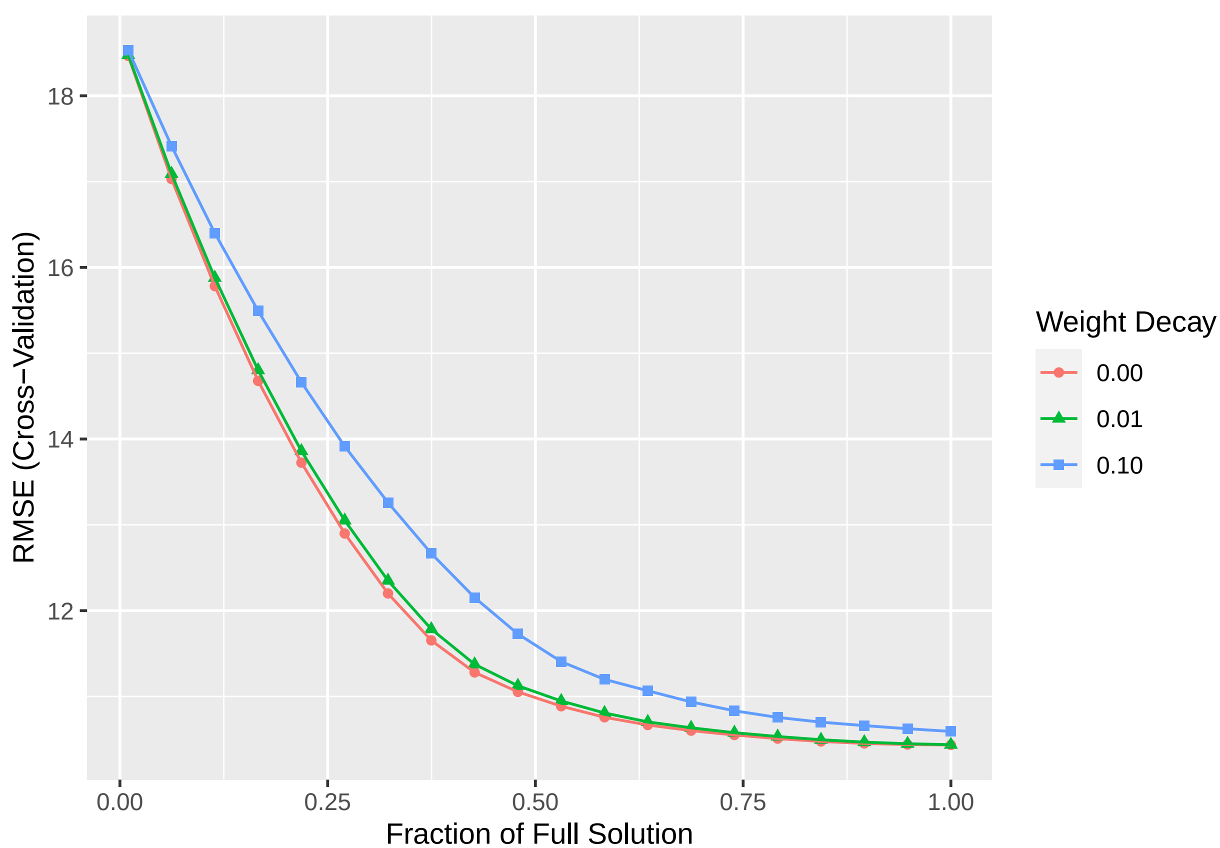
```
if (retrain_model) {
  set.seed(42)
  lassoGrid <- expand.grid(lambda = c(0,.01,.1),
                           fraction = seq(0.01, 1, length = 20))
  LASSO <- train(x =x, y = y, method = "enet",
                 tuneGrid = lassoGrid, trControl = ctrl)
}
min(LASSO$results$RMSE)
```

```
## [1] 10.43507
```

```
LASSO$bestTune
```

```
##    fraction lambda
## 20        1      0
```

```
ggplot(LASSO)
```



Model does not do well, need another. Trying an L2-norm regularized linear regression model.

```
if (retrain_model) {
  set.seed(42)
  ridgeGrid <- expand.grid(lambda = seq(0, 1, length = 30))
```

