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.

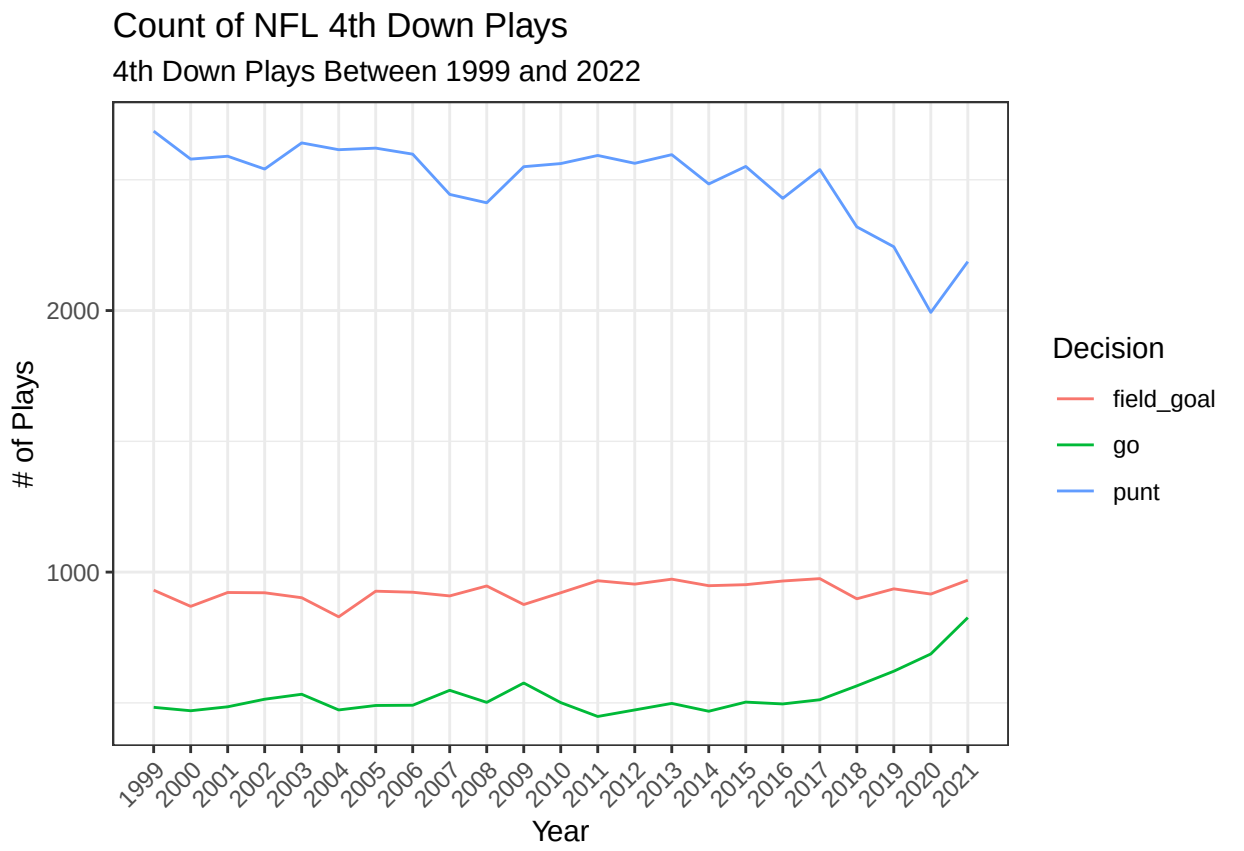
Exploratory Data Analysis

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

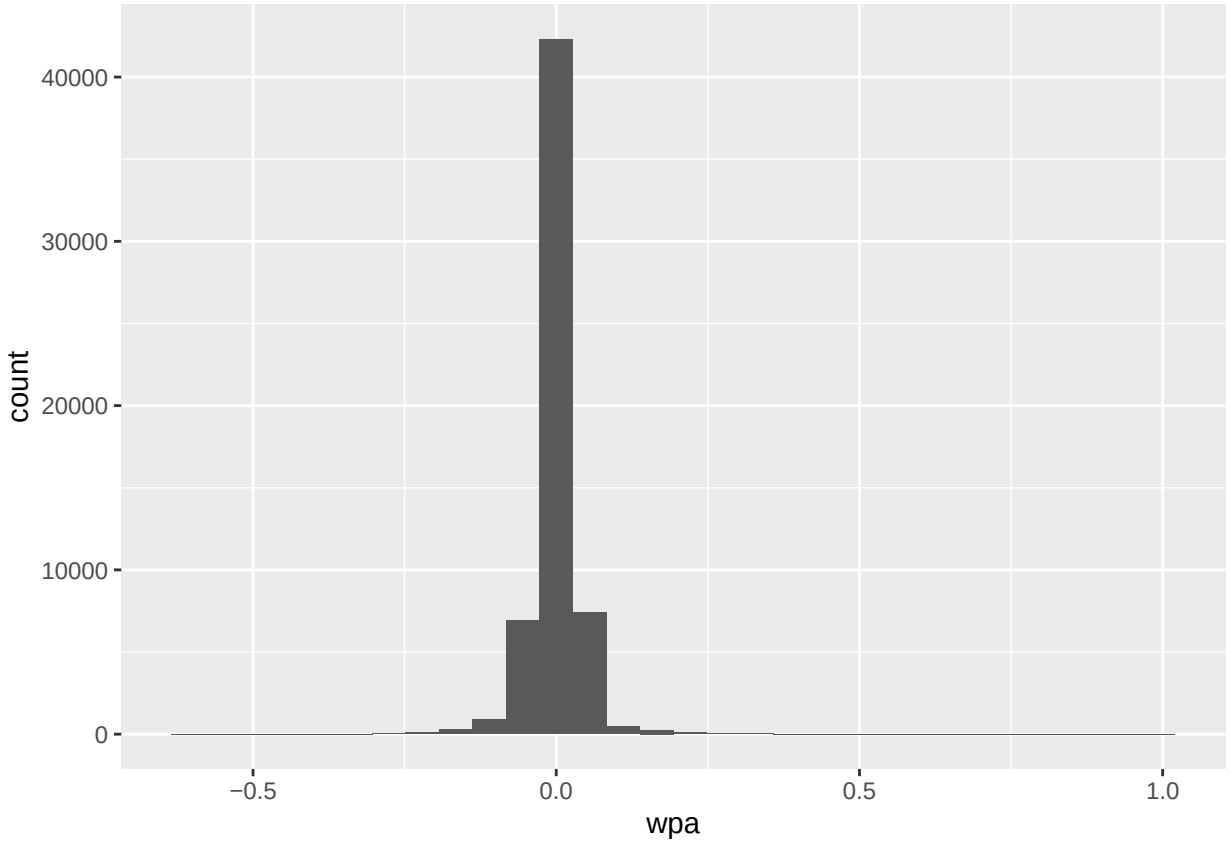
```
data_main %>%
  group_by(play_type) %>%
  summarize(count = n(), perc = n()*100/nrow(data_main)) %>%
  kable(digits = 1)
```

