Analyzing 2022 NFL 4th Down Success by Team

Jack Miller and Alex Jackson

Introduction

Our project examines NFL 4th down decision making using play-by-play data dating back to 1999. Our goal is to build three models that predict expected win probability added for each of the three decisions coaches have on fourth downs: kick a field goal, punt it, or go for it.

Exploratory Data Analysis

Our data comes from the `nflfastR` package which contains data on every play dating back to 1999.

```
data_99 <- nflfastR::load_pbp(1999) %>%
  mutate(year = 1999)
data_00 <- nflfastR::load_pbp(2000) %>%
  mutate(year = 2000)
data_01 <- nflfastR::load_pbp(2001) %>%
  mutate(year = 2001)
data_02 <- nflfastR::load_pbp(2002) %>%
  mutate(year = 2002)
data_03 <- nflfastR::load_pbp(2003) %>%
  mutate(year = 2003)
data_04 <- nflfastR::load_pbp(2004) %>%
  mutate(year = 2004)
data_05 <- nflfastR::load_pbp(2005) %>%
  mutate(year = 2005)
data_06 <- nflfastR::load_pbp(2006) %>%
  mutate(year = 2006)
data_07 <- nflfastR::load_pbp(2007) %>%
  mutate(year = 2007)
data_08 <- nflfastR::load_pbp(2008) %>%
  mutate(year = 2008)
data_09 <- nflfastR::load_pbp(2009) %>%
  mutate(year = 2009)
data_10 <- nflfastR::load_pbp(2010) %>%
  mutate(year = 2010)
data_11 <- nflfastR::load_pbp(2011) %>%
  mutate(year = 2011)
data_12 <- nflfastR::load_pbp(2012) %>%
  mutate(year = 2012)
data_13 <- nflfastR::load_pbp(2013) %>%
  mutate(year = 2013)
data_14 <- nflfastR::load_pbp(2014) %>%
  mutate(year = 2014)
```

```
data_15 <- nflfastR::load_pbp(2015) %>%
  mutate(year = 2015)
data_16 <- nflfastR::load_pbp(2016) %>%
  mutate(year = 2016)
data_17 <- nflfastR::load_pbp(2017) %>%
  mutate(year = 2017)
data_18 <- nflfastR::load_pbp(2018) %>%
  mutate(year = 2018)
data_19 <- nflfastR::load_pbp(2019) %>%
  mutate(year = 2019)
data_20 <- nflfastR::load_pbp(2020) %>%
  mutate(year = 2020)
data_21 <- nflfastR::load_pbp(2021) %>%
  mutate(year = 2021)
data_22 <- nflfastR::load_pbp(2022) %>%
  mutate(year = 2022)
data <- rbind(data_99, data_00, data_01, data_02, data_03, data_04, data_05,
              data_06, data_07, data_08, data_09, data_10, data_11, data_12,
              data_13, data_14, data_15, data_16, data_17, data_18, data_19,
              data_20, data_21, data_22)
data_main <- data %>%
  select(yardline_100, game_seconds_remaining, qtr,
         down, ydstogo, score_differential, no_score_prob,
         opp_fg_prob, opp_safety_prob, opp_td_prob, fg_prob, safety_prob,
         td_prob, wp, fourth_down_converted, fourth_down_failed,
         posteam_timeouts_remaining, defteam_timeouts_remaining, ep, epa, year,
         week, posteam, posteam_type, play_type, wpa, total_home_epa, total_away_epa) %>%
  filter(down == 4) %>%
  mutate(play_type = as.factor(ifelse(play_type == "run" | play_type == "pass", "go", play_type))) %>%
  drop_na(play_type, wpa) %>%
  filter(play_type %in% c("go", "field_goal", "punt"))

run_pass <- data %>%
  select(yardline_100, game_seconds_remaining, qtr,
         down, ydstogo, score_differential, no_score_prob,
         opp_fg_prob, opp_safety_prob, opp_td_prob, fg_prob, safety_prob,
         td_prob, wp, fourth_down_converted, fourth_down_failed,
         posteam_timeouts_remaining, defteam_timeouts_remaining, ep, epa, year,
         week, posteam, posteam_type, play_type, wpa, total_home_epa, total_away_epa) %>%
  filter(down == 4,
         play_type %in% c("run", "pass")) %>%
  drop_na(wpa)
```

```
data_main %>%
  group_by(play_type) %>%
  summarize(count = n()) %>%
  kable()
```

