Term Project

Mary McClain, Josh Mark

```
library(tidyverse)
library(caret)
library(glmnet)
library(caTools)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(rpart)
library(Cubist)
library(gbm)
library(ipred)
library(party)
library(partykit)
library(randomForest)
library(rpart)
library(Metrics)
library(caretEnsemble)
library(doParallel)
```

```
n_cores <- detectCores()
cl <- makeCluster(n_cores - 1)
registerDoParallel(cl)
```

```
# tiger <- readxl::read_xls("C:/Users/cfitch/Desktop/OSU/Predictive Analytics/Tiger-733
2.xls", sheet = "All Data") %>% data.frame()
tiger <- readxl::read_xls("~/Desktop/data/Tiger-7332.xls",
# tiger <- readxl::read_xls("Tiger-7332.xls",
                          sheet = "All Data") %>%
  data.frame() %>%
  select(-sequence_number) # drop sequence_number column

# convert factor variables to factors
non_factor_cols <- c(17:19, 24:25)
tiger[, -non_factor_cols] <- lapply(tiger[, -non_factor_cols], factor)

training <- tiger %>%
  filter(Partition == 't') %>%
  select(-Partition)

validation <- tiger %>%
  filter(Partition == 'v') %>%
  select(-Partition)

test <- tiger %>%
  filter(Partition == 's') %>%
  select(-Partition)
```

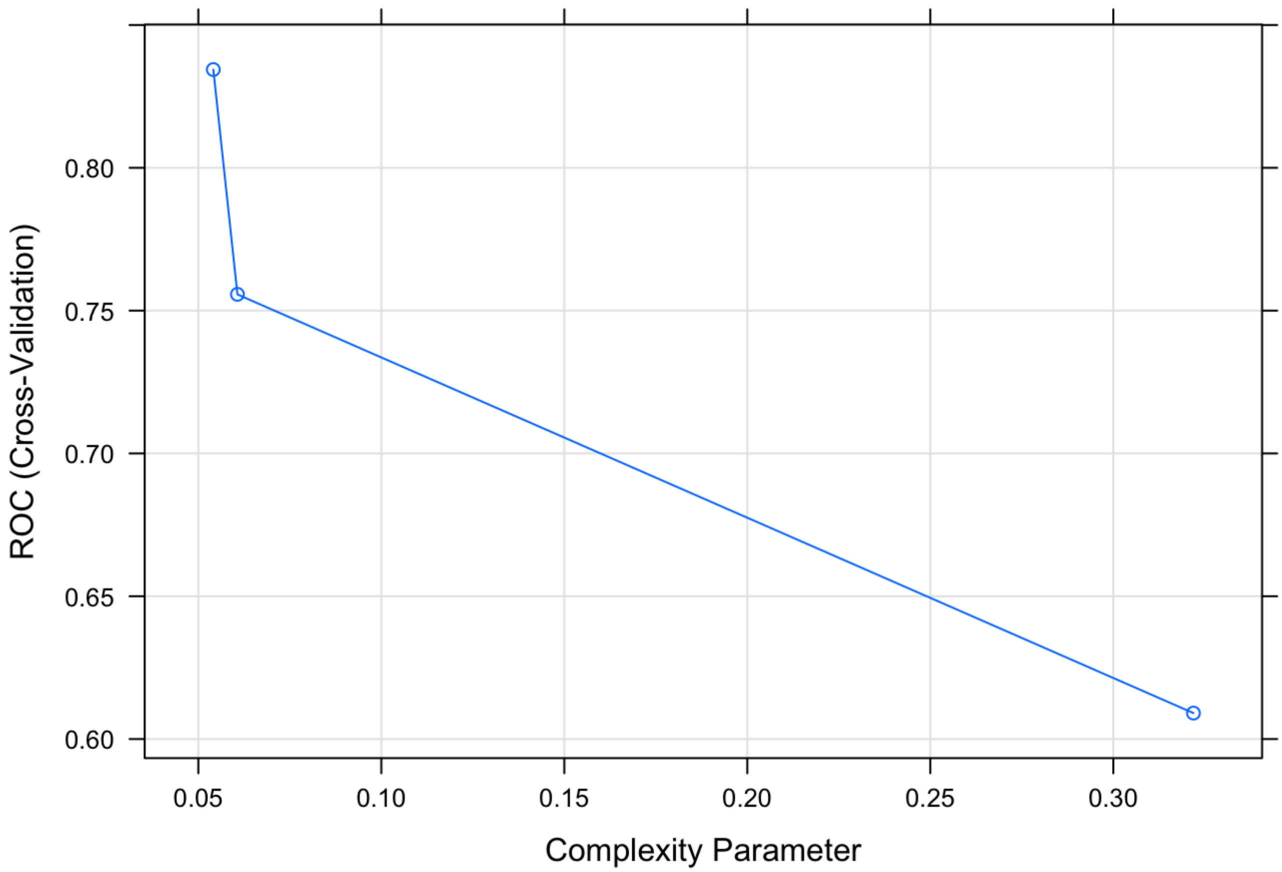
Data Prep for Classification Models

- **We removed the Partition and Spending variables from the datasets because the Partition column is only being used to tell us how to split the main Tiger dataset (Partition removed above). The Spending column we removed (below) because if we are making a prediction for a new customer, we would not know the amount they will spend. We first have to predict if they are going to purchase, then we can predict the amount they will spend, if they do purchase.**

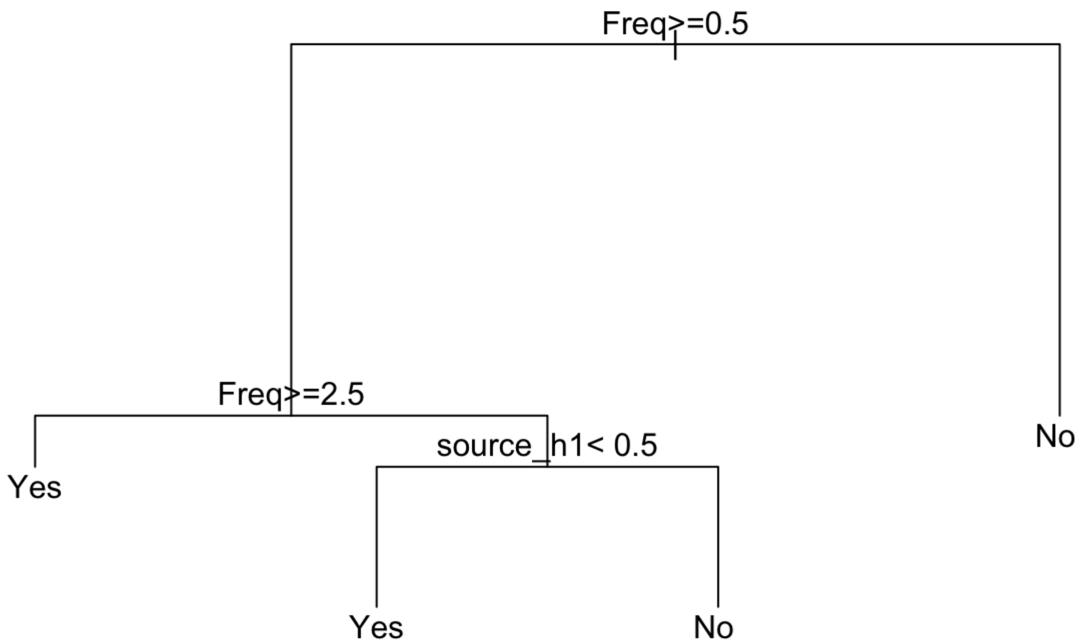
```
# Remove 'spending' variable, change levels of 'purchase' variable
training <- training %>%
  select(-Spending)
training$Purchase <- factor(ifelse(training$Purchase == 1, "Yes", "No"),
                            levels = c("Yes", "No"))

validation <- validation %>%
    select(-Spending)
validation$Purchase <- factor(ifelse(validation$Purchase == 1, "Yes", "No"),
                              levels = c("Yes", "No"))
```

Classification Tree Model

```
set.seed(1)
# classification Tree
# Fit the model
tree_model <- train(Purchase ~ .,
                    data = training,
                    method = "rpart",
                    metric = "ROC",
                    trControl = trainControl("cv",
                                             number = 5, #5-fold CV
                                             summaryFunction = twoClassSummary,
                                             classProbs = TRUE))

plot(tree_model)
```



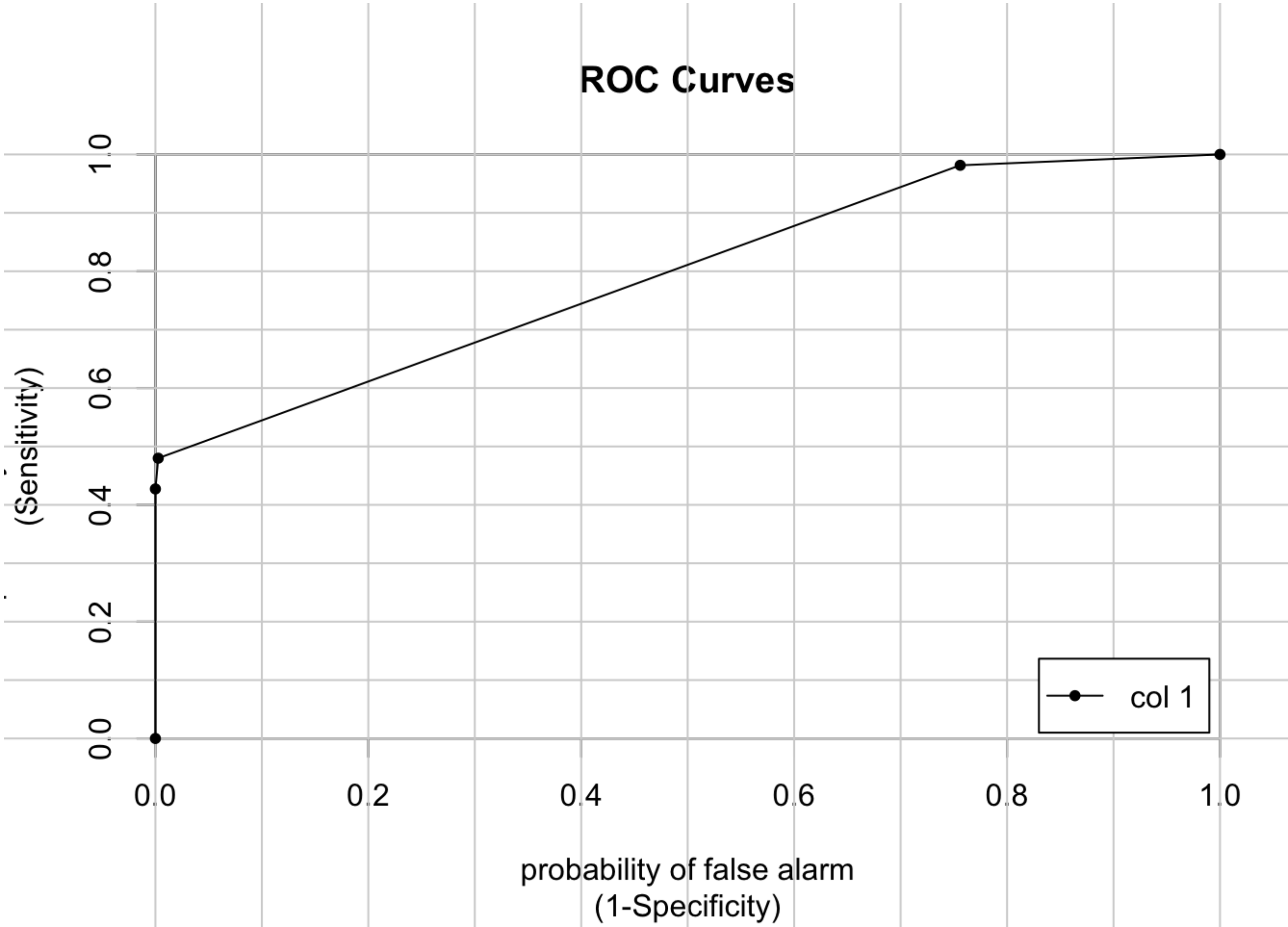

```
par(xpd = NA)
plot(tree_model$finalModel)
text(tree_model$finalModel, digits = 3)
```




```
# Make predictions on the validation set
tree_probs <- predict(tree_model, newdata = validation, type = "prob")
threshold <- 0.5
tree_preds <- factor(ifelse(tree_probs[, "Yes"] > threshold, "Yes", "No"),
                           levels = c("Yes", "No"))
confusionMatrix(tree_preds, validation$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 376 168
##        No    1 155
##
##                Accuracy : 0.7586
##                  95% CI : (0.7251, 0.7898)
##     No Information Rate : 0.5386
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4956
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9973
##             Specificity : 0.4799
##          Pos Pred Value : 0.6912
##          Neg Pred Value : 0.9936
##              Prevalence : 0.5386
##          Detection Rate : 0.5371
##    Detection Prevalence : 0.7771
##       Balanced Accuracy : 0.7386
##
##        'Positive' Class : Yes
##
```