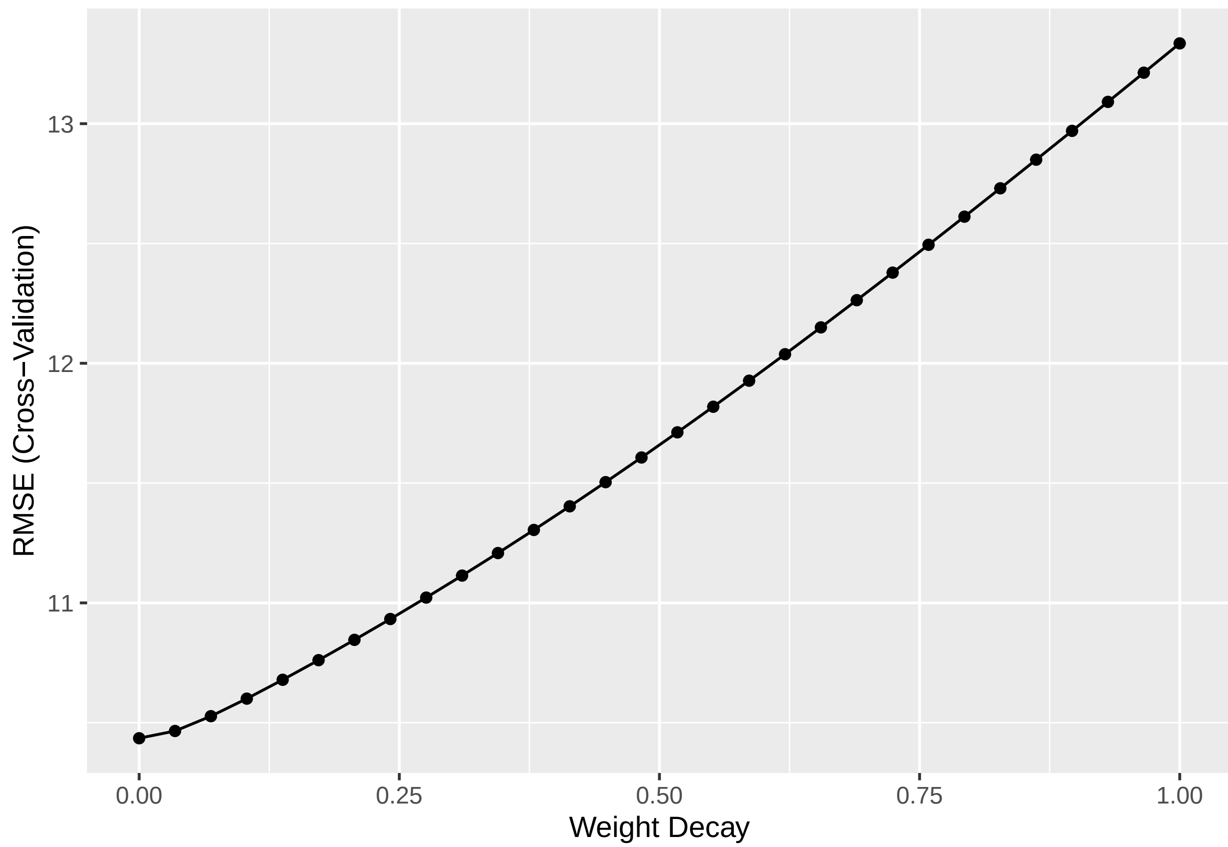
```r
  ridge_regression <- train(x = x, y = y, method = "ridge",
                            tuneGrid = ridgeGrid, trControl = ctrl)
}
min(ridge_regression$results$RMSE)
```

```
## [1] 10.43507
```

```r
ridge_regression$bestTune
```

```
##   lambda
## 1      0
```

```r
ggplot(ridge_regression)
```



Regularized linear models do not seem to be appropriate here. Trying regression trees model.

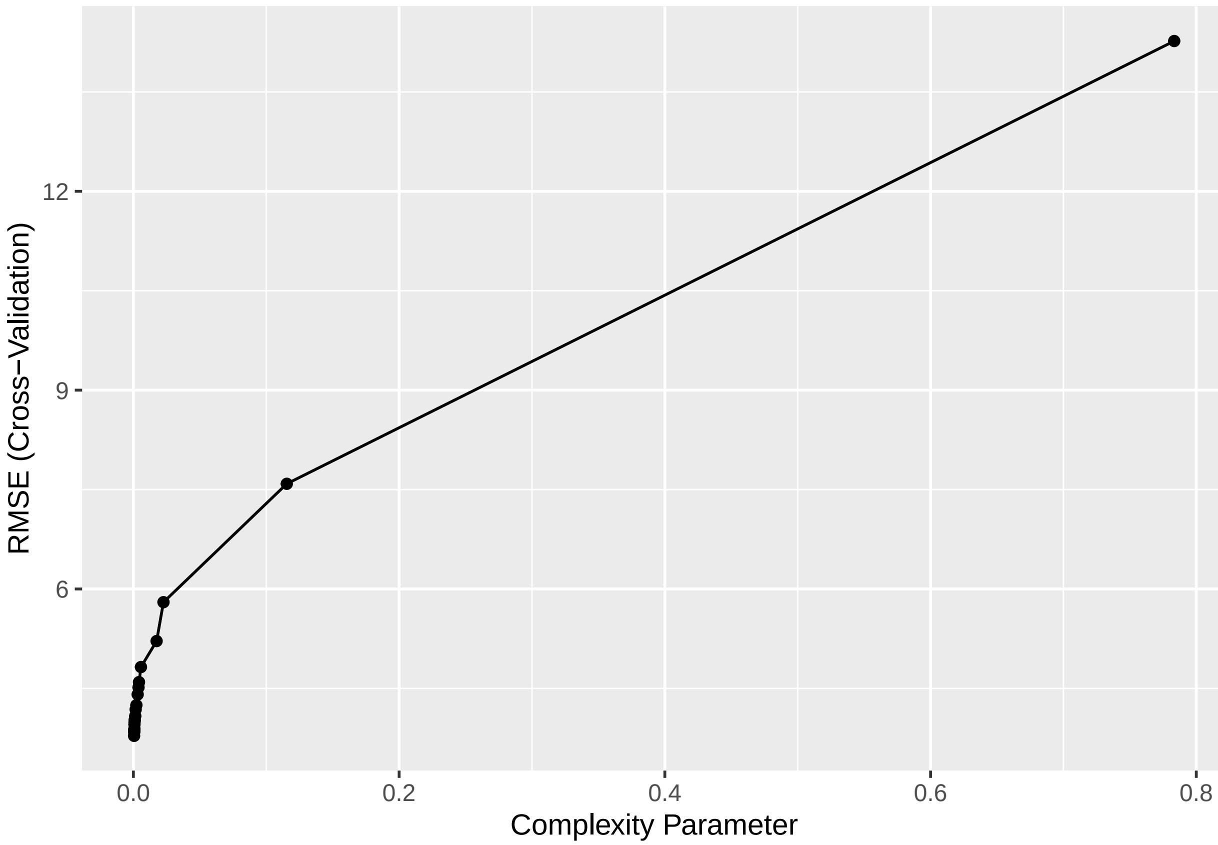
```r
if (retrain_model) {
  set.seed(42)
  regression_tree <- train(x = x, y = y, method = "rpart",
                           tuneLength = 20, trControl = ctrl)
}
min(regression_tree$results$RMSE)
```

```
## [1] 3.783559
```

```r
regression_tree$bestTune
```

```
##             cp
## 1 0.0005315864
```

```r
ggplot(regression_tree)
```