play_type	count
field_goal	21973
go	12630

play_type	count
punt	58812

Modeling

```
punt_data <- data_main %>%
  filter(play_type == "punt") %>%
  select(-c(qtr, down, fourth_down_converted, fourth_down_failed, ep, epa, year,
            week, posteam, play_type, posteam_type)) %>%
  select(wpa, everything())
kick_data <- data_main %>%
  filter(play_type == "field_goal") %>%
  select(-c(qtr, down, fourth_down_converted, fourth_down_failed, ep, epa, year,
            week, posteam, play_type, posteam_type)) %>%
  select(wpa, everything())
go_data <- data_main %>%
  filter(play_type == "go") %>%
  select(-c(qtr, down, fourth_down_converted, fourth_down_failed, ep, epa, year,
            week, posteam, play_type, posteam_type)) %>%
  select(wpa, everything())
```

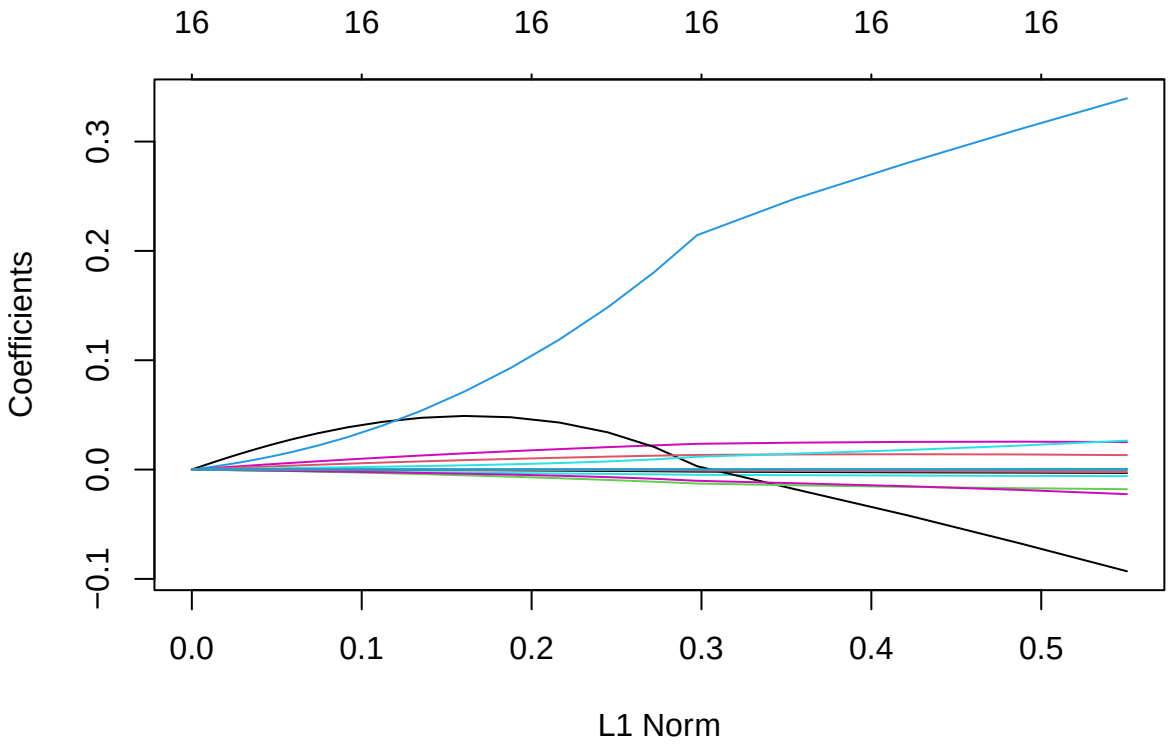
```
set.seed(12321)
sample <- sample(c(TRUE, FALSE), nrow(punt_data), replace = TRUE, prob = c(0.7, 0.3))
punt_data_train <- punt_data[sample, ]
punt_data_test <- punt_data[!sample, ]
punt_train_x <- data.matrix(punt_data_train[, -1])
punt_train_y <- punt_data_train$wpa
punt_test_x <- data.matrix(punt_data_test[, -1])
punt_test_y <- punt_data_test$wpa
punt_xgb_train = xgb.DMatrix(data = punt_train_x, label = punt_train_y)
punt_xgb_test = xgb.DMatrix(data = punt_test_x, label = punt_test_y)
punt_watchlist = list(train=punt_xgb_train, test=punt_xgb_test)
punt_model = xgb.train(data = punt_xgb_train, max.depth = 3, watchlist=punt_watchlist, nrounds = 1000, 
punt_model_xgboost = xgboost(data = punt_xgb_train, max.depth = 3, nrounds = 995, verbose = 0)
punt_pred_y = predict(punt_model_xgboost, punt_xgb_test)
mean((punt_test_y - punt_pred_y)^2)
```

```
## [1] 0.0005847609
```