play_type	count	perc
field_goal	22037	23.5
go	12672	13.5
punt	58923	62.9

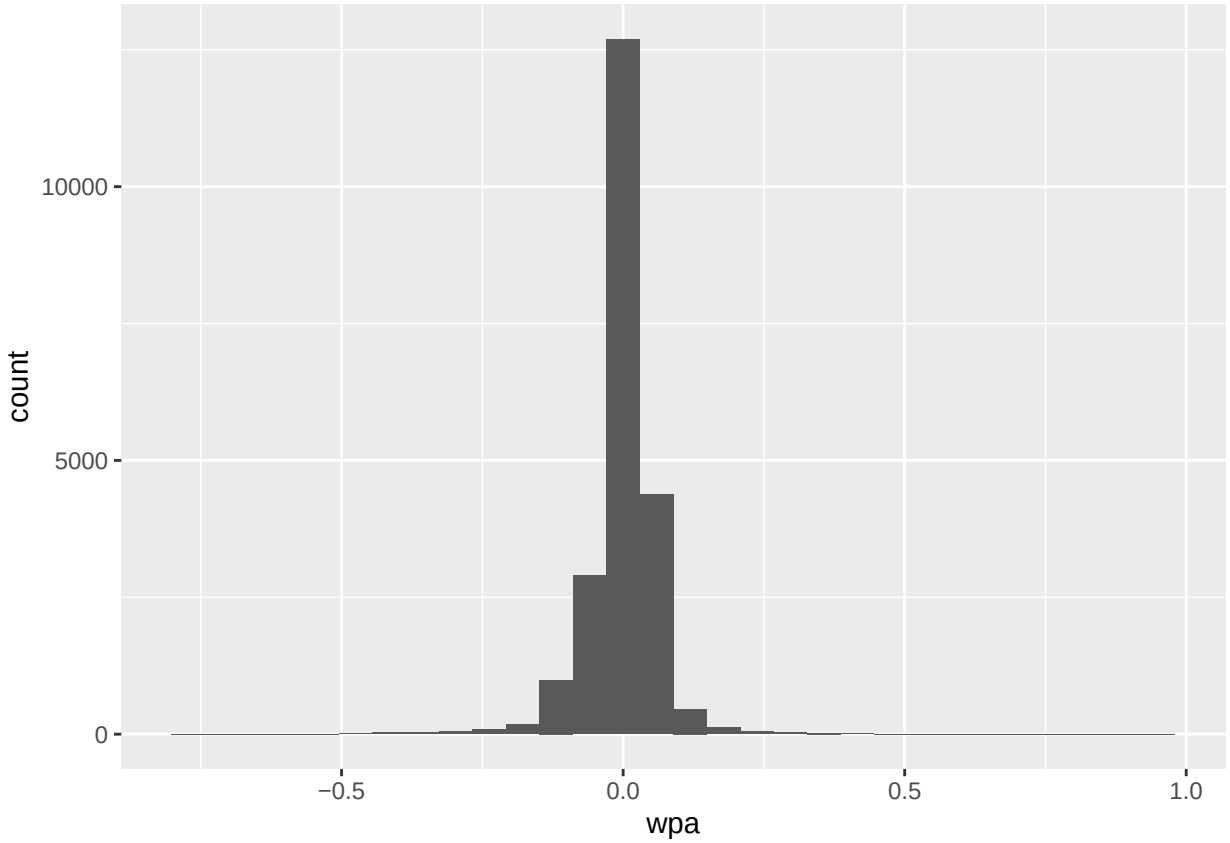
```
ggplot(data_main, aes(x=year)) + geom_line(aes(fill=..count.., color = play_type), stat="bin", binwidth=
  scale_x_continuous(breaks = c(1999:2021), labels = c(1999:2021), limits = c(1999, 2021)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust=1),
        panel.grid.minor.x = element_blank()) +
  labs(title = "Count of NFL 4th Down Plays",
       subtitle = "4th Down Plays Between 1999 and 2022",
       x = "Year", y = "# of Plays",
       color = "Decision")
```



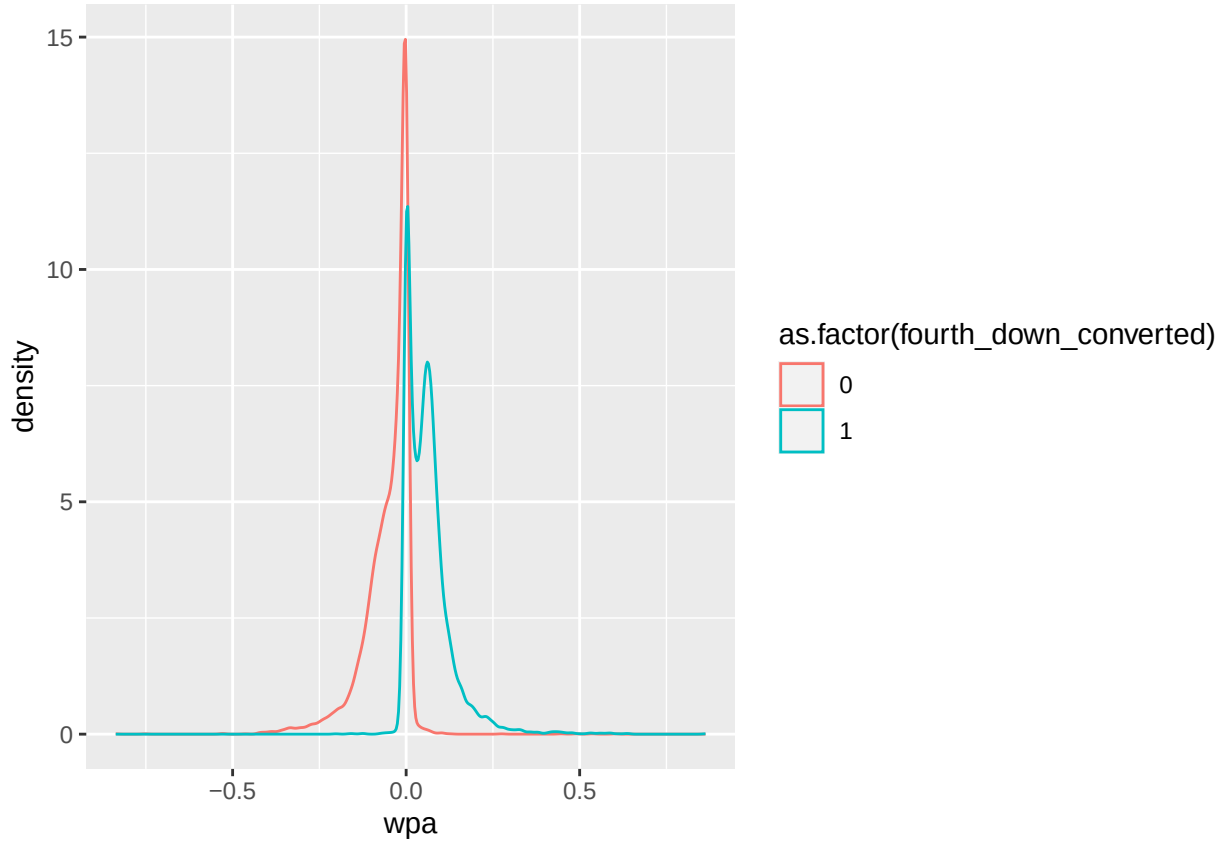

```
ggplot(punt_data, aes(x = wpa)) +
  geom_histogram()
```