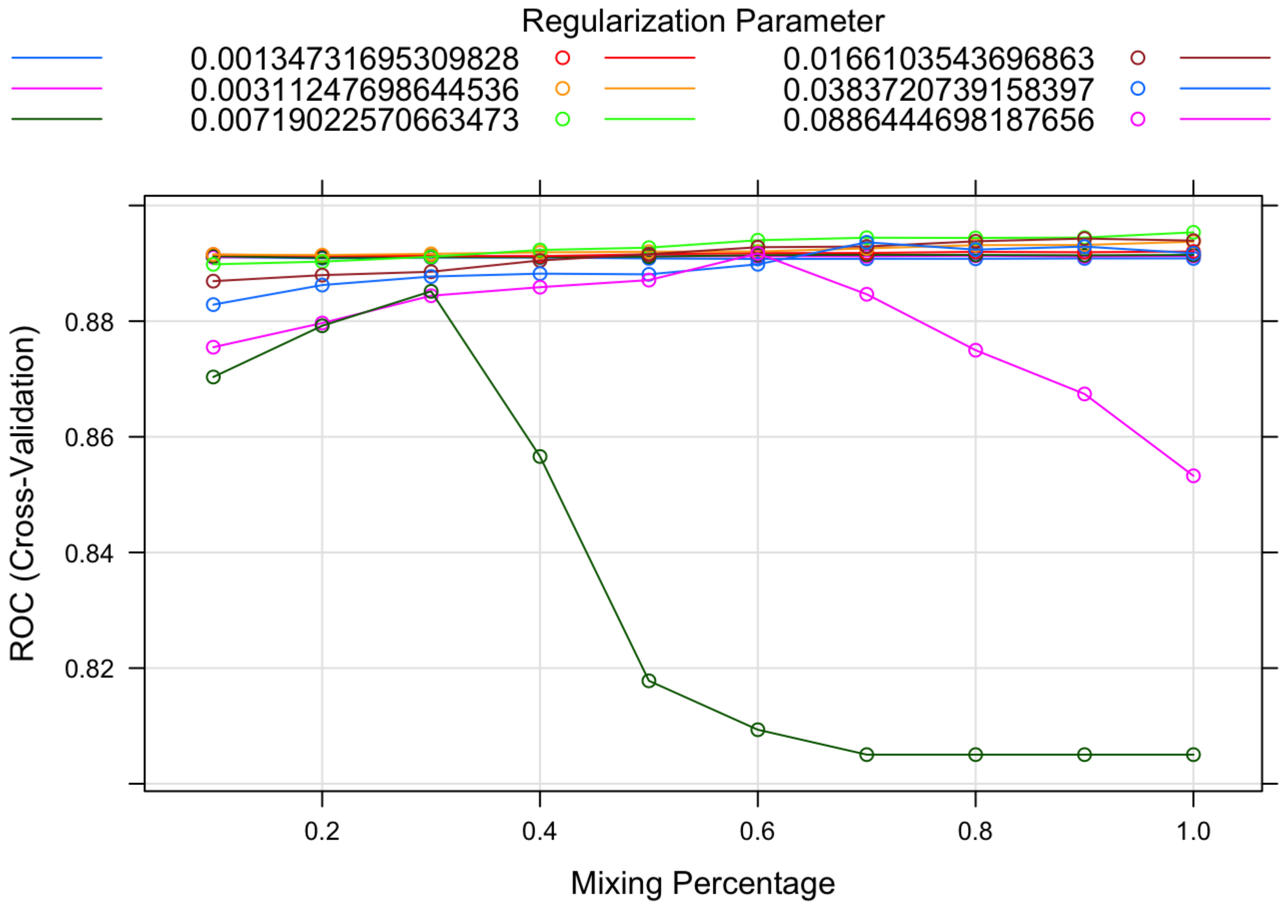
```
tree_predictions <- tree_probs[, "Yes"] # save for comparison

colAUC(tree_predictions, validation$Purchase, plotROC = TRUE)
```



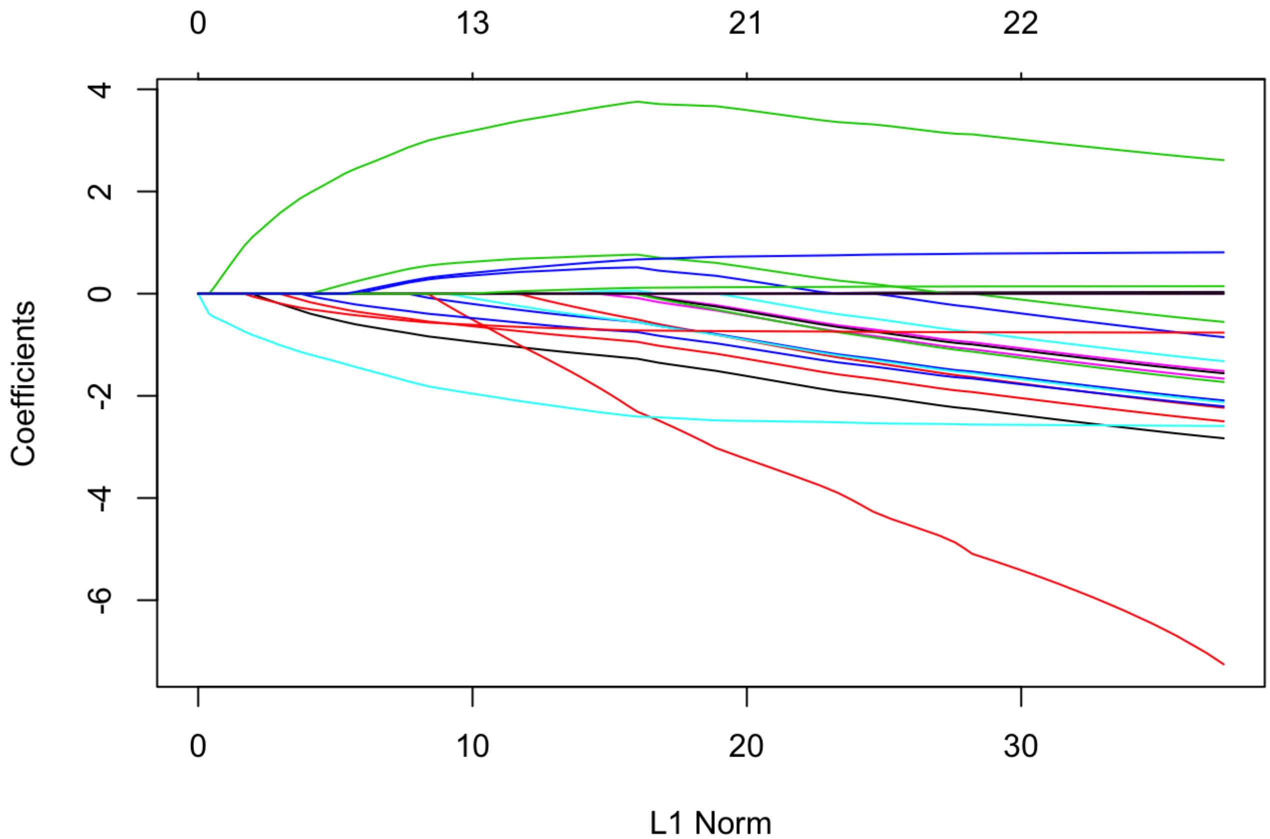

```
##                  [,1]
## Yes vs. No 0.7933786
```

Regularized Linear Model

```
set.seed(1)
# Regularized linear model using an elasicnet penalty for shrinkage
# Fit the model
enet_model <- train(Purchase ~ .,
                      data = training,
                      method = "glmnet",
                      family="binomial",
                      metric="ROC",
                      tuneLength = 10,
                      trControl = trainControl("cv", number = 5, # 5-fold CV
                                               summaryFunction=twoClassSummary,
                                               classProbs = TRUE))

plot(enet_model)
```




```
plot(enet_model$finalModel)
```



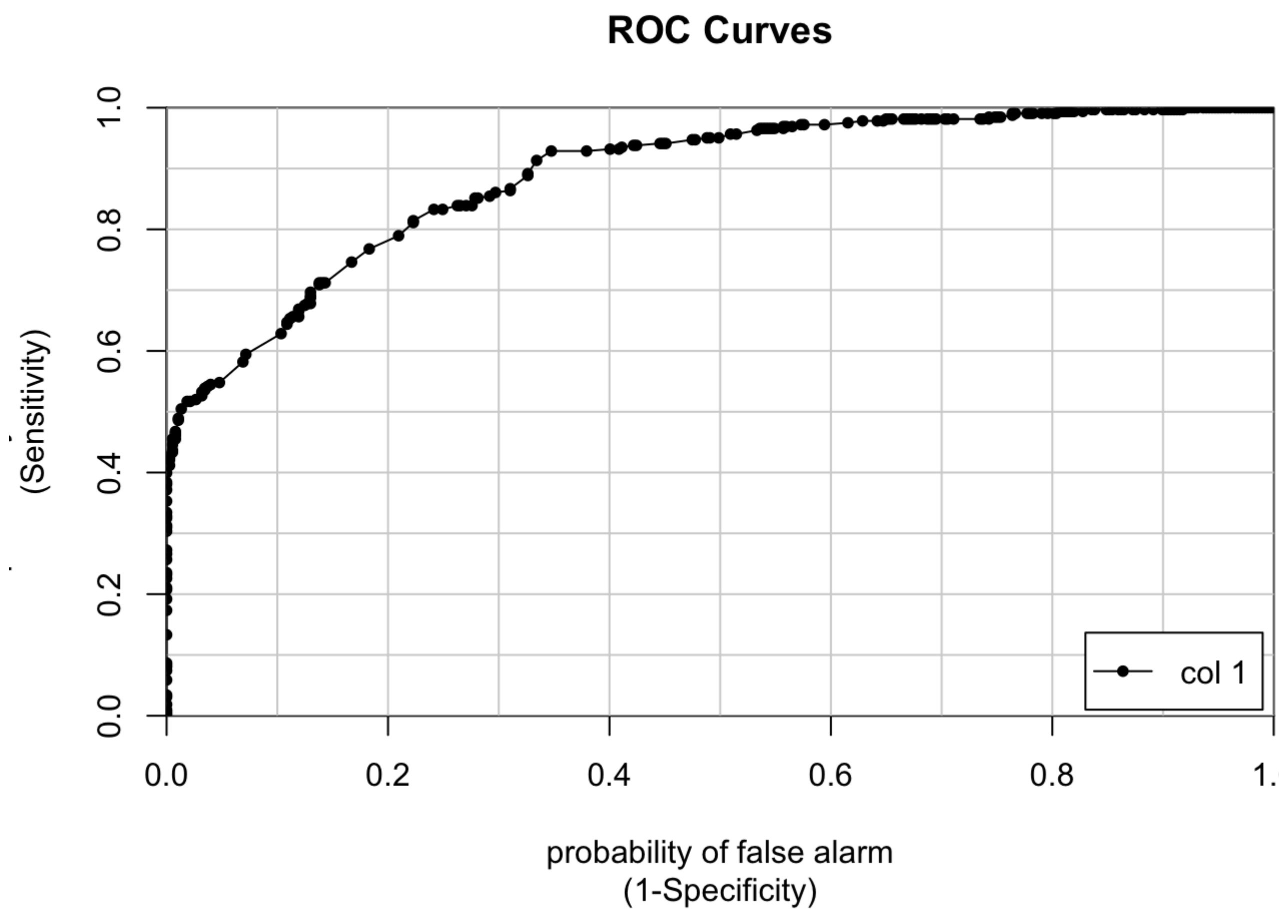

```
# alpha and lambda values for best model
tibble(alpha = enet_model$bestTune$alpha, lambda = enet_model$bestTune$lambda)
```

```
## # A tibble: 1 x 2
##   alpha  lambda
##   <dbl>   <dbl>
## 1     1 0.00719
```

```
# Make predictions on the validation set
enet_probs <- predict(enet_model, newdata = validation, type = "prob")
threshold <- 0.5
enet_preds <- factor(ifelse(enet_probs[, "Yes"] > threshold, "Yes", "No"),
                           levels = c("Yes", "No"))
confusionMatrix(enet_preds, validation$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 278  52
##        No   99 271
##
##                Accuracy : 0.7843
##                  95% CI : (0.7519, 0.8142)
##     No Information Rate : 0.5386
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5705
##  Mcnemar's Test P-Value : 0.0001815
##
##             Sensitivity : 0.7374
##             Specificity : 0.8390
##          Pos Pred Value : 0.8424
##          Neg Pred Value : 0.7324
##              Prevalence : 0.5386
##          Detection Rate : 0.3971
##    Detection Prevalence : 0.4714
##       Balanced Accuracy : 0.7882
##
##        'Positive' Class : Yes
##
```

```
enet_predictions <- enet_probs[, "Yes"] # save for comparison

colAUC(enet_predictions, validation$Purchase, plotROC = TRUE)
```