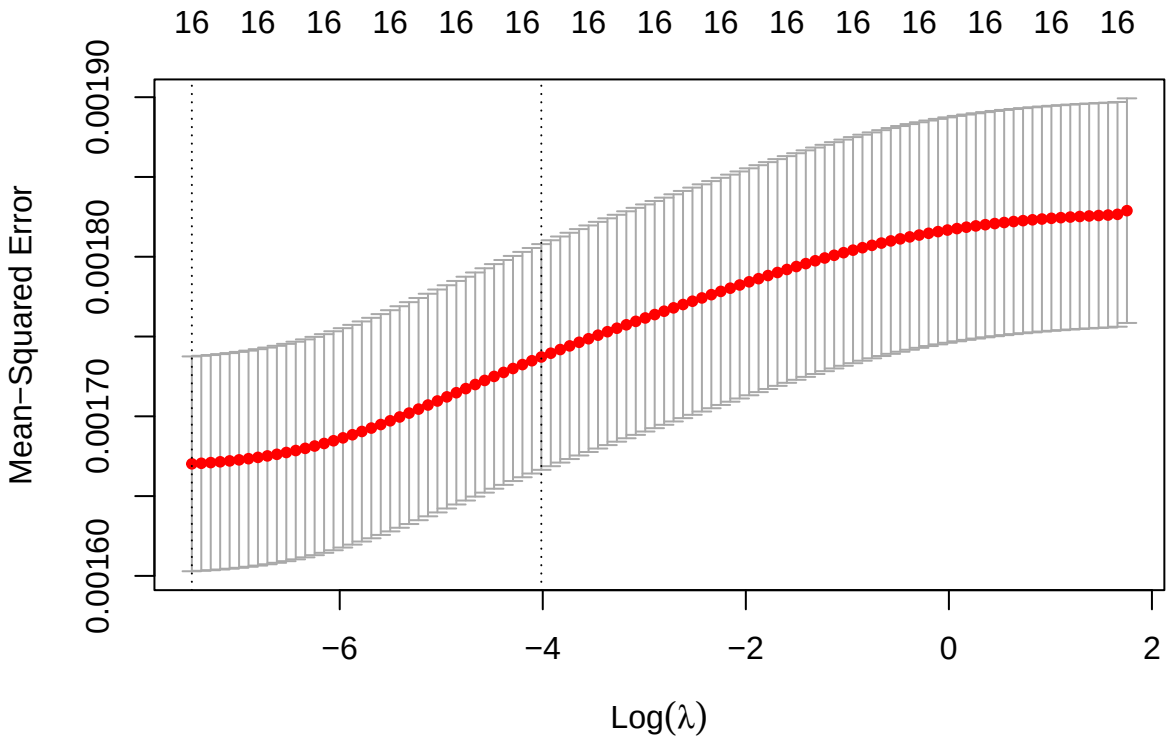
```
RMSE(punt_test_y, punt_pred_y)
```

```
## [1] 0.02418183
```

```
grid <- 10^seq(10, -2, length = 100)
lasso.mod <- glmnet(punt_train_x, punt_train_y, alpha = 0, lambda = grid)
plot(lasso.mod)
```




```
cv.out <- cv.glmnet(punt_train_x, punt_train_y, alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam, newx = punt_test_x)
mean((lasso.pred - punt_test_y)^2)
```

```
## [1] 0.001735869
```

```
RMSE(punt_test_y, lasso.pred)
```

```
## [1] 0.04166377
```

```
ridge.mod <- glmnet(punt_train_x, punt_train_y, alpha = 0, lambda = grid,
                    thresh = 1e-12)
ridge.pred <- predict(ridge.mod, s = 4, newx = punt_test_x)
mean((ridge.pred - punt_test_y)^2)
```

```
## [1] 0.001837132
```

```
RMSE(punt_test_y, ridge.pred)
```

```
## [1] 0.04286178
```