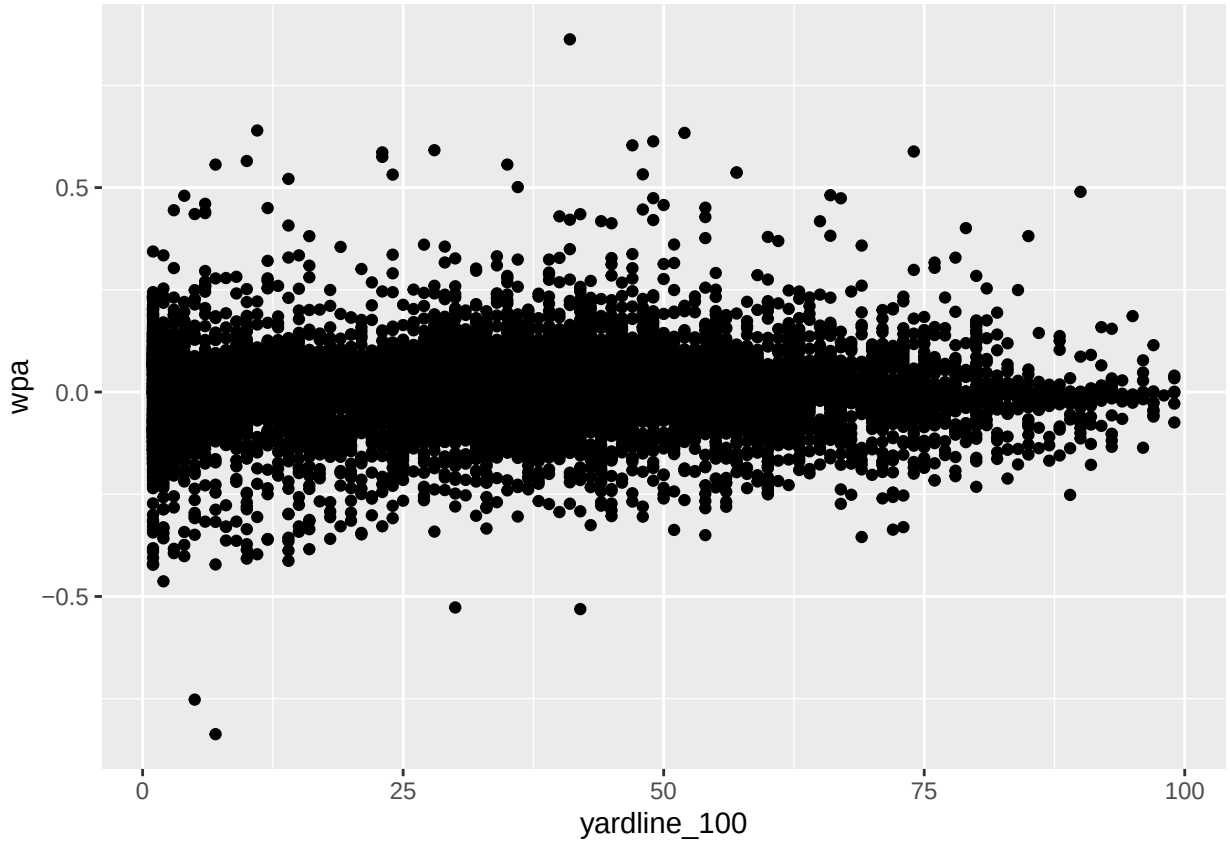

```
ggplot(kick_data, aes(x = wpa)) +
  geom_histogram()
```



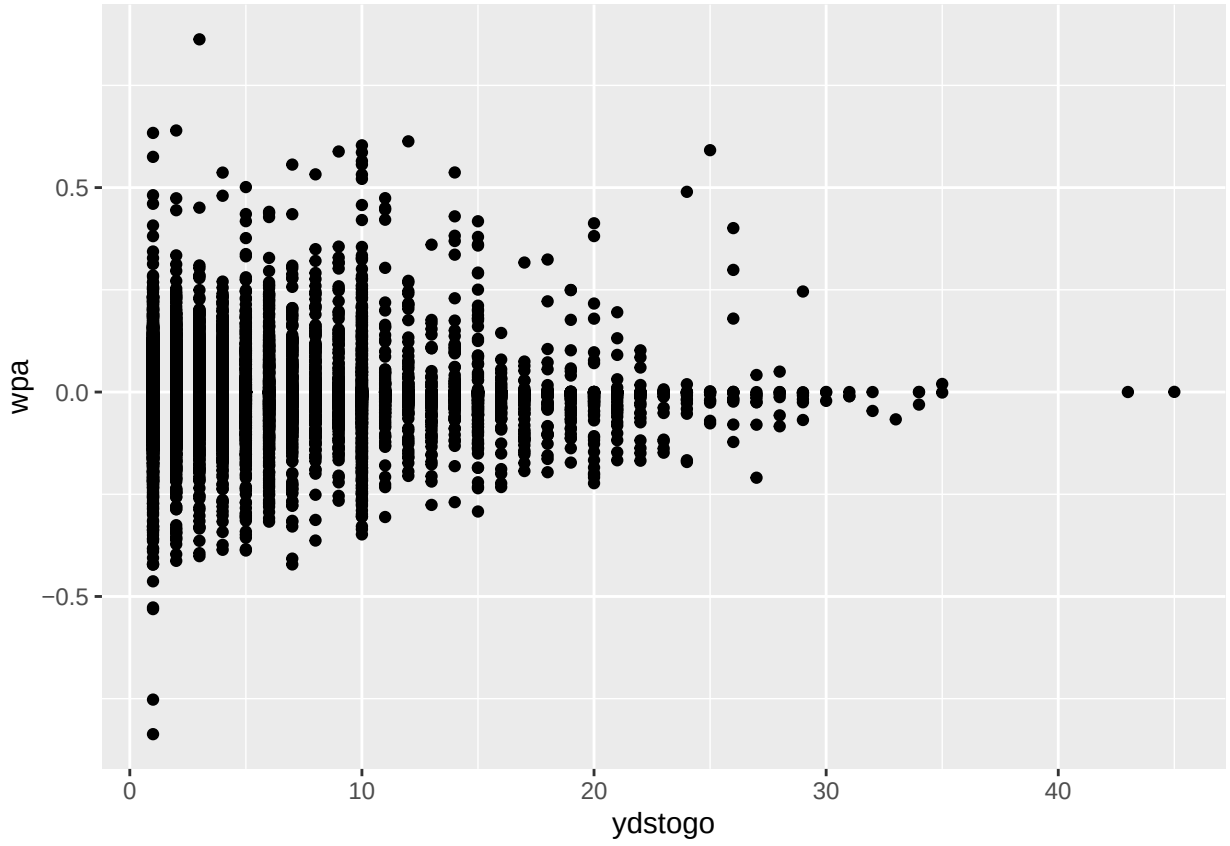

```
ggplot(go_data, aes(x = wpa, color = as.factor(fourth_down_converted))) +
  geom_density()
```




```
ggplot(go_data, aes(x = yardline_100, y = wpa)) +
  geom_point()
```




```
ggplot(go_data, aes(x = ydstogo, y = wpa)) +
  geom_point()
```




```
data_main %>%
  group_by(play_type) %>%
  summarise(mean_wpa = mean(wpa)) %>%
  kable(digits = 5)
```

play_type	mean_wpa
field_goal	-0.00037
go	0.00396
punt	-0.00044

Modeling

The goal with our models is to create the best possible models for predicting WPA (win probability added) given a certain game state.