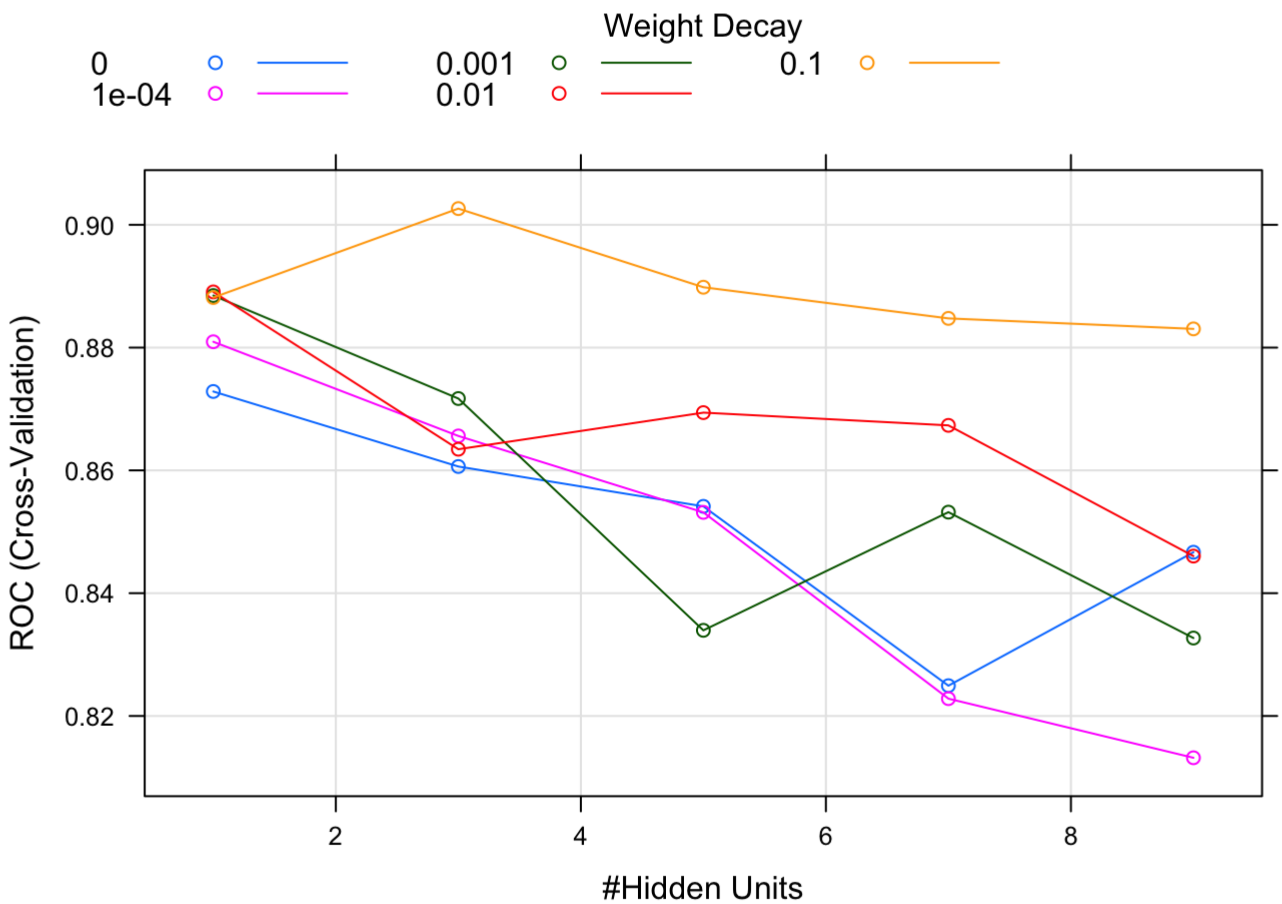



```
##                  [,1]
## Yes vs. No 0.8874363
```

Neural Network Model

```
set.seed(1)
nn_model <- train(Purchase ~ .,
                  data = training,
                  method = "nnet",
                  metric="ROC",
                  tuneLength = 5,
                  maxit=1000,
                  linout=FALSE,
                  preProcess = "range",
                  verbose = FALSE,
                  trControl = trainControl("cv", number = 5, #5-fold CV
                                           summaryFunction=twoClassSummary,
                                           classProbs = TRUE))
```

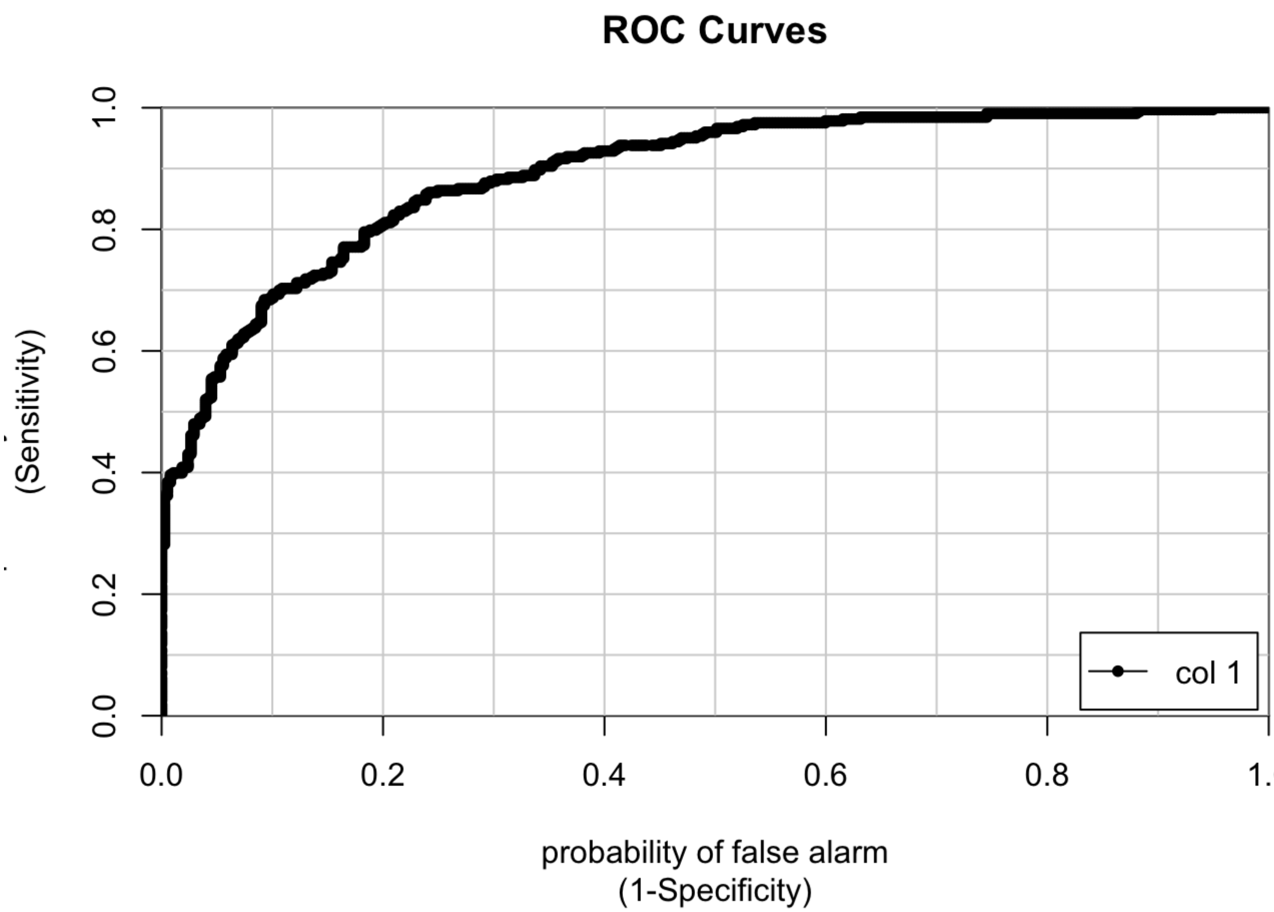
```
plot(nn_model)
```




```
# Make predictions on the validation set
nn_probs <- predict(nn_model, newdata = validation, type = "prob")
threshold <- 0.5
nn_preds <- factor(ifelse(nn_probs[, "Yes"] > threshold, "Yes", "No"),
                   levels = c("Yes", "No"))
confusionMatrix(nn_preds, validation$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 301  61
##        No   76 262
##
##                Accuracy : 0.8043
##                  95% CI : (0.7729, 0.8331)
##     No Information Rate : 0.5386
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6075
##  Mcnemar's Test P-Value : 0.2317
##
##             Sensitivity : 0.7984
##             Specificity : 0.8111
##          Pos Pred Value : 0.8315
##          Neg Pred Value : 0.7751
##              Prevalence : 0.5386
##          Detection Rate : 0.4300
##    Detection Prevalence : 0.5171
##       Balanced Accuracy : 0.8048
##
##        'Positive' Class : Yes
##
```

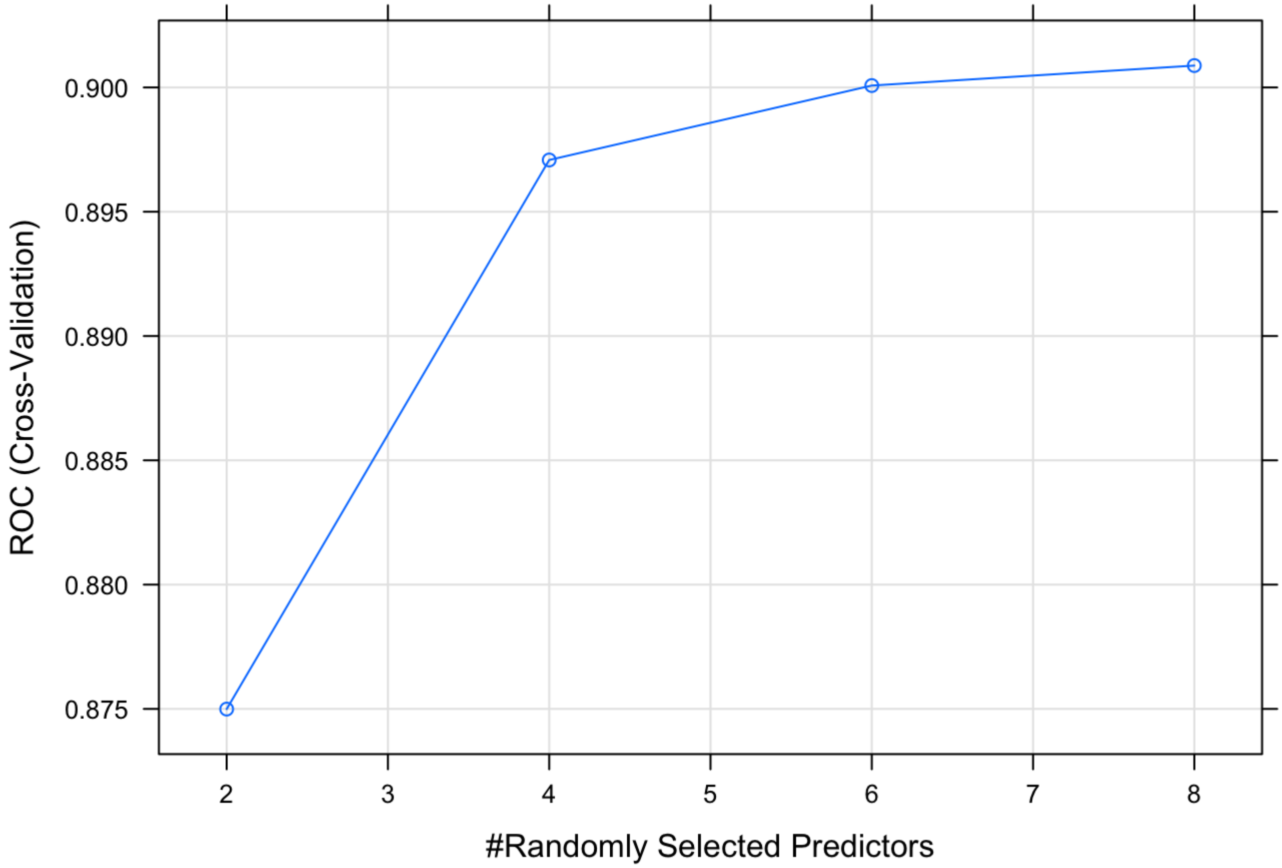
```
nnet_predictions <- nn_probs[, "Yes"] # save for comparison

colAUC(nnet_predictions, validation$Purchase, plotROC = TRUE)
```




```
##                  [,1]
## Yes vs. No 0.891025
```