RMSE is still high even for low misclassification tolerance. Trying a random forest model.

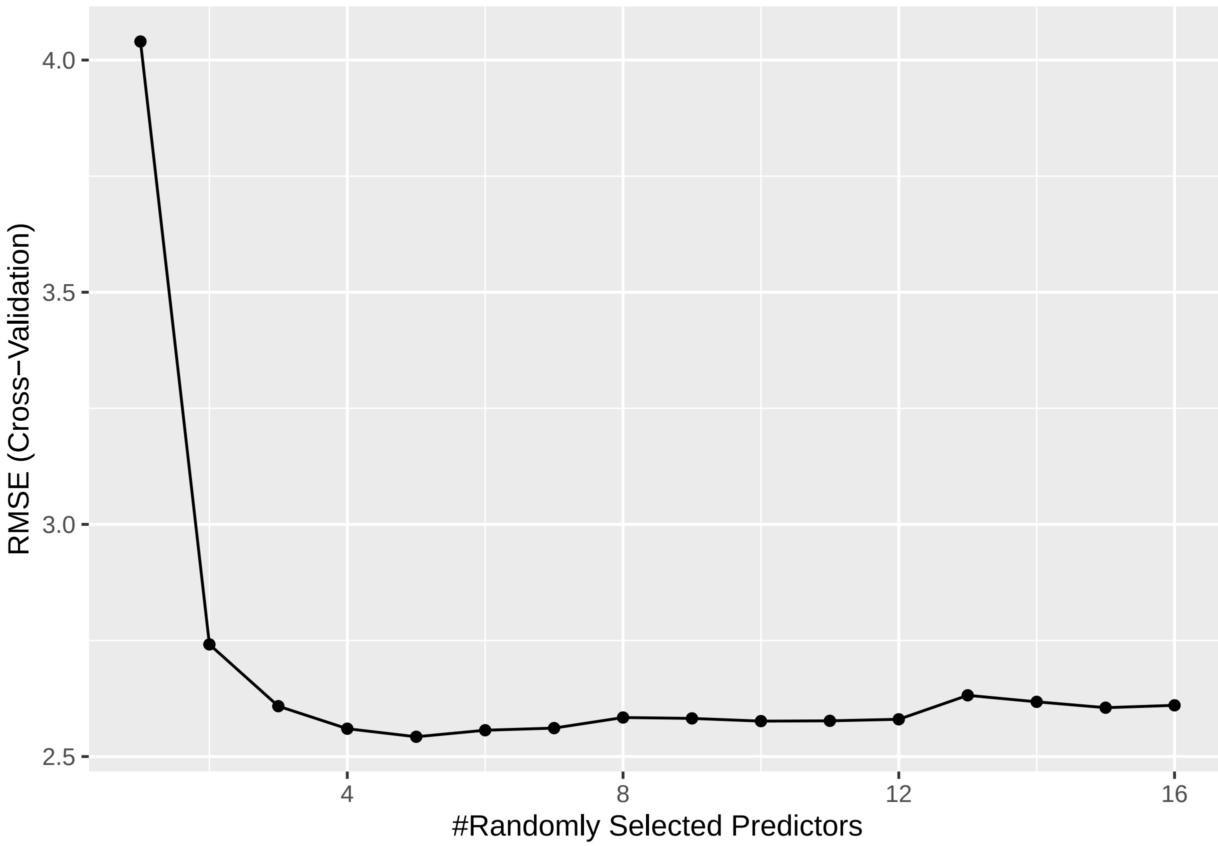
```r
if (retrain_model) {
  ##takes 8 years to run
  set.seed(42)
  mtryGrid <- data.frame(mtry = floor(seq(1, ncol(x), length = 20)))
  random_forest <- train(x = x, y = y, method = "rf",
                                    tuneGrid = mtryGrid, ntree = 100, importance = TRUE,
                                trControl = ctrl)
}
min(random_forest$results$RMSE)
```

```
## [1] 2.542564
```

```r
random_forest$bestTune
```

```
##   mtry
## 5    5
```

```r
ggplot(random_forest)
```



Model does well, but tops off above the target 2.5 RMSE value. Trying a suport vector machine model.