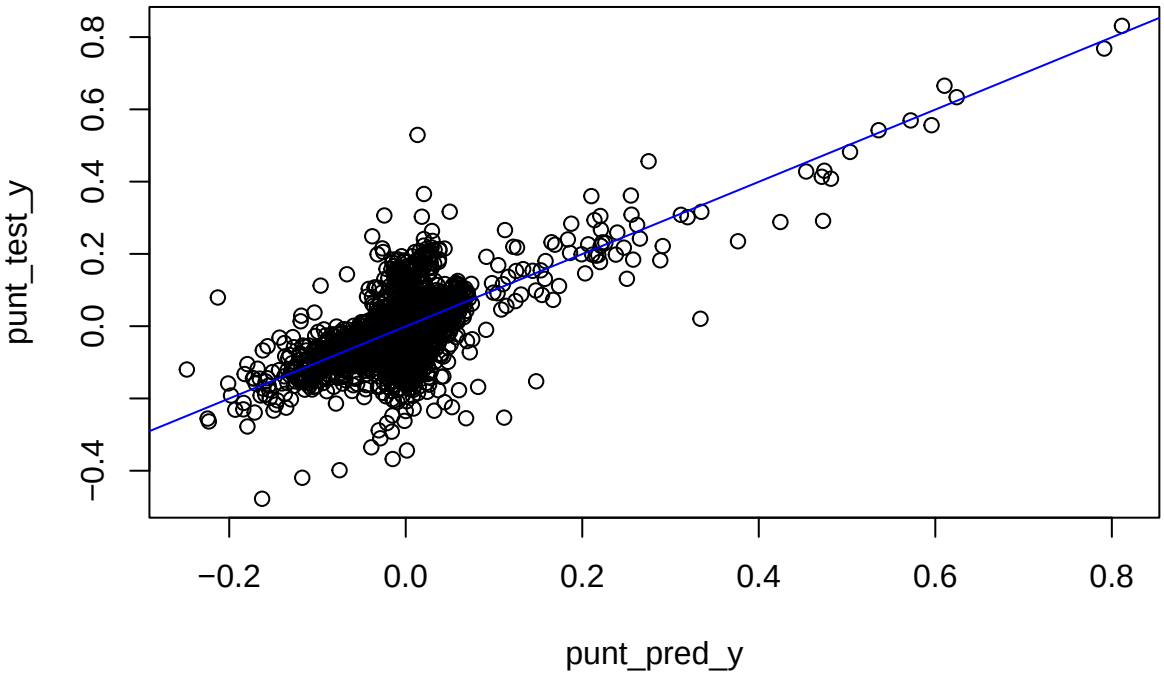
Punt Model

```
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, ]
```

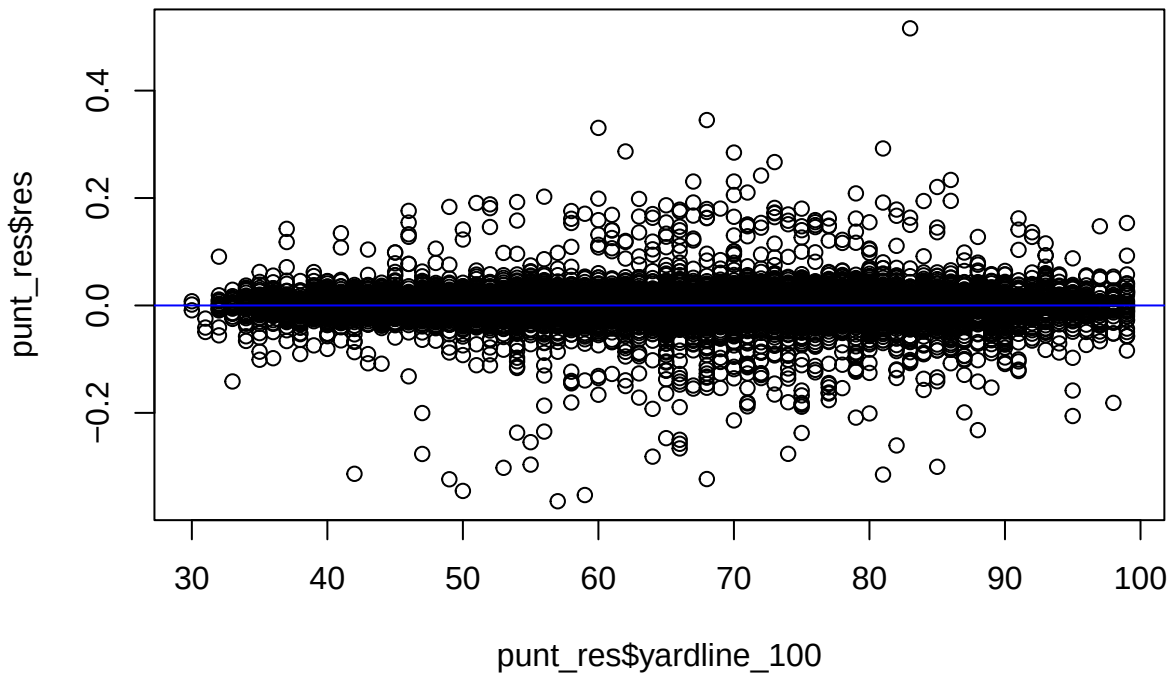
```
punt_data_test <- punt_data[!sample, ]
punt_train_x <- data.matrix(punt_data_train[, -1])
punt_train_y <- punt_data_train$wpa
punt_test_x <- data.matrix(punt_data_test[, -1])
punt_test_y <- punt_data_test$wpa
punt_xgb_train = xgb.DMatrix(data = punt_train_x, label = punt_train_y)
punt_xgb_test = 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 = 8,  colsample_bylevel = 1, colsample_bytree =
punt_model_xgboost = xgboost(data = punt_xgb_train, max_depth = 8,  colsample_bylevel = 1, colsample_by
punt_pred_y = predict(punt_model_xgboost, punt_xgb_test)
print(paste("XG Boost RMSE:", RMSE(punt_test_y, punt_pred_y)))
```

```
## [1] "XG Boost RMSE: 0.0294369905331241"
```

```
plot(y=punt_test_y, x=punt_pred_y)
abline(lm(punt_test_y ~ punt_pred_y), col = "blue")
```




```
punt_res = data.frame(test_y= punt_test_y, pred_y = punt_pred_y, res = (punt_test_y-punt_pred_y), punt_
plot(punt_res$yardline_100, punt_res$res)
abline(0,0, col="blue")
```



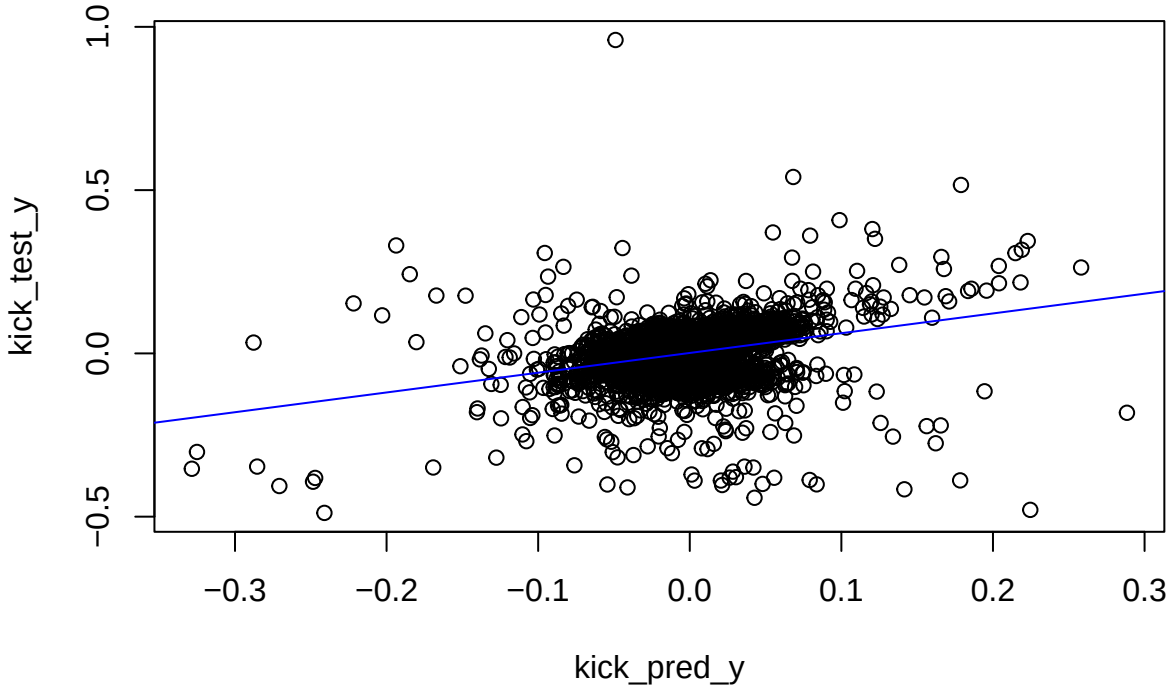
Our optimal punt XG Boost model has a RMSE of 0.0216, which is the lowest RMSE we obtained throughout our XG Boost, Lasso, and Ridge Regression modelling process. Therefore, we will continue with this as our punt WPA model.