XG Boost model performs the best for predicting WPA of punts compared to the optimal lasso and ridge regression models.

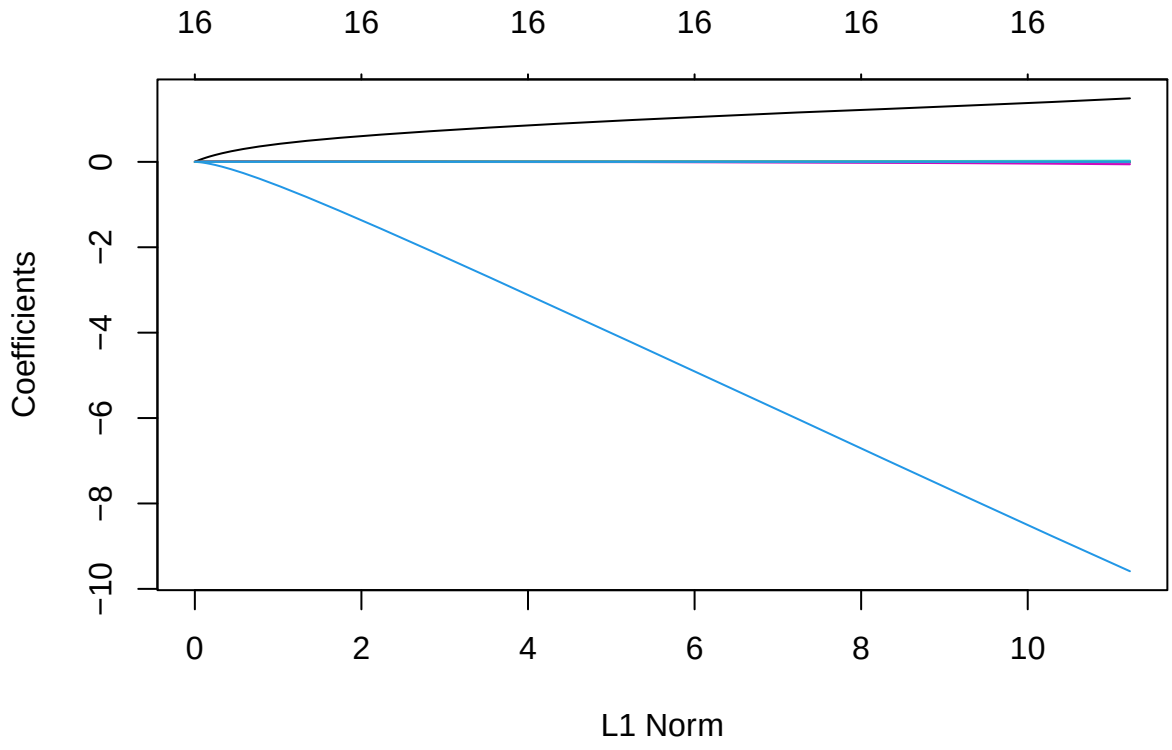
```
set.seed(12321)
sample <- sample(c(TRUE, FALSE), nrow(kick_data), replace = TRUE, prob = c(0.7, 0.3))
kick_data_train <- kick_data[sample, ]
kick_data_test <- kick_data[!sample, ]
kick_train_x <- data.matrix(kick_data_train[, -1])
kick_train_y <- kick_data_train$wpa
kick_test_x <- data.matrix(kick_data_test[, -1])
kick_test_y <- kick_data_test$wpa
kick_xgb_train = xgb.DMatrix(data = kick_train_x, label = kick_train_y)
kick_xgb_test = xgb.DMatrix(data = kick_test_x, label = kick_test_y)
kick_watchlist = list(train=kick_xgb_train, test=kick_xgb_test)
kick_model = xgb.train(data = kick_xgb_train, max.depth = 3, watchlist=kick_watchlist, nrounds = 400, v
kick_model_xgboost = xgboost(data = kick_xgb_train, max.depth = 3, nrounds = 370, verbose = 0)
kick_pred_y = predict(kick_model_xgboost, kick_xgb_test)
mean((kick_test_y - kick_pred_y)^2)
```

```
## [1] 0.001967498
```