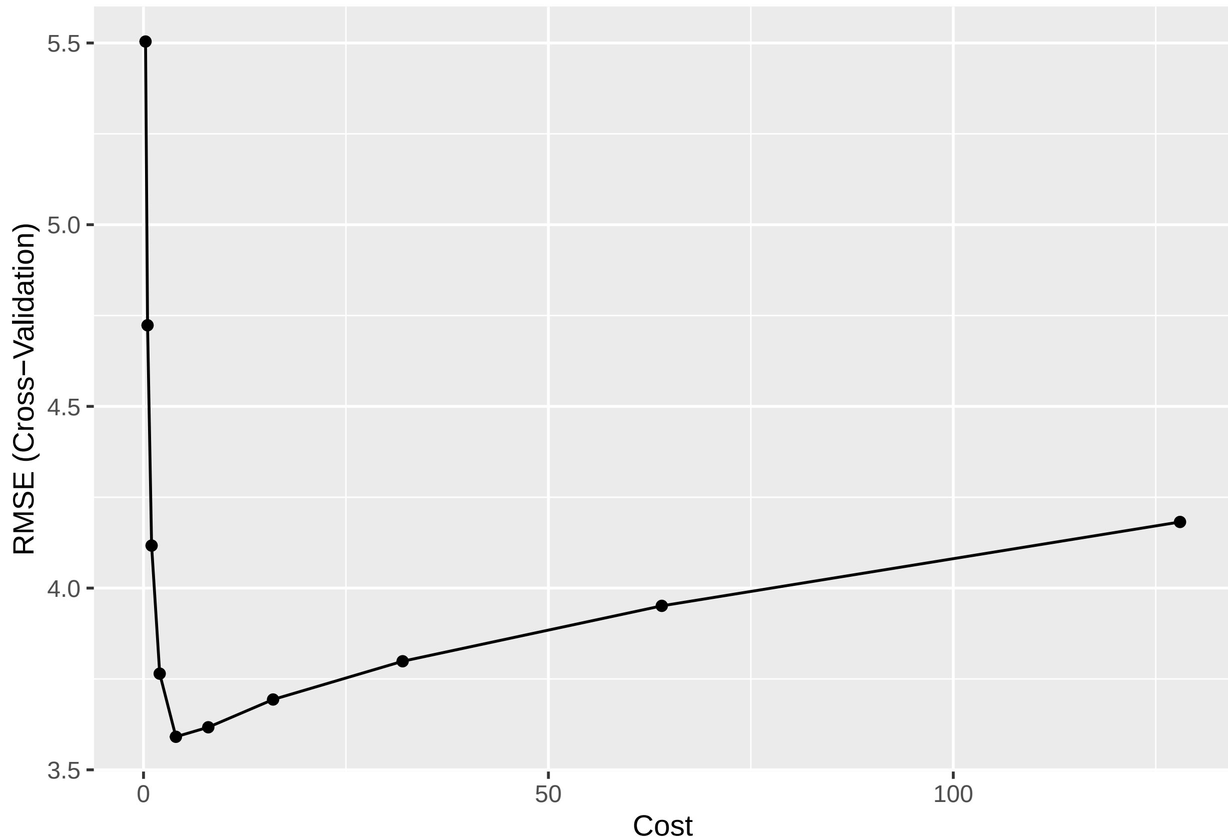
```
if (retrain_model) {
  ##takes 20 years
  set.seed(42)
  SVM <- train(x = x, y = y, method = "svmRadial",
               tuneLength = 10, epsilon = 0.01, trControl = ctrl)
}
min(SVM$results$RMSE)
```

```
## [1] 3.590872
```

```
SVM$bestTune
```

```
##        sigma C
## 5 0.07354022 4
```

```
ggplot(SVM)
```



The random forest outperformed this initial SVM try. Trying a simple neural network.

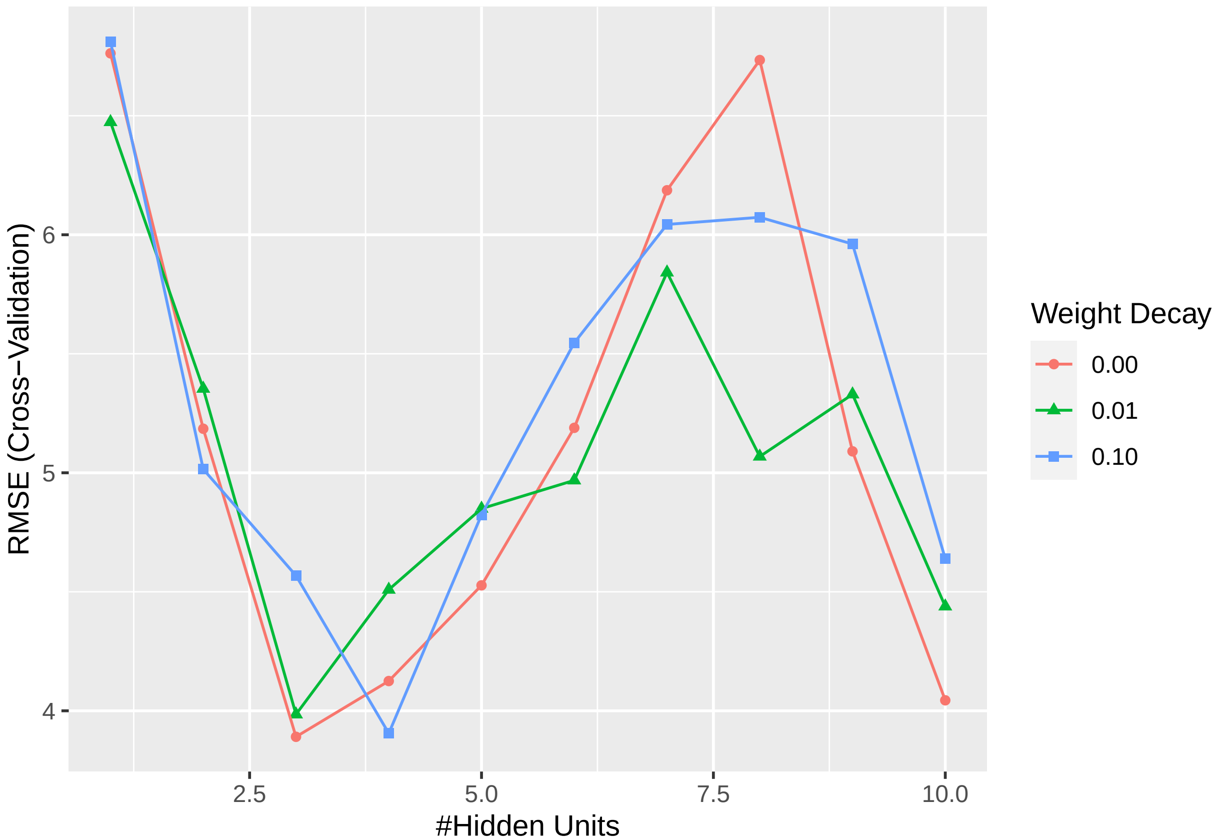
```
if (retrain_model) {
  ##takes 8 years
  set.seed(42)
  nnetGrid <- expand.grid(decay = c(0, 0.01, 0.1), size = 1:10, bag = FALSE)
  neural_net <- train(x = x, y = y, method = "avNNet",
                        tuneGrid = nnetGrid, trControl = ctrl, linout = TRUE,
                        trace = FALSE)
}
min(neural_net$results$RMSE)
```

```
## [1] 3.890814
```

```
neural_net$bestTune
```

```
##   size decay   bag
## 3    3     0 FALSE
```

```
ggplot(neural_net)
```