Kick Model

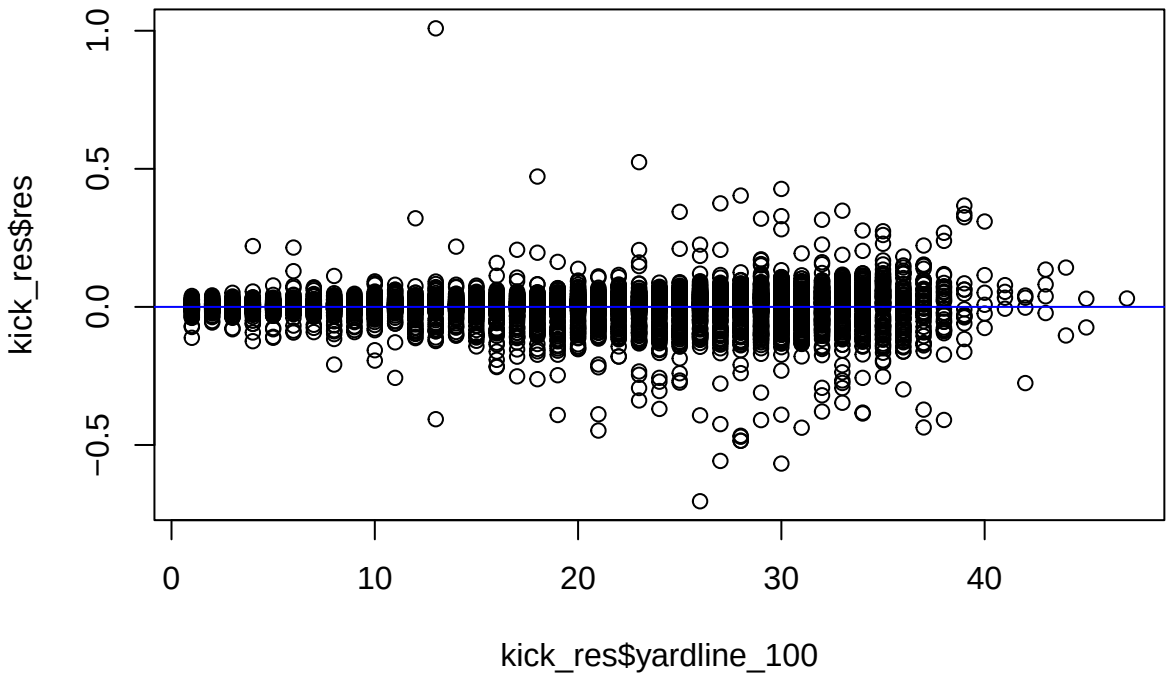
```
set.seed(12321)
sample <- sample(c(TRUE, FALSE), nrow(kick_data), replace = TRUE, prob = c(0.7, 0.3))
kick_data_train <- kick_data[sample, ]
kick_data_test <- kick_data[!sample, ]
kick_train_x <- data.matrix(kick_data_train[, -1])
kick_train_y <- kick_data_train$wpa
kick_test_x <- data.matrix(kick_data_test[, -1])
kick_test_y <- kick_data_test$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 = 6,  colsample_bylevel = 1, colsample_bytree =
kick_model_xgboost = xgboost(data = kick_xgb_train, max_depth = 6,  colsample_bylevel = 1, colsample_by
kick_pred_y = predict(kick_model_xgboost, kick_xgb_test)
print(paste("XG Boost RMSE:", RMSE(kick_test_y, kick_pred_y)))
```

```
## [1] "XG Boost RMSE: 0.0598292868845007"
```

```
plot(y=kick_test_y, x=kick_pred_y)
abline(lm(kick_test_y ~ kick_pred_y), col = "blue")
```




```
kick_res = data.frame(test_y= kick_test_y, pred_y = kick_pred_y, res = (kick_test_y-kick_pred_y), kick_
plot(kick_res$yardline_100, kick_res$res)
abline(0,0, col="blue")
```



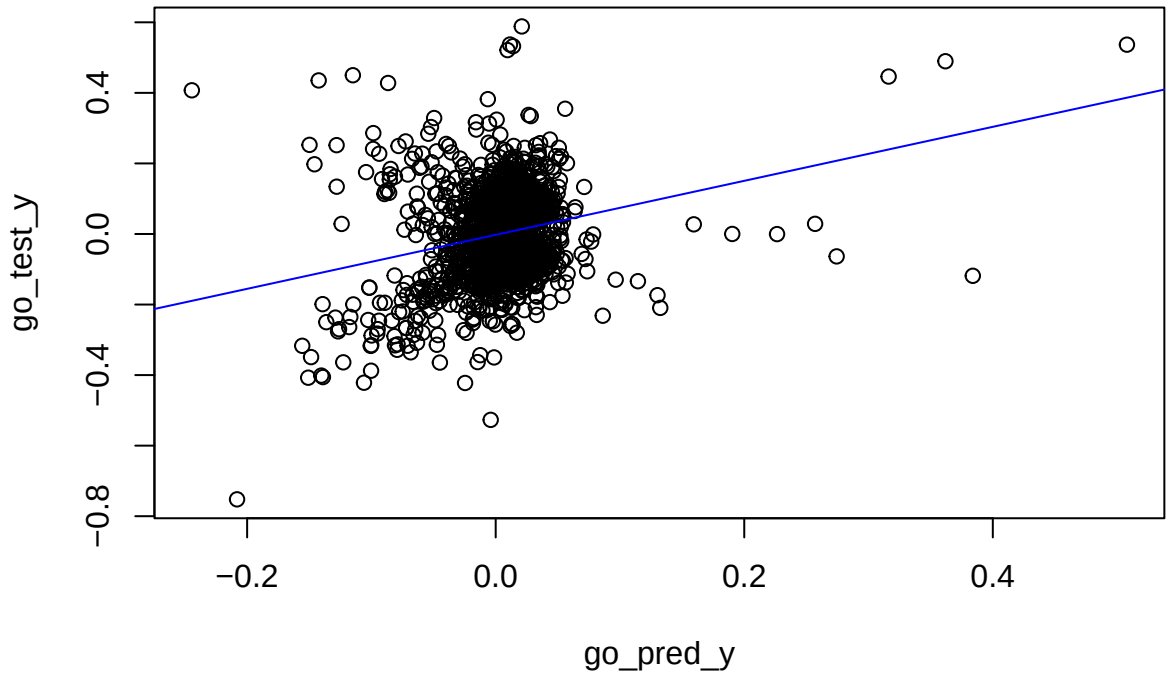
Our optimal kick XG Boost model has a RMSE of 0.3921, which is the lowest RMSE we obtained throughout our XG Boost, Lasso, and Ridge Regression modelling process. Therefore, we will continue with this as our kick WPA model.