Random Forest Model

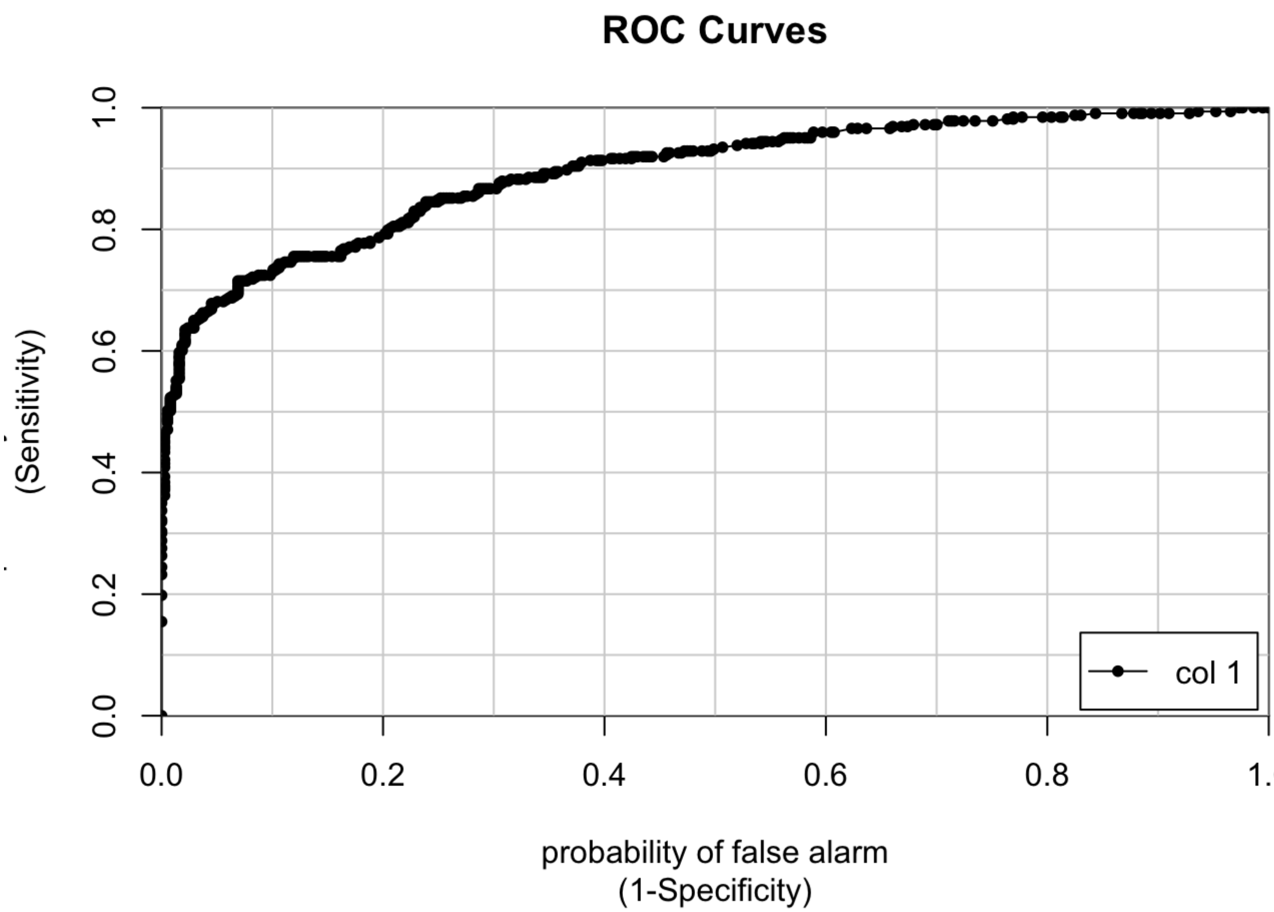
```
grid_train <- expand.grid(mtry = seq(2, 8, 2))
set.seed(1)
# Fit the model
rf_model <- train(Purchase ~ .,
                  data = training,
                  metric = 'ROC',
                  method = 'rf',
                  tuneGrid = grid_train,
                  trControl = trainControl("cv", number = 5, #5-fold CV
                                           summaryFunction=twoClassSummary,
                                           classProbs = TRUE))
# Plot the model
plot(rf_model)
```




```
# Make predictions on the validation set
rf_probs <- predict(rf_model, newdata = validation, type = "prob")
threshold <- 0.5
rf_preds <- factor(ifelse(rf_probs[, "Yes"] > threshold, "Yes", "No"),
                           levels = c("Yes", "No"))
confusionMatrix(rf_preds, validation$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 317  79
##        No   60 244
##
##                Accuracy : 0.8014
##                  95% CI : (0.7699, 0.8304)
##     No Information Rate : 0.5386
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5988
##  Mcnemar's Test P-Value : 0.1268
##
##             Sensitivity : 0.8408
##             Specificity : 0.7554
##          Pos Pred Value : 0.8005
##          Neg Pred Value : 0.8026
##              Prevalence : 0.5386
##          Detection Rate : 0.4529
##    Detection Prevalence : 0.5657
##       Balanced Accuracy : 0.7981
##
##        'Positive' Class : Yes
##
```

```
rf_predictions <- rf_probs[, "Yes"] # save for comparison

colAUC(rf_predictions, validation$Purchase, plotROC = TRUE)
```




```
##                  [,1]
## Yes vs. No 0.8944166
```

Compare the classification models

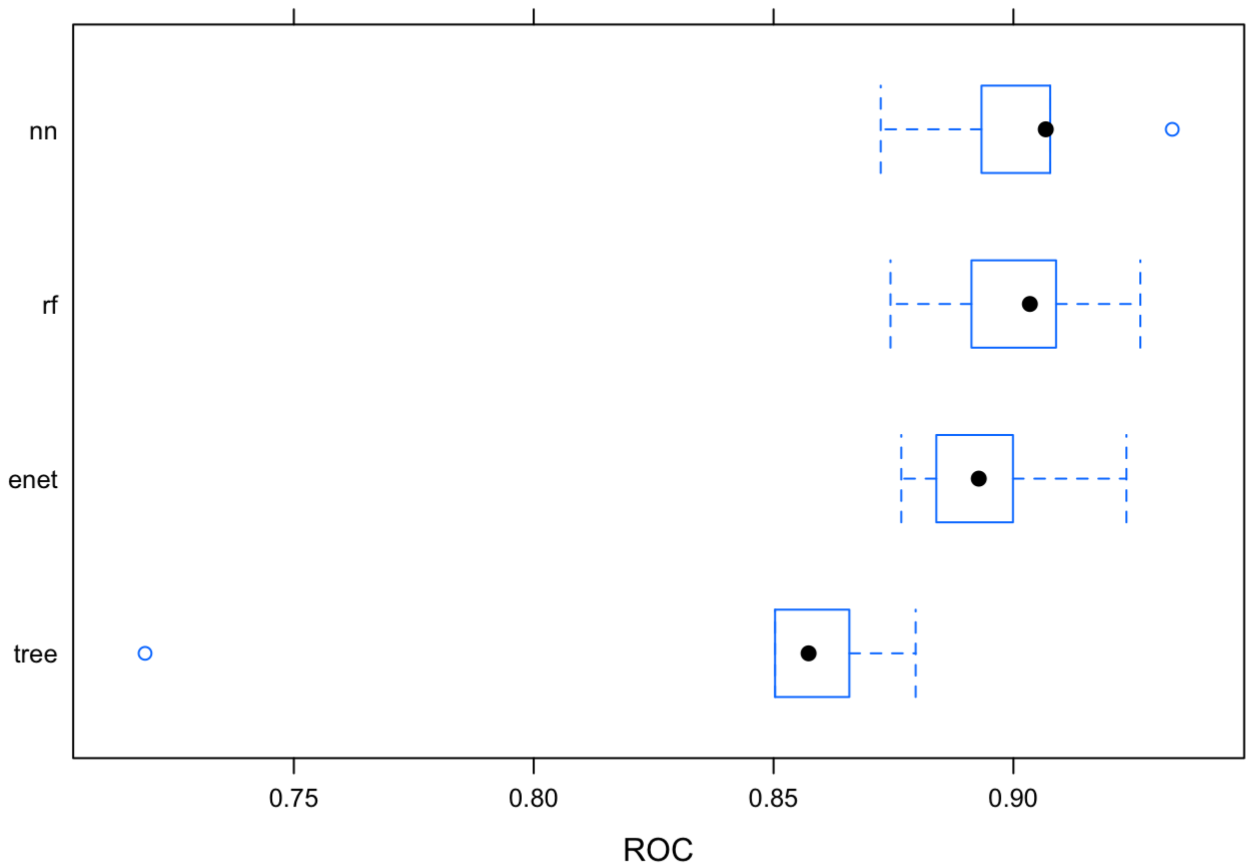
```
models <- list(tree = tree_model,
               enet = enet_model,
               nn = nn_model,
               rf = rf_model)
results <- resamples(models)
# Look at all comparison values for the models
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: tree, enet, nn, rf
## Number of resamples: 5
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## tree 0.7189850 0.8503096 0.8572995 0.8344017 0.8657937 0.8796209    0
## enet 0.8766254 0.8839286 0.8927778 0.8953593 0.8999060 0.9235589    0
## nn   0.8723371 0.8933437 0.9067460 0.9026401 0.9076598 0.9331140    0
## rf   0.8743734 0.8912539 0.9034305 0.9008807 0.9088889 0.9264568    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## tree 0.4736842 0.5921053 0.7631579 0.7125614 0.7733333 0.9605263    0
## enet 0.7105263 0.7236842 0.7333333 0.7414035 0.7368421 0.8026316    0
## nn   0.7333333 0.7763158 0.8026316 0.7914035 0.8157895 0.8289474    0
## rf   0.8026316 0.8421053 0.8421053 0.8470877 0.8552632 0.8933333    0
##
## Spec
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## tree 0.4642857 0.6941176 0.7738095 0.7554902 0.8571429 0.9880952    0
## enet 0.7647059 0.7857143 0.7857143 0.8172269 0.8571429 0.8928571    0
## nn   0.7380952 0.7976190 0.8117647 0.8052101 0.8333333 0.8452381    0
## rf   0.7142857 0.7857143 0.7857143 0.7837255 0.7976190 0.8352941    0
```

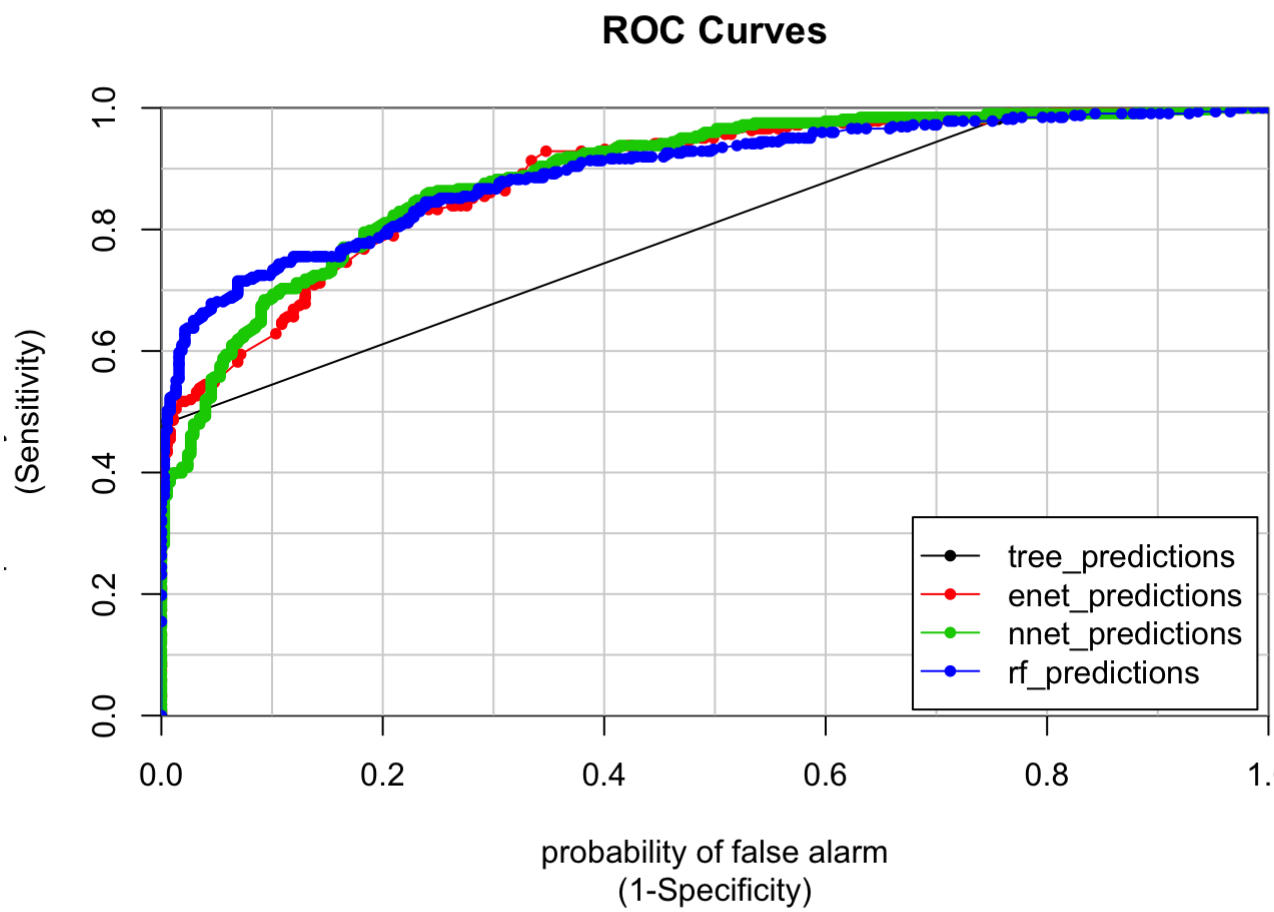
```
# Compare only area under the ROC curve for the models
summary(results, metric="ROC")
```

```
##
## Call:
## summary.resamples(object = results, metric = "ROC")
##
## Models: tree, enet, nn, rf
## Number of resamples: 5
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## tree 0.7189850 0.8503096 0.8572995 0.8344017 0.8657937 0.8796209    0
## enet 0.8766254 0.8839286 0.8927778 0.8953593 0.8999060 0.9235589    0
## nn   0.8723371 0.8933437 0.9067460 0.9026401 0.9076598 0.9331140    0
## rf   0.8743734 0.8912539 0.9034305 0.9008807 0.9088889 0.9264568    0
```

```
# Visualize AUC comparisons
bwplot(results, metric="ROC")
```




```
# ROC curves plotted together
colAUC(cbind(tree_predictions, enet_predictions, nnet_predictions, rf_predictions),
       validation$Purchase, plotROC=TRUE)
```




```
##              tree_predictions enet_predictions nnet_predictions
## Yes vs. No          0.7933786        0.8874363         0.891025
##              rf_predictions
## Yes vs. No        0.8944166
```