Random forest still more promising. Trying a MARS adaptive model.

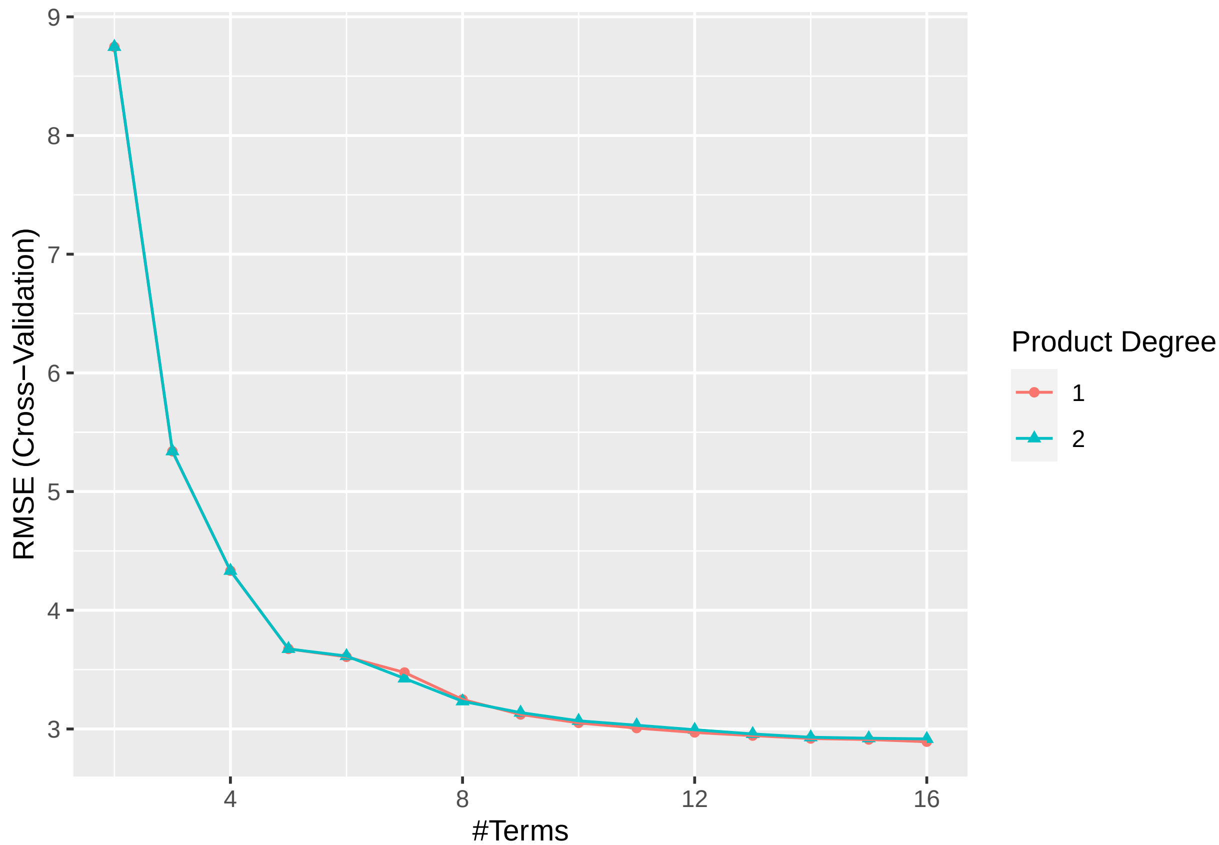
```
if (retrain_model) {
  set.seed(42)
  marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:16)
  MARS <- train(x = x, y = y, method = "earth",
                    tuneGrid = marsGrid, trControl = ctrl)
}
min(MARS$results$RMSE)
```

```
## [1] 2.892685
```

```
MARS$bestTune
```

```
##    nprune degree
## 15     16      1
```

```
ggplot(MARS)
```



Close to random forest, but still not better. Trying a k-nearest neighbors model.