Go Model

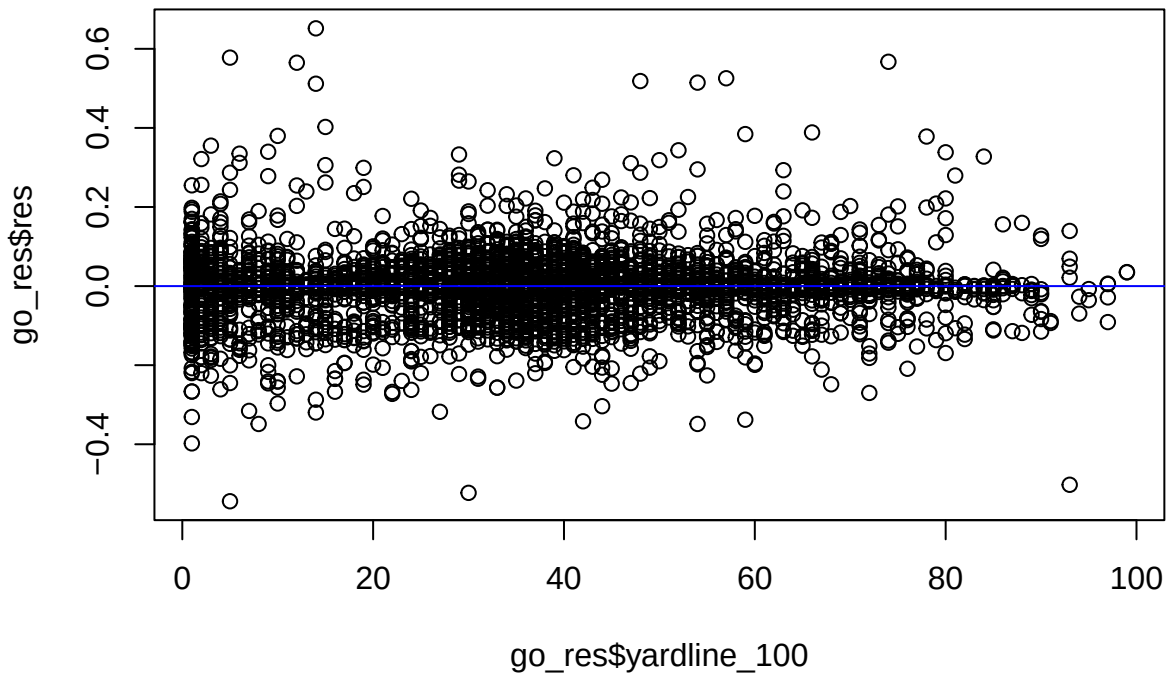
```
go_data <- go_data %>%
  select(-fourth_down_converted)

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 = 8,  colsample_bylevel = 1, colsample_bytree = 1, 
go_model_xgboost = xgboost(data = go_xgb_train, max_depth = 6,  colsample_bylevel = 1, colsample_bytree
go_pred_y = predict(go_model_xgboost, go_xgb_test)
print(paste("XG Boost RMSE:", RMSE(go_test_y, go_pred_y)))
```

```
## [1] "XG Boost RMSE: 0.0867906629769721"
```

```
plot(y=go_test_y, x=go_pred_y)
abline(lm(go_test_y ~ go_pred_y), col = "blue")
```




```
go_res = data.frame(test_y= go_test_y, pred_y = go_pred_y, res = (go_test_y-go_pred_y), go_test_x)
plot(go_res$yardline_100, go_res$res)
abline(0,0, col="blue")
```

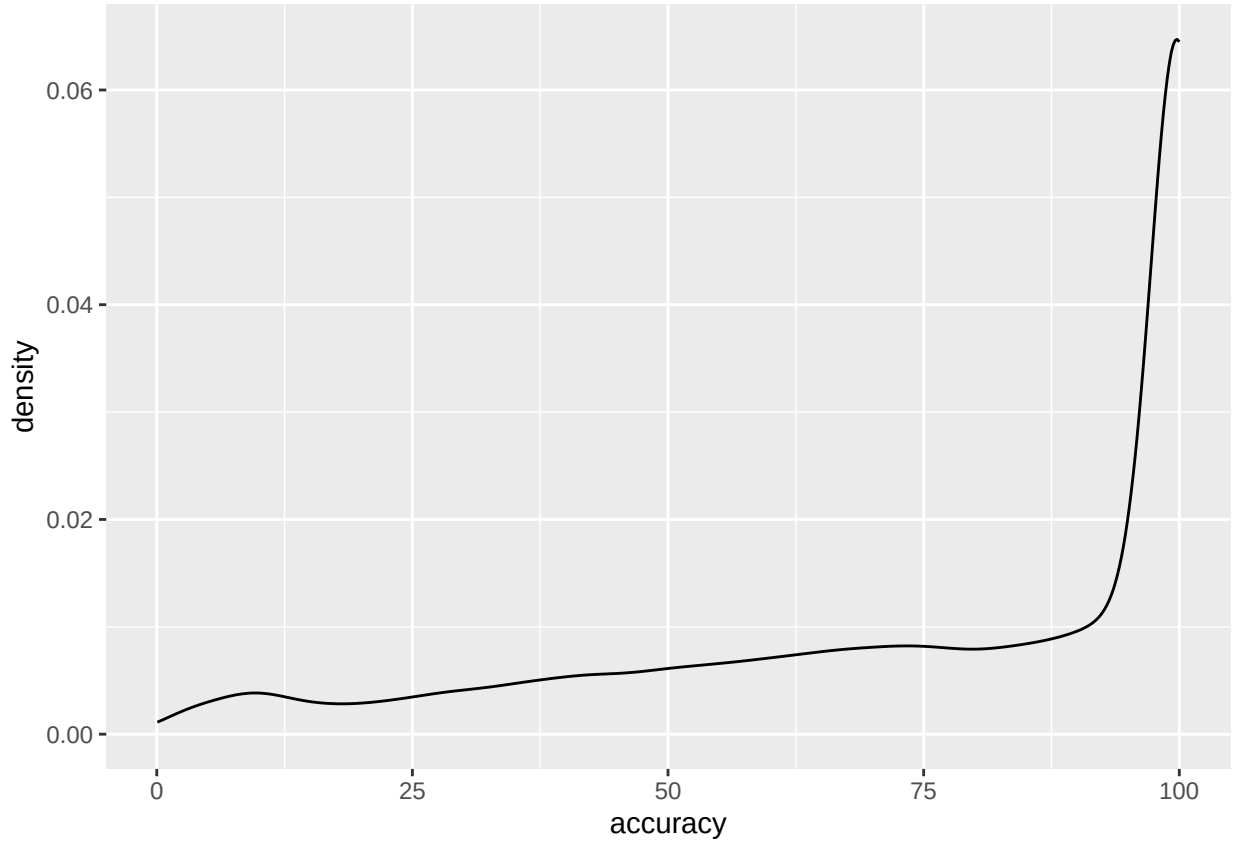


Our optimal go for it XG Boost model has a RMSE of 0.5492, which is the lowest RMSE we obtained throughout our XG Boost, Lasso, and Ridge Regression modelling process. Therefore, we will continue with this as our go for it WPA model.