Compare Elastic Net and Neural Network classification models

```
compare_models(nn_model, enet_model)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 0.80796, df = 4, p-value = 0.4644
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01773866  0.03230024
## sample estimates:
##   mean of x
## 0.007280788
```

```
compare_models(rf_model, enet_model)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 0.66913, df = 4, p-value = 0.5401
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01738867  0.02843138
## sample estimates:
##   mean of x
## 0.005521352
```

Select a Classification Model

- **Based on the estimated generalization error that we see above in the ROC curves plotted together for all the classification models, we selected the Elastic Net (regularized linear model) model as the best model to use for prediction. The elastic net, neural net, and random forest models were the 3 models we were deciding between, because they all had superior AUC compared to the simple tree model, and all 3 were statistically different from the tree model. However, between those 3 candidate models, there was not a statistically difference amongst them. Following** Occam's Razor principle**, we selected the simplest of the 3 candidate models: the elastic net model.**

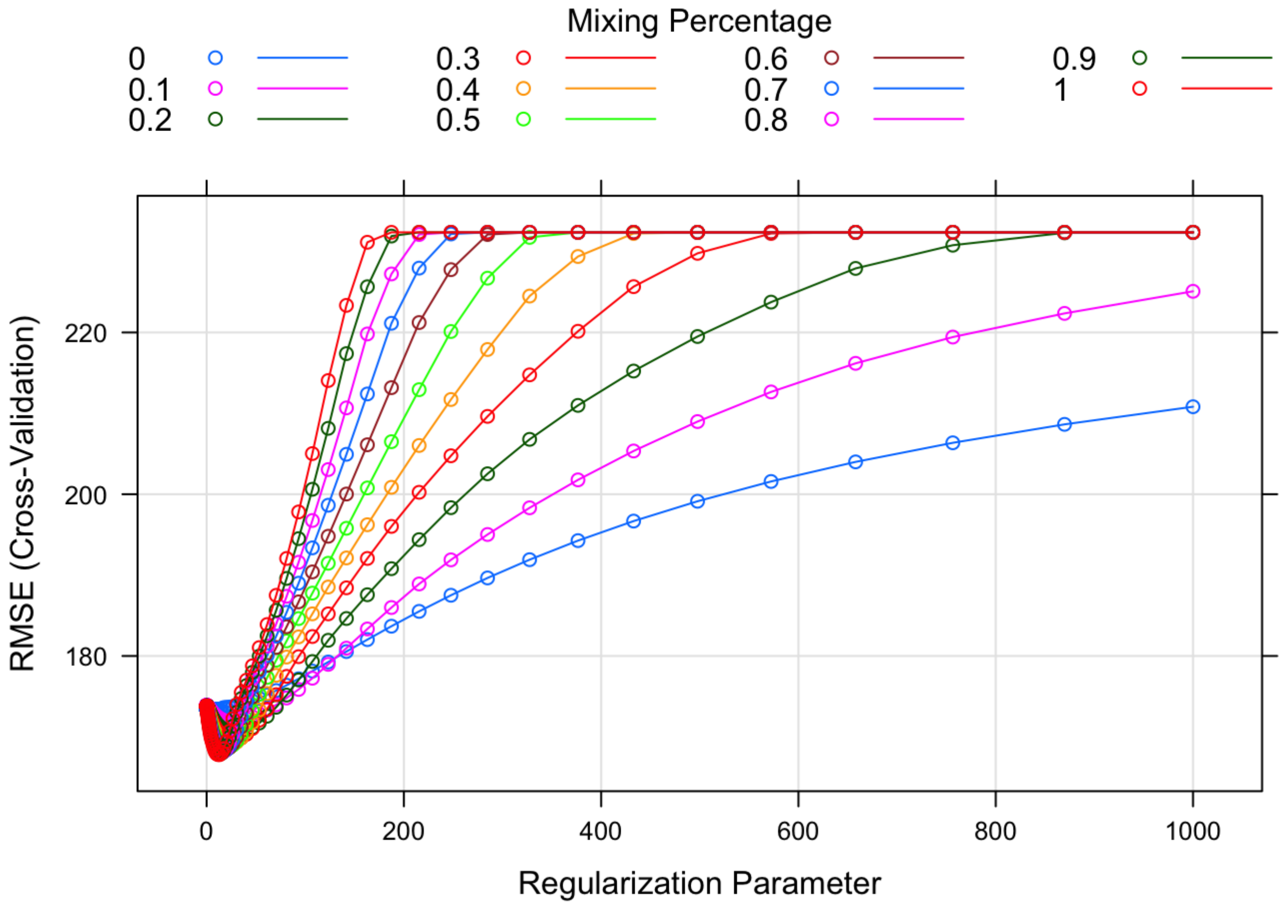
Data Prep for REGRESSION (Spending) Models

- **Filtered the training and validation datasets to only Purchasers (Purchase variable = 1) for the Spending regression model building and fine-tuning.**

```
# Remove 'spending' variable, change levels of 'purchase' variable
training <- tiger %>%
  filter(Partition == 't' & Purchase == 1) %>%  # Purchasers in training data
  select(-Partition)
training$Purchase <- factor(ifelse(training$Purchase == 1, "Yes", "No"),
                            levels = c("Yes", "No"))

validation <- tiger %>%
  filter(Partition == 'v' & Purchase == 1) %>%  # Purchasers in training data
  select(-Partition)
validation$Purchase <- factor(ifelse(validation$Purchase == 1, "Yes", "No"),
                            levels = c("Yes", "No"))
```

Regularized Multiple Linear Regression (Elastic Net)

```
set.seed(1)
lambda <- 10^seq(-3, 3, length = 100)
spend_enet_model <- train(Spending ~ .,
                         data = training,
                         method = 'glmnet',
                         trControl = trainControl("cv", number = 5), #5-fold CV
                        tuneGrid = expand.grid(alpha = seq(0, 1, by=0.1),
                                               lambda = lambda))
plot(spend_enet_model)
```




```
# Coefficients for the best model
coef(spend_enet_model$finalModel, spend_enet_model$bestTune$lambda)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           58.33112253
## US1                    3.57235607
## source_a1              .
## source_c1              .
## source_b1              .
## source_d1              .
## source_e1              .
## source_m1              .
## source_o1              .
## source_h1              .
## source_r1              .
## source_s1              .
## source_t1              .
## source_u1              .
## source_p1              .
## source_x1              .
## source_w1              .
## Freq                  86.15942987
## last_update_days_ago  -0.01398325
## X1st_update_days_ago   .
## Web.order1             .
## Gender.male1           .
## Address_is_res1      -54.59338818
## PurchaseNo             .
```

```
# best alpha and lambda
tibble(alpha = spend_enet_model$bestTune$alpha,
       lambda = spend_enet_model$bestTune$lambda)
```

```
## # A tibble: 1 x 2
##   alpha lambda
##   <dbl>  <dbl>
## 1     1   13.2
```

```
# make predictions on the validation data
spend_enet_preds <- predict(spend_enet_model, newdata = validation)

# RMSE and R2 values for elasticnet model
data.frame(
  RMSE = RMSE(spend_enet_preds, validation$Spending),
  Rsquare = R2(spend_enet_preds, validation$Spending))
```

```
##       RMSE   Rsquare
## 1 163.1215 0.4573211
```

Regression Tree

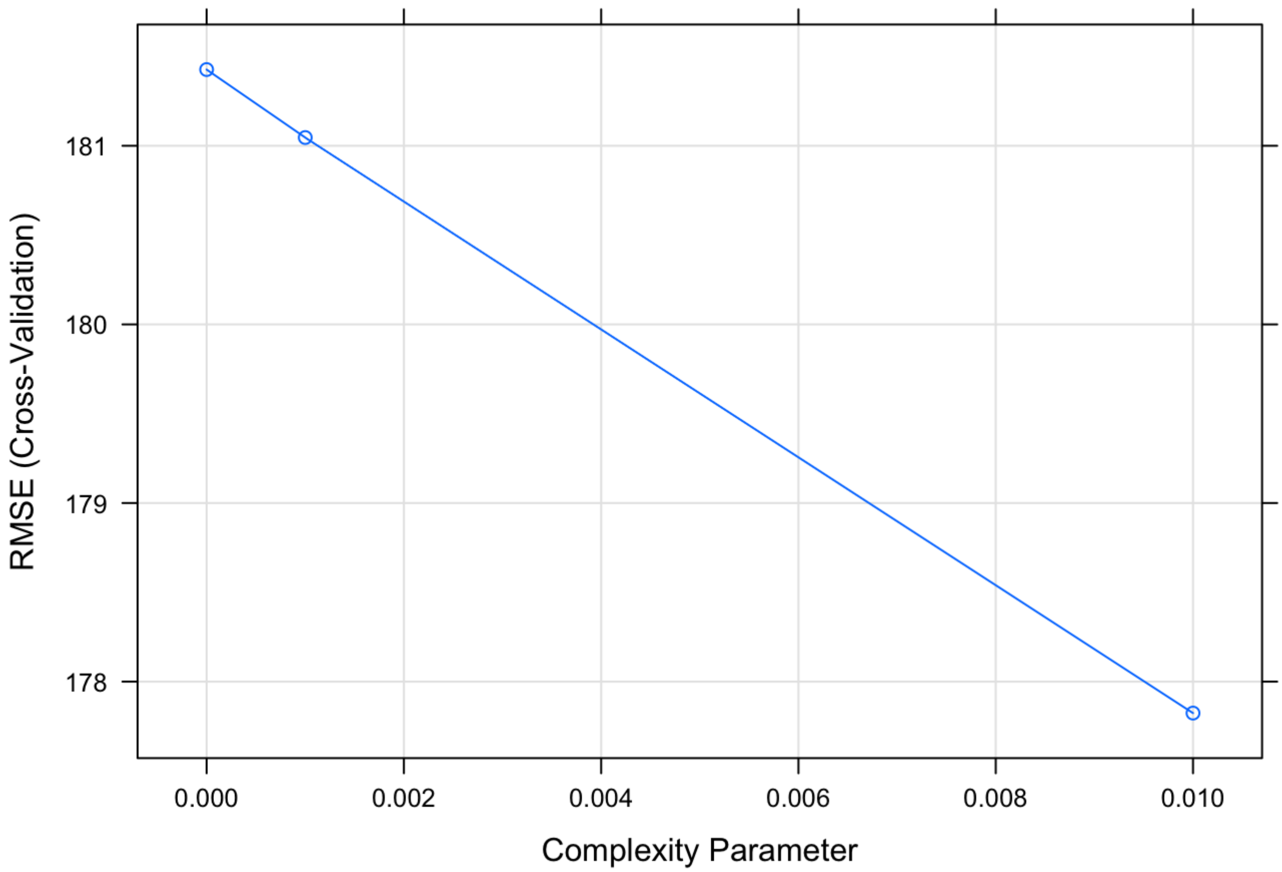
```
set.seed(1)
spend_tree_model <- train(Spending ~ .,
                          data = training,
                          method = "rpart",
                          trControl = trainControl("cv", number = 5), #5-fold CV
                          tuneGrid = expand.grid(cp=c(0, 0.001, 0.01)))

# complexity parameter for best model
spend_tree_model$bestTune$cp
```

```
## [1] 0.01
```