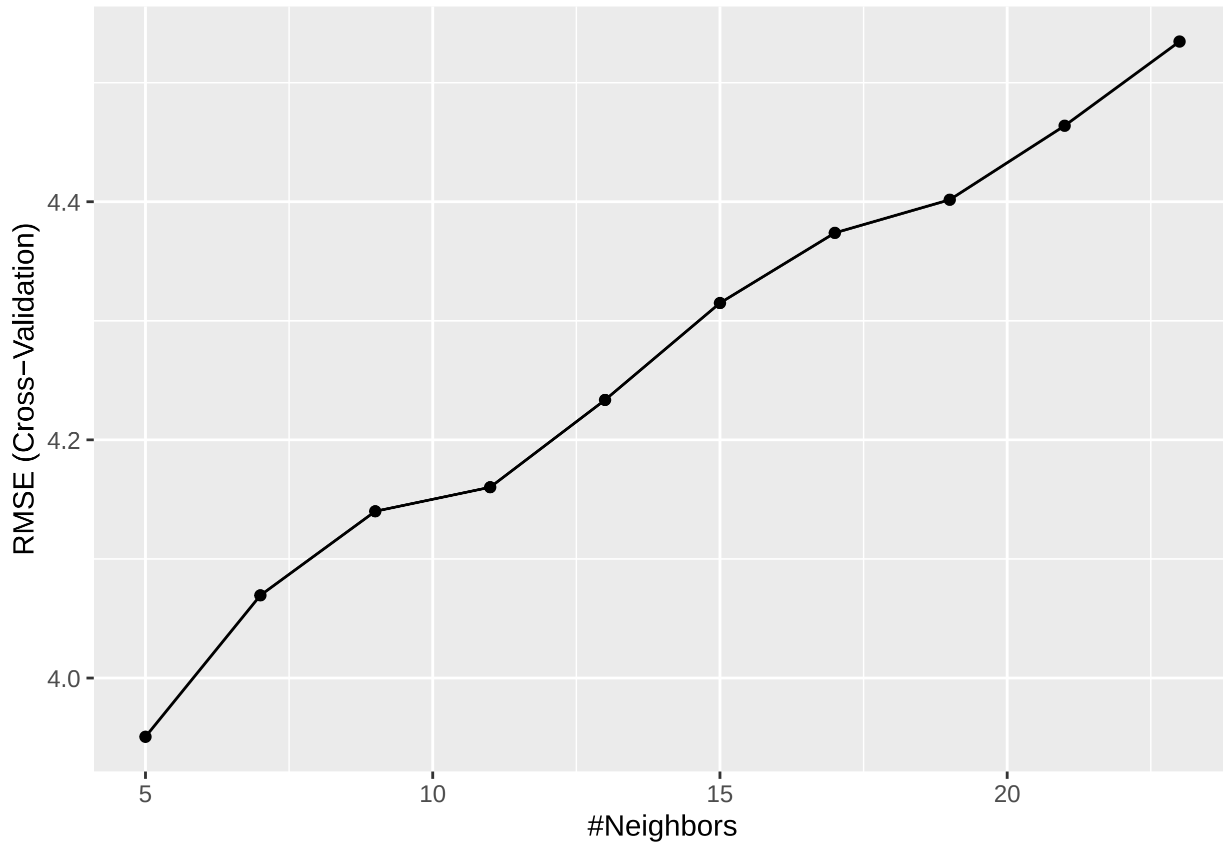
```
if (retrain_model) {
  set.seed(42)
  KNN <- train(x = x, y = y, method = "knn", trControl = ctrl,
                   tuneLength = 10)
}
min(KNN$results$RMSE)
```

```
## [1] 3.950686
```

```
KNN$bestTune
```

```
##   k
## 1 5
```

```
ggplot(KNN)
```



If the trend continues, KNN would not improve much below an RMSE of 4.0. Next we try a gradient boosted trees model, which can do better than the random forest by sequentially training new trees to characterize missed data from the rpevious tree. Trying a gradient boosted trees model.