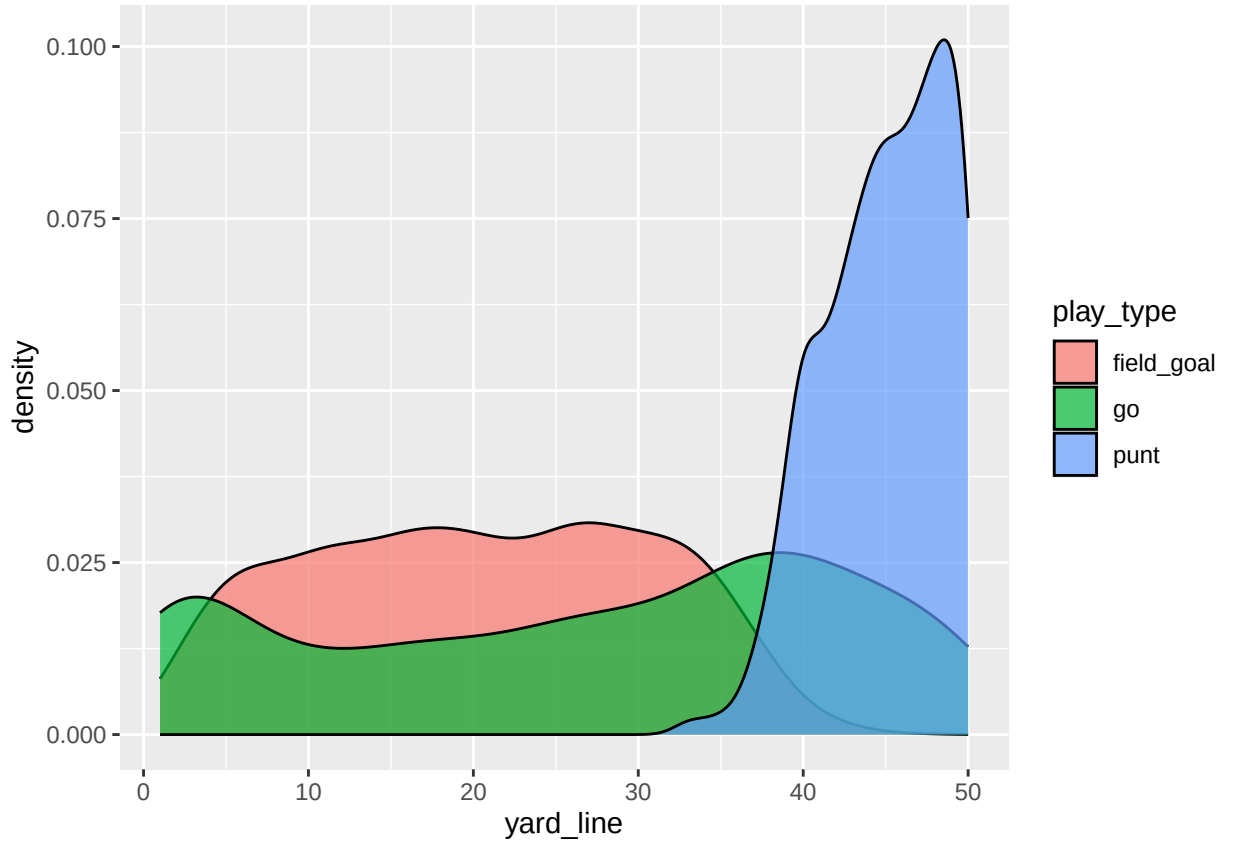
Results

```
pred_data <- data_main %>%
    select(-c(down, fourth_down_converted, fourth_down_failed, ep, epa, year,
              week, posteam, play_type, posteam_type)) %>%
  select(wpa, everything())

data_x <- data.matrix(pred_data[, -1])
data_y <- pred_data$wpa
xgb_matrix = xgb.DMatrix(data = data_x, label = data_y)

ewpa_punt = predict(punt_model_xgboost, xgb_matrix)
ewpa_kick = predict(kick_model_xgboost, xgb_matrix)
ewpa_go = predict(go_model_xgboost, xgb_matrix)

pred_data <- pred_data %>%
  mutate(yard_line = ifelse(yardline_100 > 50, -(100-yardline_100), yardline_100),
         game_minutes_remaining = paste0(trunc(game_seconds_remaining/60), ":", game_seconds_remaining%%

pred_data = cbind(pred_data, play_type = data_main$play_type, team = data_main$posteam, year = data_mai
```

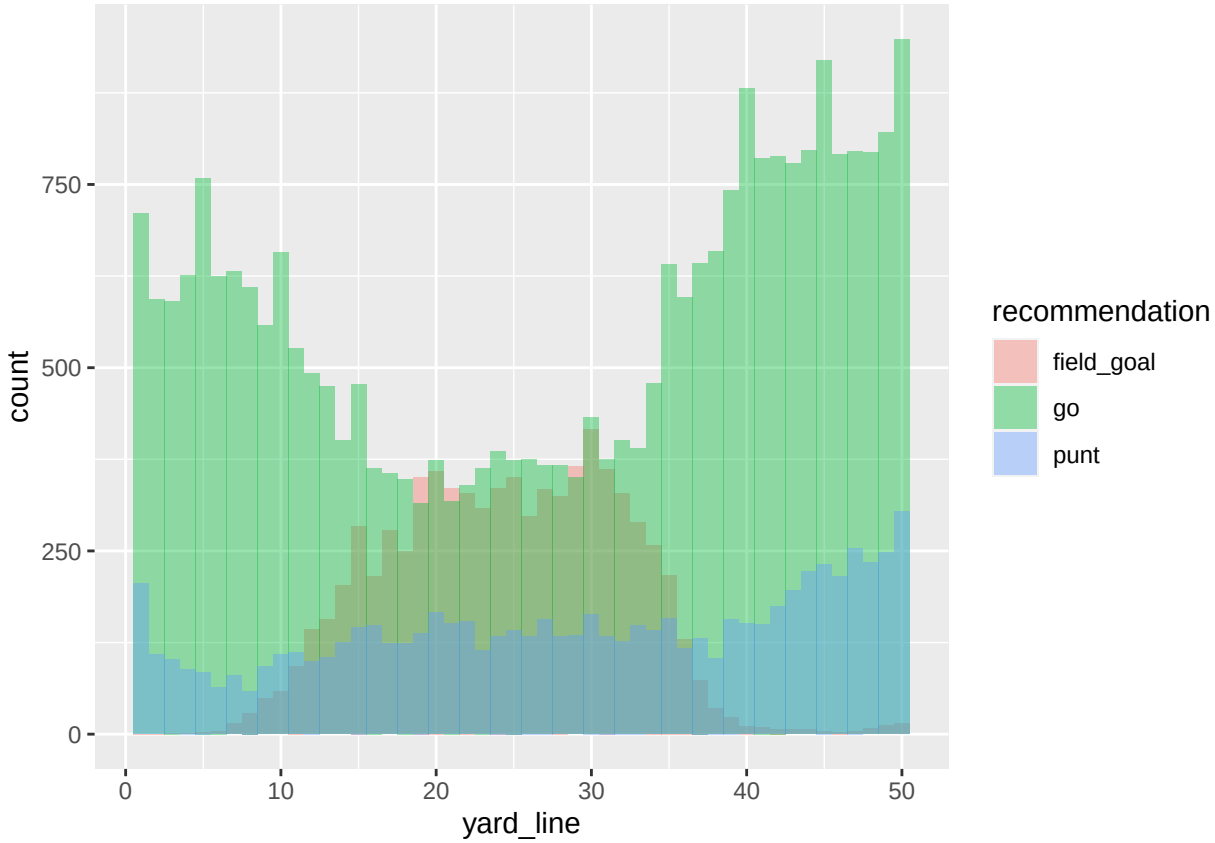
```
  mutate(
    #ewpa_punt = ifelse(yard_line < 31, -1, ewpa_punt),
         recommendation = case_when((ewpa_punt > ewpa_kick) & (ewpa_punt > ewpa_go) ~ "punt",
                                    (ewpa_kick > ewpa_punt) & (ewpa_kick > ewpa_go) ~ "field_goal",
                                    (ewpa_go > ewpa_kick) & (ewpa_go > ewpa_punt) ~ "go"),
         correct_call = ifelse(play_type == recommendation, "Yes", "No"),
         ewpa_max = pmax(ewpa_punt, ewpa_kick, ewpa_go),
         ewpa_play = case_when((play_type == "punt") ~ ewpa_punt,
                               (play_type == "field_goal") ~ ewpa_kick,
                               (play_type == "go") ~ ewpa_go),
         execution = wpa - ewpa_max)

pred_data$ewpa_max_percentile <- ecdf(pred_data$ewpa_max)(pred_data$ewpa_max)
pred_data$ewpa_play_percentile <- ecdf(pred_data$ewpa_max)(pred_data$ewpa_play)
pred_data$accuracy <- 100 * (1 - (pred_data$ewpa_max_percentile - pred_data$ewpa_play_percentile))

#pred_data %>%
  #filter(correct_call == "No") %>%
  #select(yard_line, game_minutes_remaining, ydstogo, score_differential, play_type, recommendation)

#example <- confusionMatrix(data=predicted_value, reference = expected_value)
#example
ggplot(pred_data, aes(x = accuracy)) +
  geom_density()
```



```
pred_data %>%
  filter(yard_line>0,
         year > 2020) %>%
  ggplot(aes(x=yard_line, fill = play_type)) + geom_density(alpha=0.7)
```