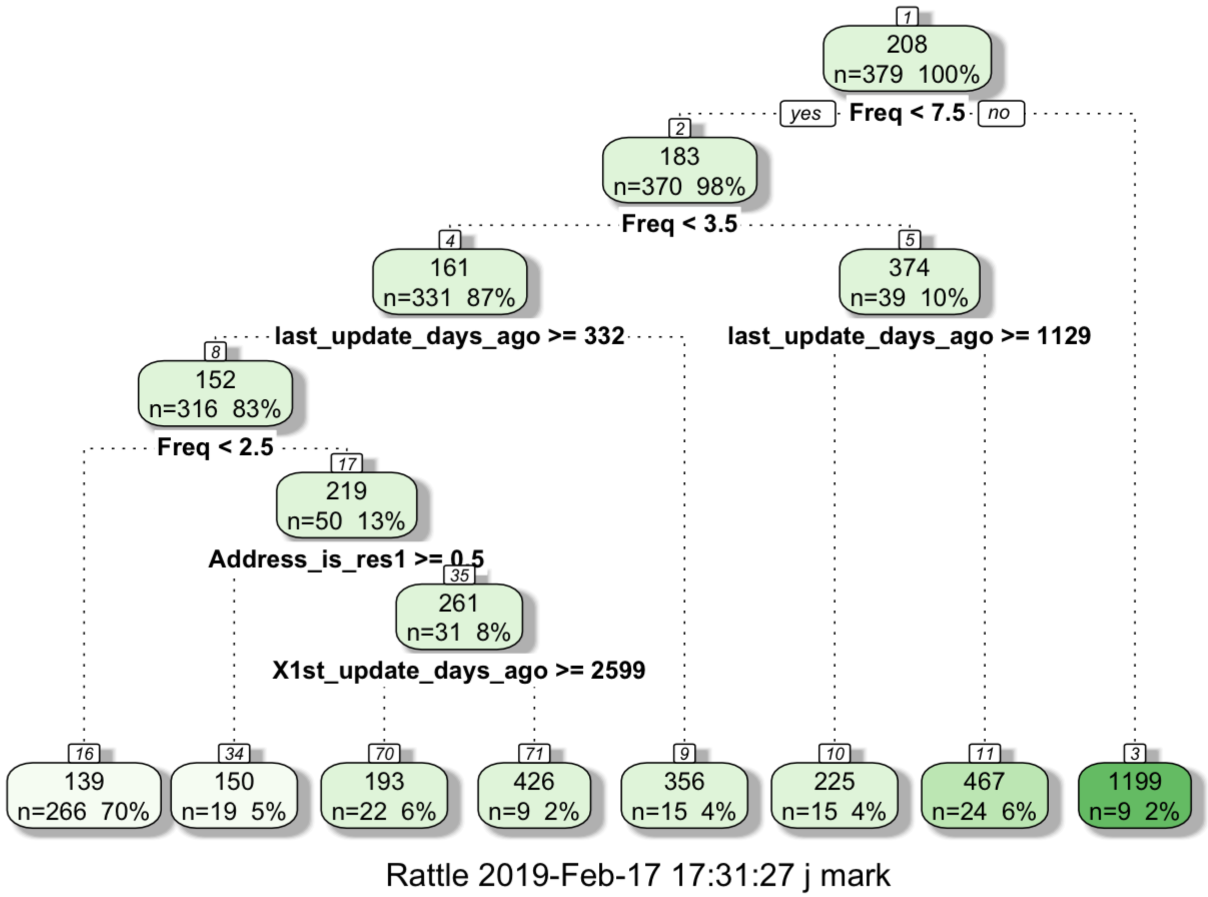
```
# make predictions on the validation set using the regression tree model
spend_tree_preds <- predict(spend_tree_model, newdata = validation)

# RMSE and R2 values for regression tree
data.frame(RMSE = RMSE(spend_tree_preds, validation$Spending),
           R2 = R2(spend_tree_preds, validation$Spending))
```

```
##       RMSE        R2
## 1 185.7709 0.3084411
```

```
# plot the regression tree model
plot(spend_tree_model)
```

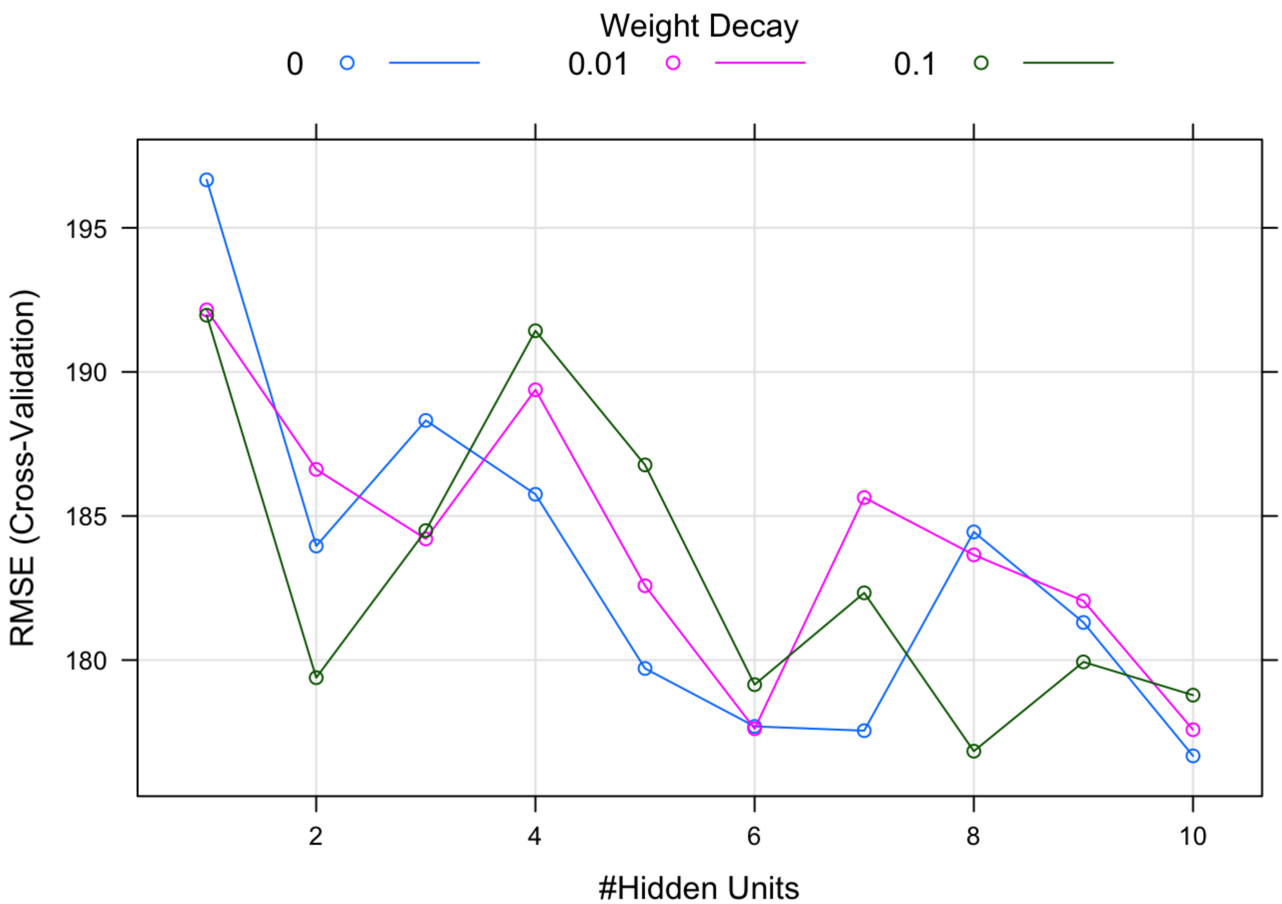



```
fancyRpartPlot(spend_tree_model$finalModel)
```



Neural Net for Spending (Regression)

```
set.seed(1)
nnetGrid <- expand.grid(.decay = c(0, 0.01, 0.1),
                        .size = c(1:10),
                        .bag = FALSE)
spend_nnet_model <- train(Spending ~ .,
                          data = training,
                          method = "avNNet",
                          trControl = trainControl("cv", number = 5),
                          tuneGrid = nnetGrid,
                          preProc = c("center", "scale"),
                          linout = TRUE,
                          trace = FALSE,
                          maxNWts = 10 * (ncol(training) + 1) + 10 + 1,
                          maxit = 500)
plot(spend_nnet_model)
```




```
# Parameters of best model
data.frame(size = spend_nnet_model$bestTune$size,
           decay = spend_nnet_model$bestTune$decay,
           bag = spend_nnet_model$bestTune$bag)
```

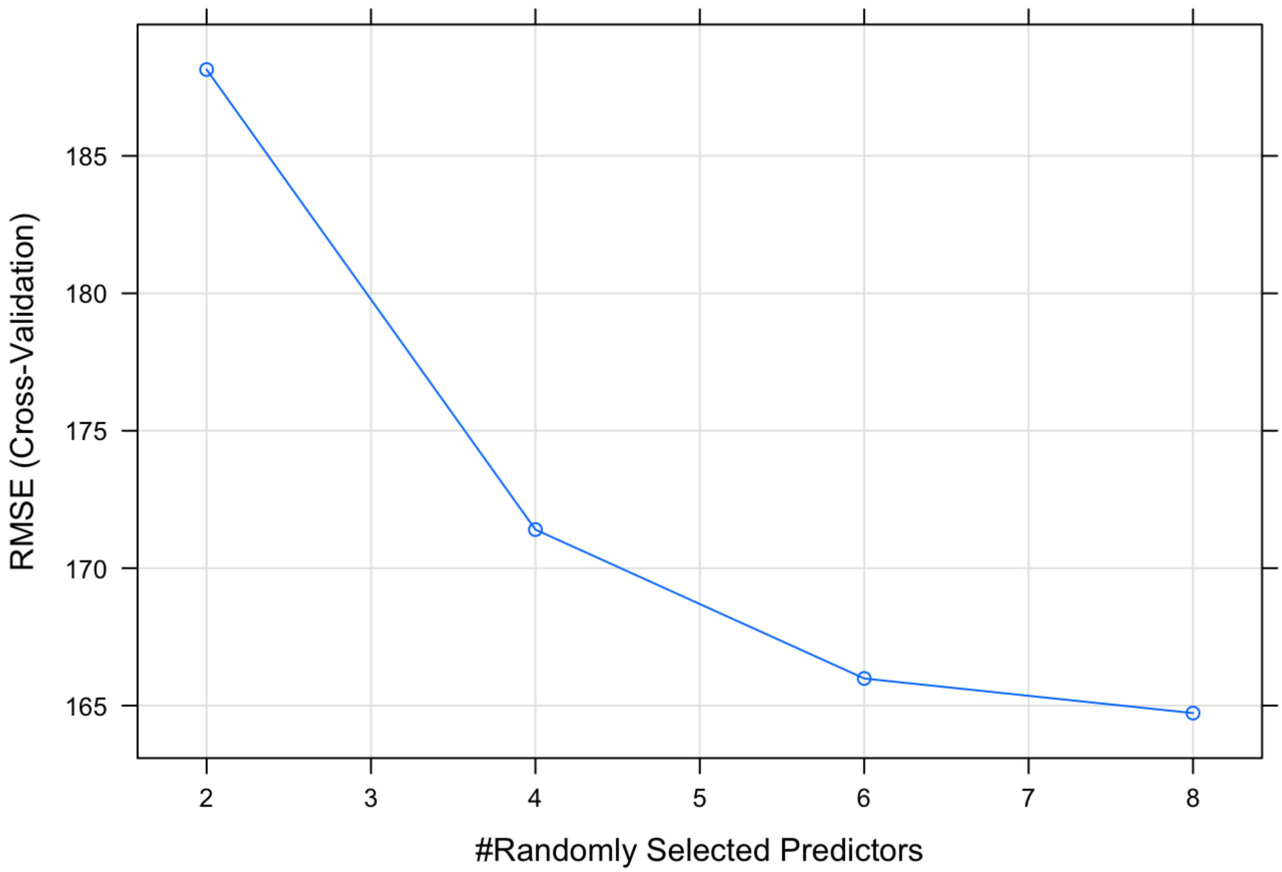
```
##   size decay   bag
## 1   10     0 FALSE
```

```
# make predictions on the validation set using the regression tree model
spend_nnet_preds <- predict(spend_nnet_model, newdata = validation)

# RMSE and R2 values
data.frame(RMSE = RMSE(spend_nnet_preds, validation$Spending),
           R2 = R2(spend_nnet_preds, validation$Spending))
```

```
##      RMSE        R2
## 1 180.629 0.3688446
```

Random Forest Model

```
grid_train <- expand.grid(mtry = seq(2, 8, 2))
set.seed(1)
spend_rf_model <- train(Spending ~ .,
                        data = training,
                        method = 'rf',
                        tuneGrid = grid_train,
                        trControl = trainControl("cv", number = 5))
plot(spend_rf_model)
```




```
# Look at the results of different values for mtry
spend_rf_model$results
```

```
##   mtry     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1    2 188.1414 0.4485064 116.3362 35.59182  0.2625364  9.36101
## 2    4 171.4037 0.4617129 106.5865 41.80119  0.2523629 10.58953
## 3    6 165.9868 0.4710122 103.0619 44.38146  0.2430195 11.46920
## 4    8 164.7273 0.4707067 102.5076 45.80257  0.2420020 11.51231
```

```
# best tune
spend_rf_model$bestTune
```

```
##   mtry
## 4    8
```

```
# make predictions
spend_rf_preds <- predict(spend_rf_model, newdata = validation)

# RMSE and R2 values
data.frame(RMSE = RMSE(spend_rf_preds, validation$Spending),
           R2 = R2(spend_rf_preds, validation$Spending))
```

```
##       RMSE       R2
## 1 171.1085 0.398986
```