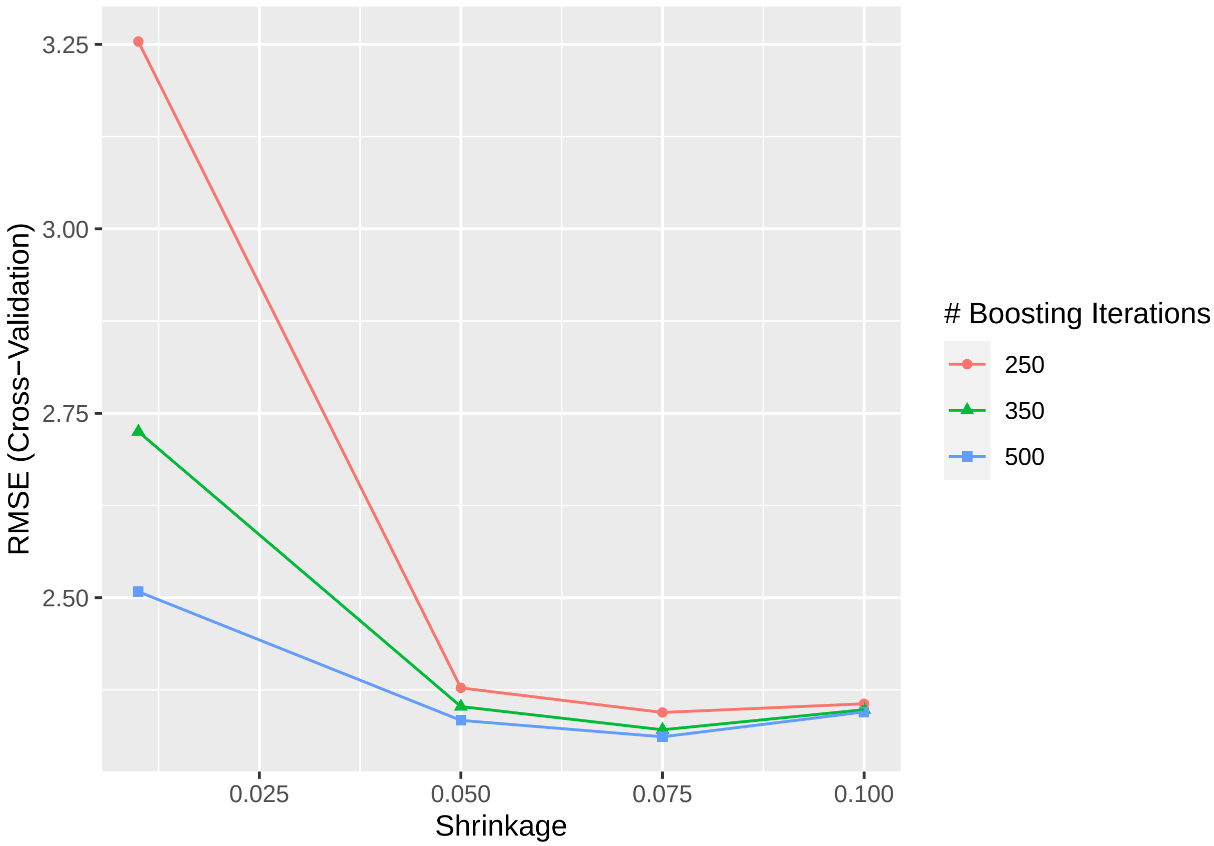
```
if (retrain_model) {
  set.seed(42)
  gbmGrid <- expand.grid(n.trees = c(250,350,500),
             interaction.depth = 10,
             n.minobsinnode = 10,
             shrinkage = c(0.01,0.05,0.075,0.1))
  gbm <- train(x = x, y = y, distribution = "gaussian",
             method = "gbm", verbose = FALSE,
             trControl = ctrl,
             tuneGrid = gbmGrid)
}
min(gbm$results$RMSE)
```

```
## [1] 2.311403
```

```
gbm$bestTune
```

```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 9     500                10     0.075             10
```

```
ggplot(gbm)
```



Out of all models, the GBM seems to perform the best. Time to try on the validation set to see brand new data.

```
Method <- c("Linear Regression", "LASSO", "Ridge Regression",
            "Regression Tree", "Random Forest", "SVM", "Neural Net", "MARS",  "Nearest Neighbor", "GBM

RMSE <- c(signif(linear_regression$results$RMSE, digits = 4),
          signif(min(LASSO$results$RMSE), digits = 4),
          signif(min(ridge_regression$results$RMSE), digits = 4),
          signif(min(regression_tree$results$RMSE), digits = 4),
          signif(min(random_forest$results$RMSE), digits = 4),
          signif(min(SVM$results$RMSE), digits = 4),
          signif(min(neural_net$results$RMSE), digits = 4),
          signif(min(MARS$results$RMSE), digits = 4),
          signif(min(KNN$results$RMSE), digits = 4),
          signif(min(gbm$results$RMSE), digits = 4))

compare.df <- data.frame(Method, RMSE)
compare.df %>% arrange(RMSE)
```

```
##               Method  RMSE
## 1                GBM 2.311
## 2      Random Forest 2.543
## 3               MARS 2.893
## 4                SVM 3.591
## 5    Regression Tree 3.784
## 6         Neural Net 3.891
## 7   Nearest Neighbor 3.951
```

```
## 8  Linear Regression 10.440
## 9              LASSO 10.440
## 10  Ridge Regression 10.440
```

Generating the preprocessed predictors / outcome from the new validation set for final check on model performance.

```
validation_data <- read.csv("/home/evm/coursework/CourseContent/Assignments/competition/competition-vali
predictors_val_only <- data.frame(validation_data %>% select(-outcome))
preprocess_validation <- predict(preprocess_fit, predictors_val_only)

removed_predval <- preprocess_validation %>%
  select(-all_of(names_of_predictors_to_remove))
x_val <- data.frame(removed_predval)
sum(is.na(x_val))
```

```
## [1] 0
```

```
y_val <- validation_data$outcome
sum(is.na(y_val))
```

```
## [1] 0
```

Applying the gradient boosted trees model to the transformed data.