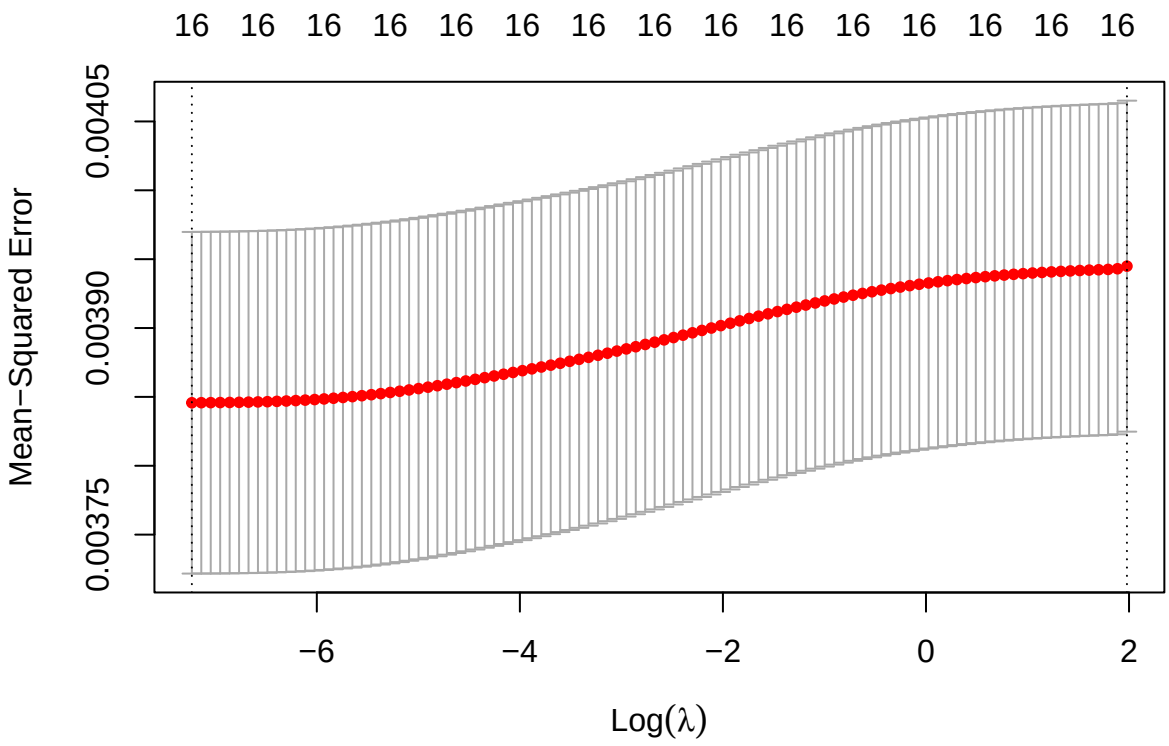
```
RMSE(kick_test_y, kick_pred_y)
```

```
## [1] 0.04435649
```

```
lasso.mod.kick <- glmnet(kick_train_x, kick_train_y, alpha = 0, lambda = grid)
plot(lasso.mod.kick)
```



```
cv.out.kick <- cv.glmnet(kick_train_x, kick_train_y, alpha = 0)
plot(cv.out.kick)
```




```
bestlam <- cv.out.kick$lambda.min
lasso.pred.kick <- predict(lasso.mod.kick, s = bestlam, newx = kick_test_x)
mean((lasso.pred.kick - kick_test_y)^2)
```

```
## [1] 0.003729859
```

```
RMSE(kick_test_y, lasso.pred.kick)
```

```
## [1] 0.06107257
```

```
ridge.mod.kick <- glmnet(kick_train_x, kick_train_y, alpha = 0, lambda = grid,
                    thresh = 1e-12)
ridge.pred.kick <- predict(ridge.mod.kick, s = 4, newx = kick_test_x)
mean((ridge.pred.kick - kick_test_y)^2)
```

```
## [1] 0.003812989
```

```
RMSE(kick_test_y, ridge.pred.kick)
```

```
## [1] 0.0617494
```

XG Boost model performs the best for predicting WPA of field goals compared to the optimal lasso and ridge regression models.