Regression Model Comparisons

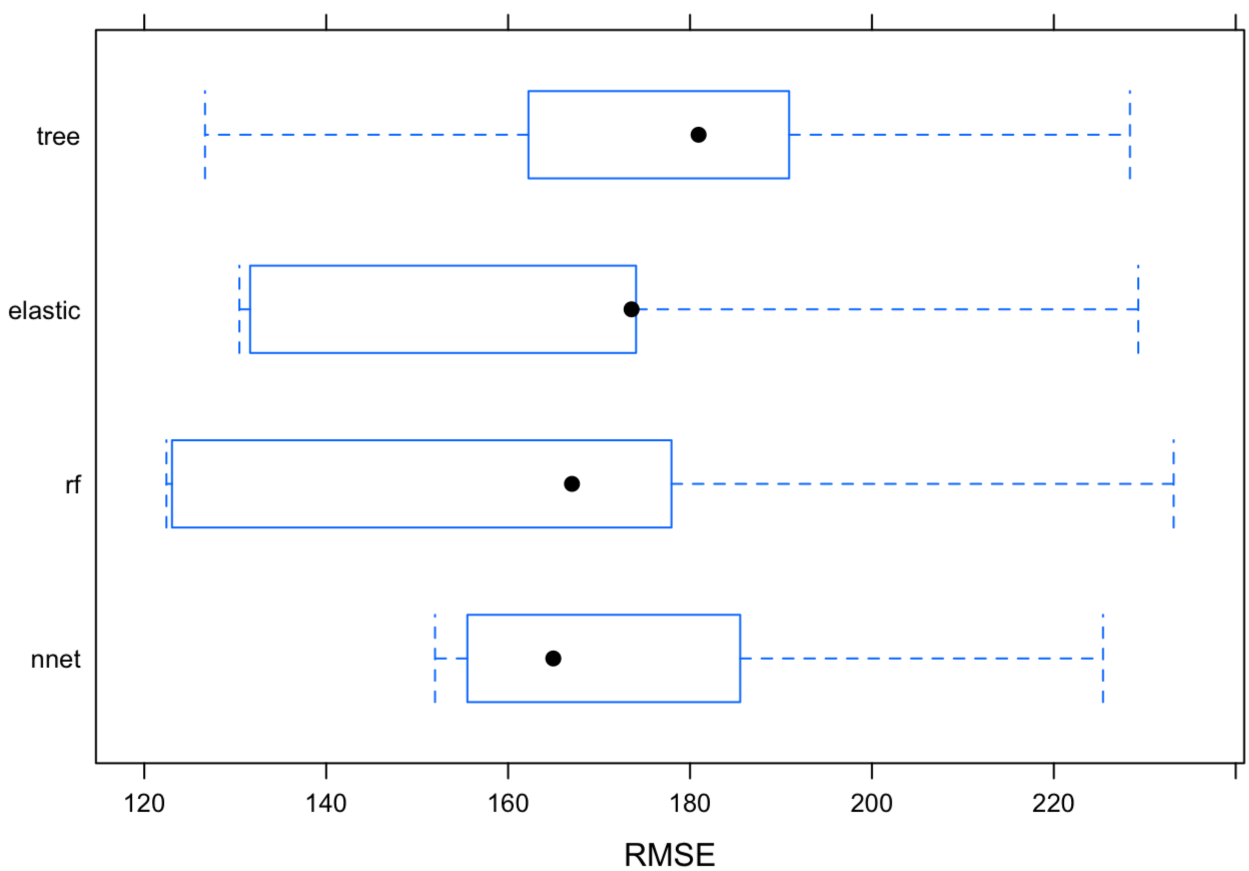
```
models <- list(elastic = spend_enet_model,
               nnet = spend_nnet_model,
               tree = spend_tree_model,
               rf = spend_rf_model)
results <- resamples(models)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: elastic, nnet, tree, rf
## Number of resamples: 5
##
## MAE
##              Min.   1st Qu.    Median     Mean  3rd Qu.     Max. NA's
## elastic  98.01112 100.69394 106.56777 109.6646 121.2982 121.7519    0
## nnet    109.93870 114.29684 115.93898 115.5867 116.4282 121.3310    0
## tree     86.63424 108.22142 110.98407 107.5187 111.6957 120.0581    0
## rf       89.41542  96.98464  97.59627 102.5076 110.5722 117.9694    0
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic 130.4571 131.6331 173.5687 167.8017 174.0669 229.2828    0
## nnet    151.9645 155.5194 164.9673 176.6754 185.5164 225.4092    0
## tree    126.6774 162.2352 180.9407 177.8239 190.8967 228.3697    0
## rf      122.4352 123.0457 167.0200 164.7273 177.9603 233.1755    0
##
## Rsquared
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## elastic 0.14279151 0.3986987 0.4114463 0.4683601 0.5732576 0.8156064    0
## nnet    0.12142240 0.3933719 0.4291756 0.4187840 0.4791543 0.6707956    0
## tree    0.09166286 0.3579207 0.3850518 0.4279889 0.6465014 0.6588079    0
## rf      0.18132031 0.3664838 0.3859760 0.4707067 0.6148265 0.8049270    0
```

```
summary(results, metric = "RMSE")
```

```
##
## Call:
## summary.resamples(object = results, metric = "RMSE")
##
## Models: elastic, nnet, tree, rf
## Number of resamples: 5
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic 130.4571 131.6331 173.5687 167.8017 174.0669 229.2828    0
## nnet    151.9645 155.5194 164.9673 176.6754 185.5164 225.4092    0
## tree    126.6774 162.2352 180.9407 177.8239 190.8967 228.3697    0
## rf      122.4352 123.0457 167.0200 164.7273 177.9603 233.1755    0
```

```
bwplot(results, metric = "RMSE")
```



```
regression_results <- data.frame (
  model = c("elastic", "neural_net", "tree", "rf"),
  validation_rmse = c(RMSE(predict(spend_enet_model, validation),
                  validation$Spending),
               RMSE(predict(spend_nnet_model, validation),
                  validation$Spending),
               RMSE(predict(spend_tree_model, validation),
                  validation$Spending),
               RMSE(predict(spend_rf_model, validation),
                  validation$Spending)))
regression_results
```

```
##        model validation_rmse
## 1    elastic        163.1215
## 2 neural_net        180.6290
## 3       tree        185.7709
## 4         rf        171.1085
```

```
compare_models(spend_tree_model, spend_enet_model)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 1.5105, df = 4, p-value = 0.2054
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -8.400151 28.444578
## sample estimates:
## mean of x
##  10.02221
```

```
compare_models(spend_rf_model, spend_nnet_model)
```

```
##
##  One Sample t-test
##
## data:  x
## t = -1.2651, df = 4, p-value = 0.2745
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -38.17036  14.27430
## sample estimates:
## mean of x
## -11.94803
```

Select a Regression Model

- **When we compare the RMSE values of our regression models, we see that the random forest model and elasticnet model seemingly outperform the other two models. The random forest model has the lowest mean RMSE and the elasticnet has the lowest median RMSE while the tree model and neural network model both have higher RMSEs. When we look for a statistical difference between models however, we see that there is none. In order to remain consistent with our logic from the selection of our classification model, we again rely on** Occam's Razor principle **in our decision and select the least complex model for ease of interpretability. Thus, our selected regression model is again the regularized multiple linear regression model, the elasticnet.**

Create score_analysis

```
# create copy of test dataset created
score_analysis <- test
score_analysis$Purchase <- factor(ifelse(score_analysis$Purchase == 1, "Yes", "No"),
                                  levels = c("Yes", "No"))
```

A. **Score (or copy the scores, the "predictedprobabilityof success"(success = purchase) that were generated in part 1 to this sheet) using the chosen classification model from part 1.**

```
# make classification predictions
purchase_predictions <- predict(enet_model, newdata = score_analysis,
                                type = "prob")

# add predicted purchase probability on to score_analysis
score_analysis <- cbind(score_analysis,
                        pred_purchase_prob = purchase_predictions[, "Yes"])
```

B. **Score the cases/observations in this dataset using the chosen prediction model for spending (from part 2 above).**

```
# make spending (regression) predictions
spend_predictions <- predict(spend_enet_model, newdata = score_analysis)

# add predicted spending to score_analysis
score_analysis <- cbind(score_analysis, pred_spending = spend_predictions)
```

C. **Arrange the following columns so they are adjacent:**

- Predicted probability of purchase (Success)
- Predicted spending (dollars)
- Actual spending (dollars)

```
score_analysis <- score_analysis %>%
  select(pred_purchase_prob, pred_spending, Spending, Purchase, everything())
```

D. **Add a column for "adjusted probability of purchase" by multiplying "predicted probability of purchase" by 0.107. (This is to adjust for oversampling the purchasers noted above).**

E. **Add a column for expected spending (adjusted probability of purchase X predicted spending).**

F. **Sort all records on the "expected spending" column.**

```
score_analysis <- score_analysis %>%
  mutate(adj_purchase_prob = pred_purchase_prob * 0.107) %>%  # STEP D
  mutate(expected_spending = adj_purchase_prob * pred_spending) %>% # STEP E
  select(pred_purchase_prob, pred_spending, Spending, Purchase,
         adj_purchase_prob, expected_spending, everything()) %>% # re-arrange columns
  arrange(-expected_spending) # STEP F
```

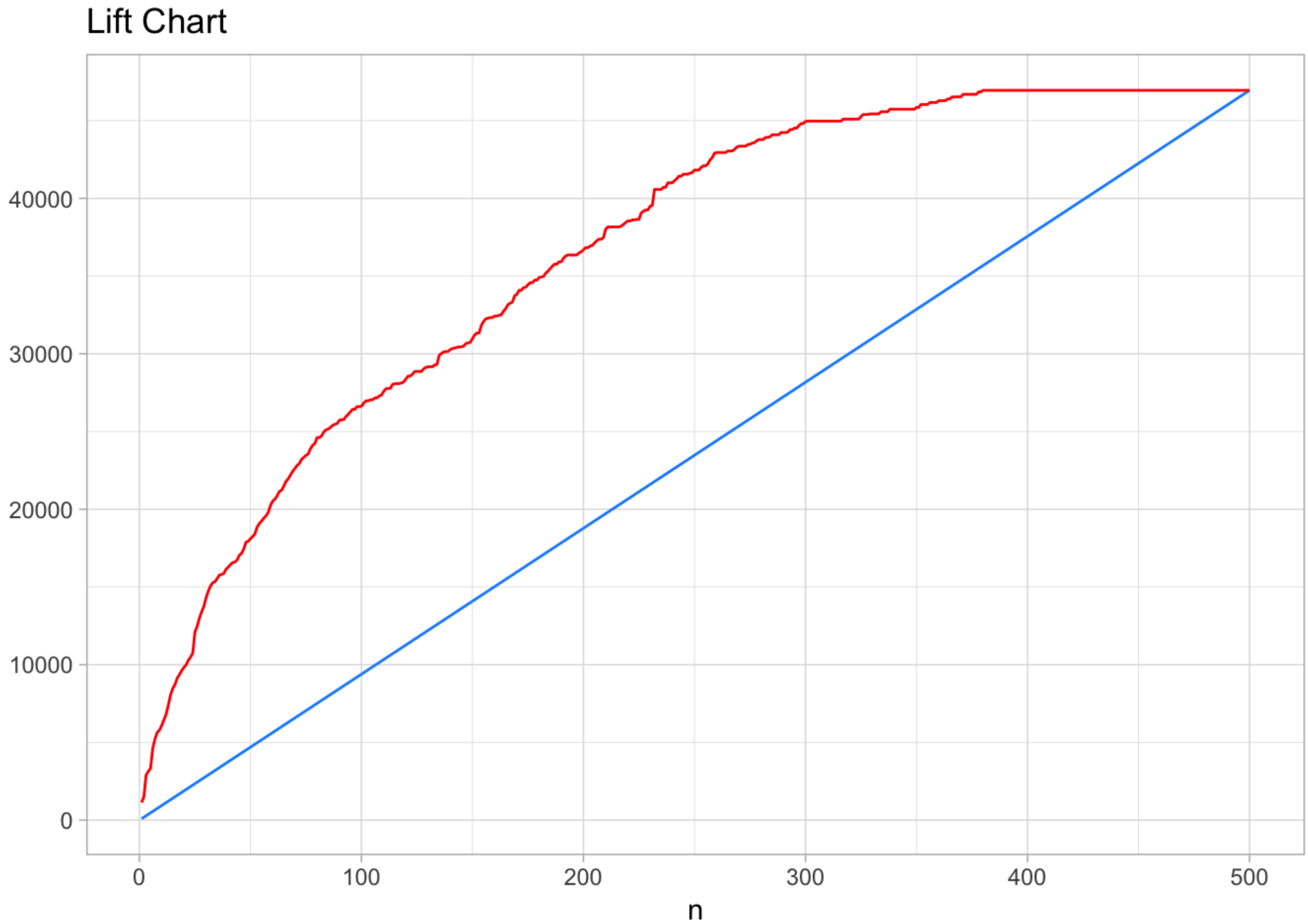
G. **Calculate cumulative lift (cumulative "actual spending" divided by the cumulative average spending that would result from random selection) - note that total spending in the test data partition was $46951 from 500 customers.**

```
score_analysis <- score_analysis %>%
  mutate(cum_actual_spend = cumsum(Spending),
         cum_avg_rand_spend = row_number() * (46951 / 500),
         cum_lift = cum_actual_spend / cum_avg_rand_spend)
sum(score_analysis$cum_lift)
```

```
## [1] 1101.812
```