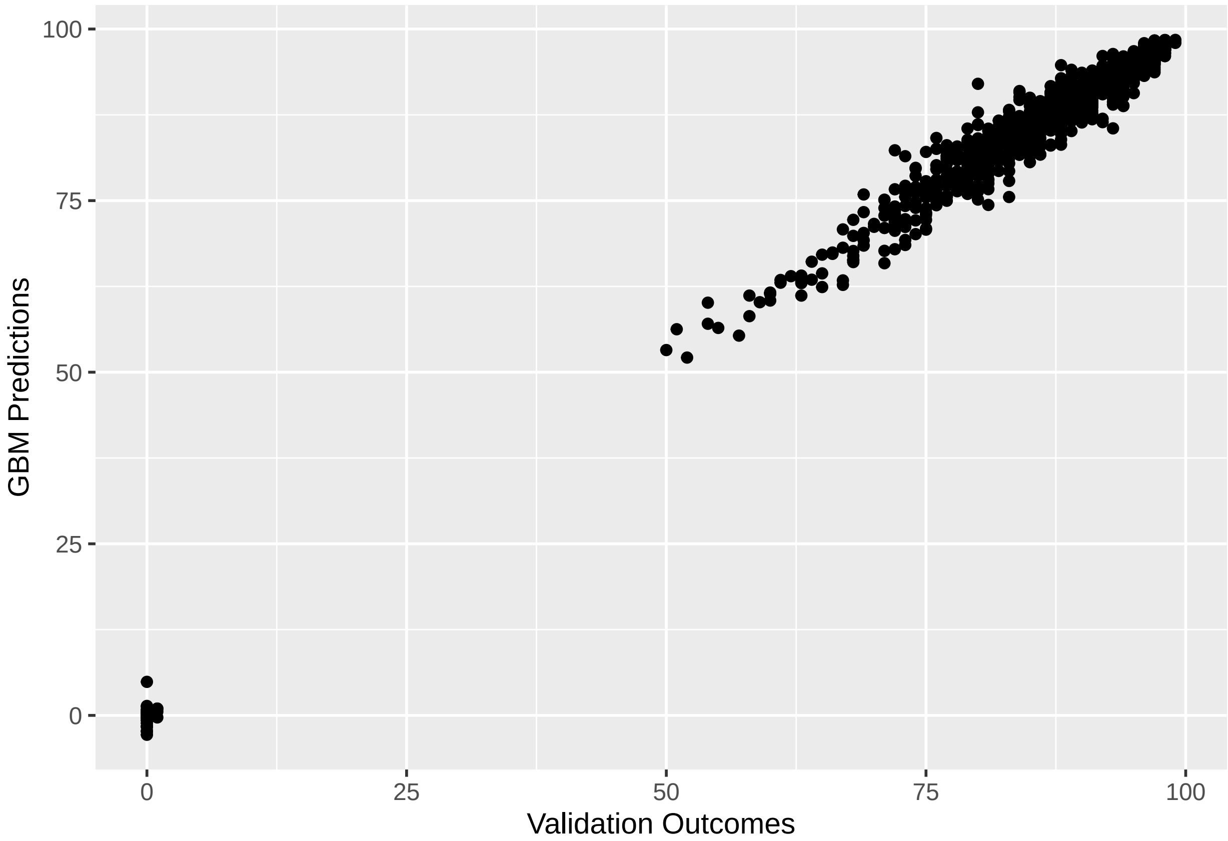
```
linear_check <- predict(linear_regression,x_val)
linear_RMSE <- RMSE(linear_check,y_val)
LASSO_check <- predict(LASSO,x_val)
LASSO_RMSE <- RMSE(LASSO_check,y_val)
ridge_check <- predict(ridge_regression,x_val)
ridge_RMSE <- RMSE(ridge_check,y_val)
rtree_check <- predict(regression_tree,x_val)
rtree_RMSE <- RMSE(rtree_check,y_val)
rf_check <- predict(random_forest,x_val)
rf_RMSE <- RMSE(rf_check,y_val)
SVM_check <- predict(SVM,x_val)
SVM_RMSE <- RMSE(SVM_check,y_val)
nn_check <- predict(neural_net,x_val)
nn_RMSE <- RMSE(nn_check,y_val)
MARS_check <- predict(MARS,x_val)
MARS_RMSE <- RMSE(MARS_check,y_val)
KNN_check <- predict(KNN,x_val)
KNN_RMSE <- RMSE(KNN_check,y_val)
gbm_check <- predict(gbm,x_val)
gbm_RMSE <- RMSE(gbm_check,y_val)
RMSE.check <- c(linear_RMSE,LASSO_RMSE,ridge_RMSE,
                   rtree_RMSE,rf_RMSE,SVM_RMSE,nn_RMSE,
                   MARS_RMSE,KNN_RMSE,gbm_RMSE)
check.df <- data.frame(Method, RMSE.check)
check.df %>% arrange(RMSE.check)
```

```
##                Method RMSE.check
## 1                 GBM   2.194355
## 2       Random Forest   2.407017
## 3                MARS   2.770861
## 4                 SVM   3.417531
## 5     Regression Tree   3.553377
```

```
## 6   Nearest Neighbor   3.866759
## 7         Neural Net   4.647964
## 8  Linear Regression  10.201972
## 9              LASSO  10.201972
## 10  Ridge Regression  10.201972
```

```
gbm.df <- data.frame(y_val,gbm_check)

ggplot(data=gbm.df,aes(x=y_val,gbm_check)) + geom_point() +
  xlab("Validation Outcomes") + ylab("GBM Predictions")
```



Best model remains GBM after checking with validation set. Generating our final predictions for the competition.

```
final_data <- read.csv("/home/evm/coursework/CourseContent/Assignments/competition/competition-test-x-va
preprocess_final <- predict(preprocess_fit, final_data)

removed_predfin <- preprocess_final %>%
  select(-all_of(names_of_predictors_to_remove))
x_fin <- data.frame(removed_predfin)
sum(is.na(x_fin))
```

```
## [1] 0
```

```
final <- predict(gbm,x_fin)
write.csv(final,"/home/evm/coursework/IE2064-competition/competition-test-outcome.csv")
```