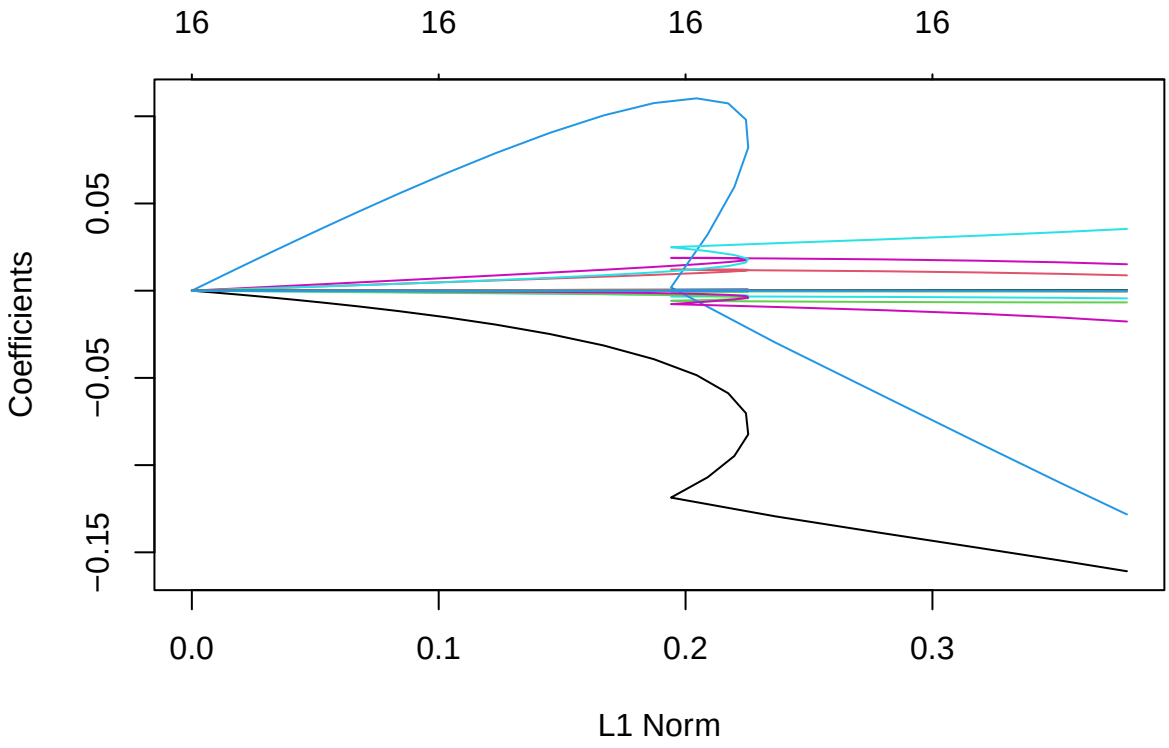
```
set.seed(12321)
sample <- sample(c(TRUE, FALSE), nrow(go_data), replace = TRUE, prob = c(0.7, 0.3))
go_data_train <- go_data[sample, ]
go_data_test <- go_data[!sample, ]
go_train_x <- data.matrix(go_data_train[, -1])
go_train_y <- go_data_train$wpa
go_test_x <- data.matrix(go_data_test[, -1])
go_test_y <- go_data_test$wpa
go_xgb_train = xgb.DMatrix(data = go_train_x, label = go_train_y)
go_xgb_test = xgb.DMatrix(data = go_test_x, label = go_test_y)
go_watchlist = list(train=go_xgb_train, test=go_xgb_test)
go_model = xgb.train(data = go_xgb_train, max.depth = 3, watchlist=go_watchlist, nrounds = 250, verbose
go_model_xgboost = xgboost(data = go_xgb_train, max.depth = 3, nrounds = 204, verbose = 0)
go_pred_y = predict(go_model_xgboost, go_xgb_test)
mean((go_test_y - go_pred_y)^2)
```

```
## [1] 0.003556852
```