H. **Plot the lift chart for your targeting model.**

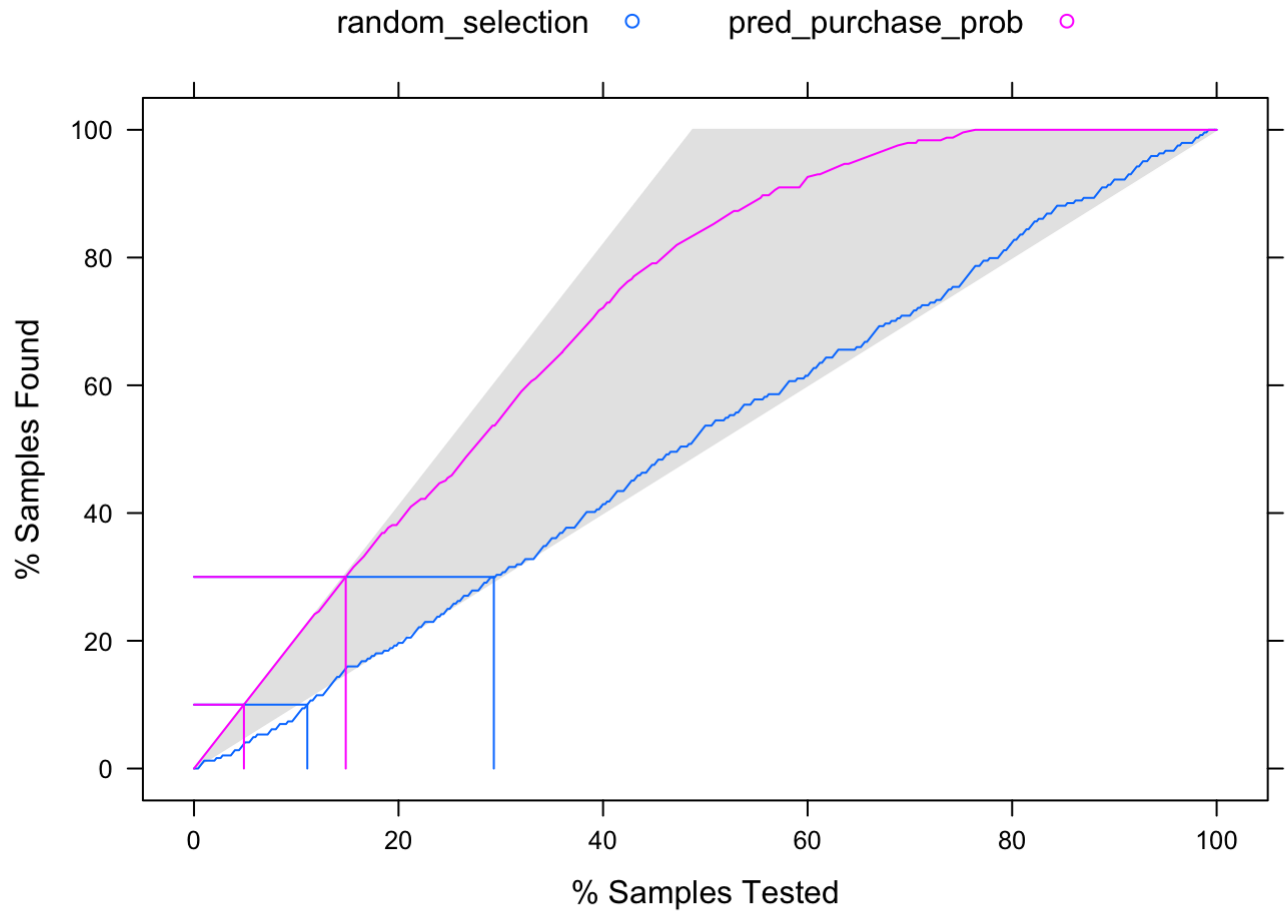
```
score_analysis %>%
  mutate(n = row_number()) %>%
  ggplot() +
  geom_line(aes(n, cum_avg_rand_spend), col = 'dodgerblue') +
  geom_line(aes(n, cum_actual_spend), col = 'red') +
  labs(title = 'Lift Chart', y = NULL) +
  theme_light()
```



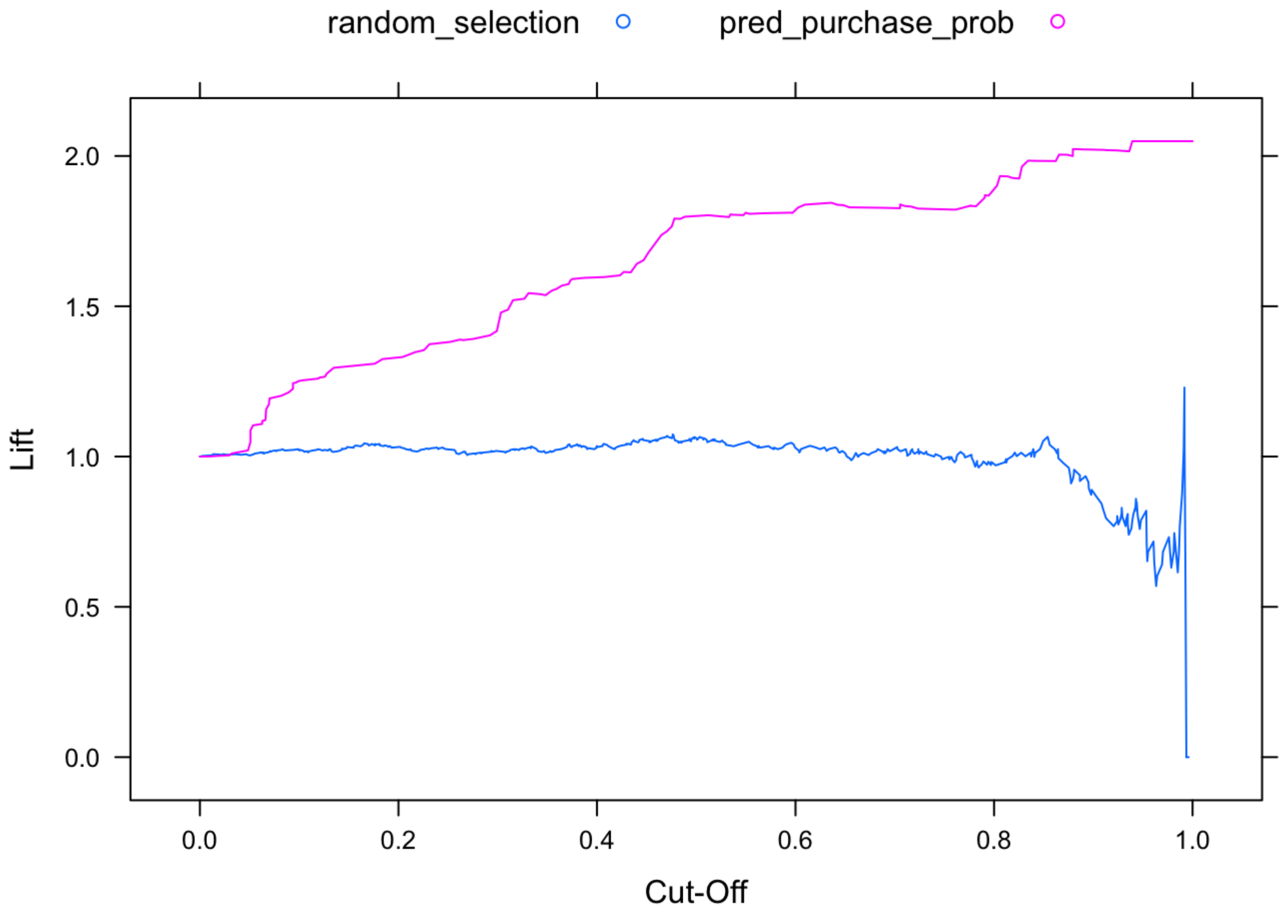

```
set.seed(1)
random_selection <- runif(500)
lift_calc <- lift(Purchase ~ random_selection + pred_purchase_prob,
              data = score_analysis)
lift_calc
```

```
##
## Call:
## lift.formula(x = Purchase ~ random_selection + pred_purchase_prob, data
##  = score_analysis)
##
## Models: random_selection, pred_purchase_prob
## Event: Yes (48.8%)
```

```
xyplot(lift_calc, auto.key = list(columns = 2), value = c(10, 30))
```




```
xyplot(lift_calc, plot = "lift", auto.key = list(columns = 2))
```



4. Each Catalog costs approximately 2 dollars to mail (including printing, postage and mailing costs). Estimate the gross profit that the firm could expect from the remaining 180,000 names (prospective customers) if it selected them randomly from the pool.

```
mailed_to_people <- 180000 * .5
mailed_to_people
```

```
## [1] 90000
```

```
avg_exp_spending <- (46951 / 500)
avg_exp_spending
```

```
## [1] 93.902
```

```
mailed_to_purchasers <- mailed_to_people * .053
mailed_to_purchasers
```

```
## [1] 4770
```

```
mailed_to_exp_spending <- mailed_to_purchasers * avg_exp_spending
mailed_to_exp_spending
```

```
## [1] 447912.5
```

```
mailing_cost <- mailed_to_people * 2
mailing_cost
```

```
## [1] 180000
```

```
random_mailing_gross_profit <- mailed_to_exp_spending - mailing_cost
random_mailing_gross_profit
```

```
## [1] 267912.5
```

Without any kind of targeting, we assume Tiger Software would randomly mail catalogs to half of the pool which would be 90,000 people. At a cost of $2/mailing this would cost the company $180,000 in printing, postage and other mailing costs. From the data provided, we determined average actual spending of purchasers to be $93.90. With a response rate of approximately 5%, this would yield $447,913 in revenue making gross profit of the randomly sampled mailing for the firm $267,913.

5. Using the cumulative lift from 3(g), estimate the gross profit that would result from mailing to the 180,000 names selected using your data mining models. Comment on the value of your modeling effort.

Tiger Software would see a significant lift in gross profit by leveraging a targeting model such as ours. Using our model to target people more likely to respond (those with a predicted probability of response greater than 0.5), Tiger Software would send out fewer mailings to people more likely to respond. Of the pool of 180,000 people they would send mailings to 74,880 people which would cost $149,760. We determined average estimated spending using the preditions from our regression model of those likely to make a purchase (probability > 0.5) to be $212.15. Our model has a true positive rate of 36.6%, which means that we correctly identified 27,406 purchasers of the 74,880 people we mailed to yielding $5,814,306 in revenue. By targeting those more likely to make a purchase using our model Tiger Software would see a gross profit of $5,664,546 over 21 times higher than if they were to use random sampling.

Customer we mailed to (True Positives and False Positives)

```
test_preds <- factor(ifelse(purchase_predictions[,1] > .5, 1, 0), levels = c(0, 1))
confusion_matrix <- confusionMatrix(test_preds, test$Purchase)
confusion_matrix <- data.frame(confusion_matrix$table)
confusion_matrix <- confusion_matrix %>%
  mutate(total = sum(Freq), pct = Freq / total)
confusion_matrix
```

```
##   Prediction Reference Freq total   pct
## 1          0         0  231   500 0.462
## 2          1         0   25   500 0.050
## 3          0         1   61   500 0.122
## 4          1         1  183   500 0.366
```

```r
total_positive_rate <- confusion_matrix %>%
  filter(Prediction == 1) %>%
  summarise(Positives = sum(pct)) %>%
  select(Positives) %>% as.numeric()
total_positive_rate
```

```
## [1] 0.416
```

```r
customers_mailedto <- (total_positive_rate) * 180000
customers_mailedto
```

```
## [1] 74880
```

Profitable customers (True Positives)

```r
true_positive_rate <- confusion_matrix %>%
  filter(Prediction == 1 & Reference == 1) %>%
  select(pct) %>% as.numeric()
true_positive_rate
```

```
## [1] 0.366
```

```r
profitable_customers <- (true_positive_rate) * customers_mailedto
profitable_customers
```

```
## [1] 27406.08
```

```r
# average predicted spending per person from our model
avg_predicted_spending <- score_analysis %>%
  filter(pred_purchase_prob > 0.5) %>% # spending predictions for predicted purchasers
  summarise(mean_pred_spending = mean(pred_spending, na.rm = T)) %>%
  as.numeric()
avg_predicted_spending
```

```
## [1] 212.1539
```

Calculate total predicted spending (revenue)

```r
(targeted_revenue <- profitable_customers * avg_predicted_spending)
```

```
## [1] 5814306
```

Calculate gross profit

```r
(total_mailing_cost <- customers_mailedto * 2)
```

```
## [1] 149760
```

```
(targeted_gross_profit <- targeted_revenue - total_mailing_cost)
```

```
## [1] 5664546
```