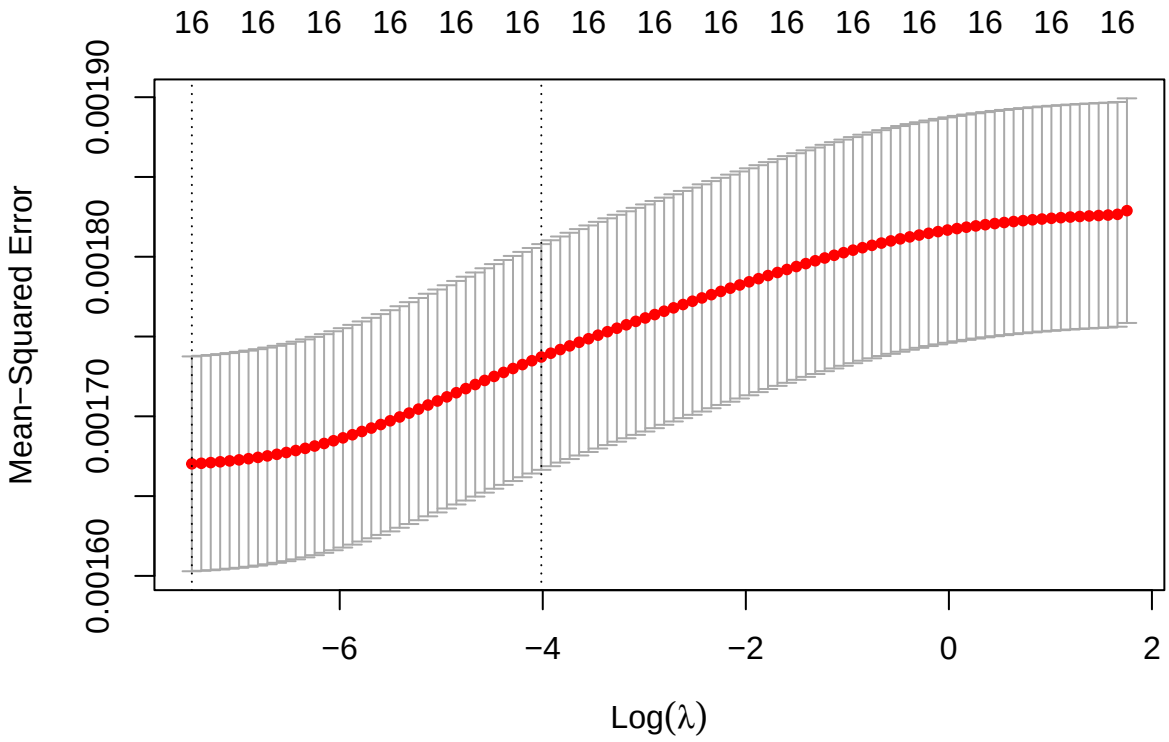
```
RMSE(go_test_y, go_pred_y)
```

```
## [1] 0.05963935
```

```
lasso.mod.go <- glmnet(go_train_x, go_train_y, alpha = 0, lambda = grid)
plot(lasso.mod.go)
```




```
cv.out.go <- cv.glmnet(go_train_x, go_train_y, alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out.go$lambda.min
lasso.pred.go <- predict(lasso.mod.go, s = bestlam, newx = go_test_x)
mean((lasso.pred.go - go_test_y)^2)
```

```
## [1] 0.007758653
```

```
RMSE(go_test_y, lasso.pred)
```

```
## [1] 0.08904686
```

```
ridge.mod.go <- glmnet(go_train_x, go_train_y, alpha = 0, lambda = grid,
                       thresh = 1e-12)
ridge.pred.go <- predict(ridge.mod.go, s = 4, newx = go_test_x)
mean((ridge.pred.go - go_test_y)^2)
```

```
## [1] 0.007926037
```

```
RMSE(go_test_y, ridge.pred.go)
```

```
## [1] 0.08902829
```

XG Boost model performs the best for predicting WPA of going for it compared to the optimal lasso and ridge regression models.

Discussion and Conclusion