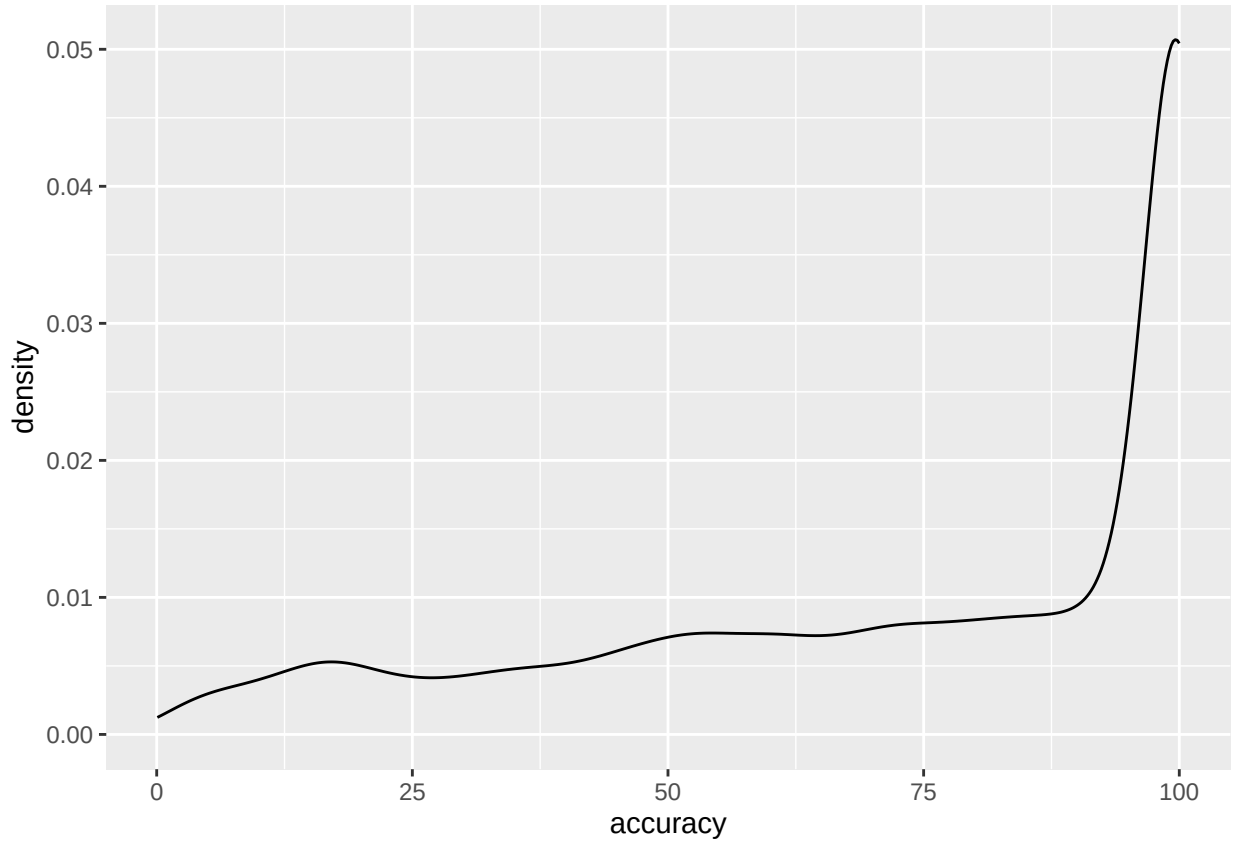



```
pred_data %>%
  filter(yard_line>0) %>%
  ggplot(aes(x=yard_line, fill = recommendation)) + geom_histogram(binwidth = 1, alpha=.4, position="id
```




```
crucial_data <- data_main %>%
  select(-c(down, fourth_down_converted, fourth_down_failed, ep, epa, year,
            week, posteam, play_type, posteam_type)) %>%
  select(wpa, everything()) %>%
  filter((yardline_100 < 51 | ydstogo < 5) & (abs(score_differential) < 14.5) & (wp > 0.1))

data_x <- data.matrix(crucial_data[, -1])
data_y <- crucial_data$wpa
xgb_matrix = xgb.DMatrix(data = data_x, label = data_y)

ewpa_punt = predict(punt_model_xgboost, xgb_matrix)
ewpa_kick = predict(kick_model_xgboost, xgb_matrix)
ewpa_go = predict(go_model_xgboost, xgb_matrix)

filtered_data <- data_main %>%
  filter((yardline_100 < 51 | ydstogo < 5) & (abs(score_differential) < 14.5) & (wp > 0.1))

crucial_data <- cbind(crucial_data, play_type = filtered_data$play_type,
          team = filtered_data$posteam, year = filtered_data$year, ewpa_punt, ewpa_kick, ewpa_go) %>%
  mutate(recommendation = case_when((ewpa_punt > ewpa_kick) & (ewpa_punt > ewpa_go) ~ "punt",
                                    (ewpa_kick > ewpa_punt) & (ewpa_kick > ewpa_go) ~ "field_goal",
                                    (ewpa_go > ewpa_kick) & (ewpa_go > ewpa_punt) ~ "go"),
         correct_call = ifelse(play_type == recommendation, "Yes", "No"),
         ewpa_max = pmax(ewpa_punt, ewpa_kick, ewpa_go),
         ewpa_play = case_when((play_type == "punt") ~ ewpa_punt,
                               (play_type == "field_goal") ~ ewpa_kick,
```

```
                                (play_type == "go") ~ ewpa_go),
         execution = wpa - ewpa_max)

crucial_data$ewpa_max_percentile <- ecdf(crucial_data$ewpa_max)(crucial_data$ewpa_max)
crucial_data$ewpa_play_percentile <- ecdf(crucial_data$ewpa_max)(crucial_data$ewpa_play)
crucial_data$accuracy <- 100 * (1 - (crucial_data$ewpa_max_percentile - crucial_data$ewpa_play_percentil


ggplot(crucial_data, aes(x = accuracy)) +
  geom_density()
```



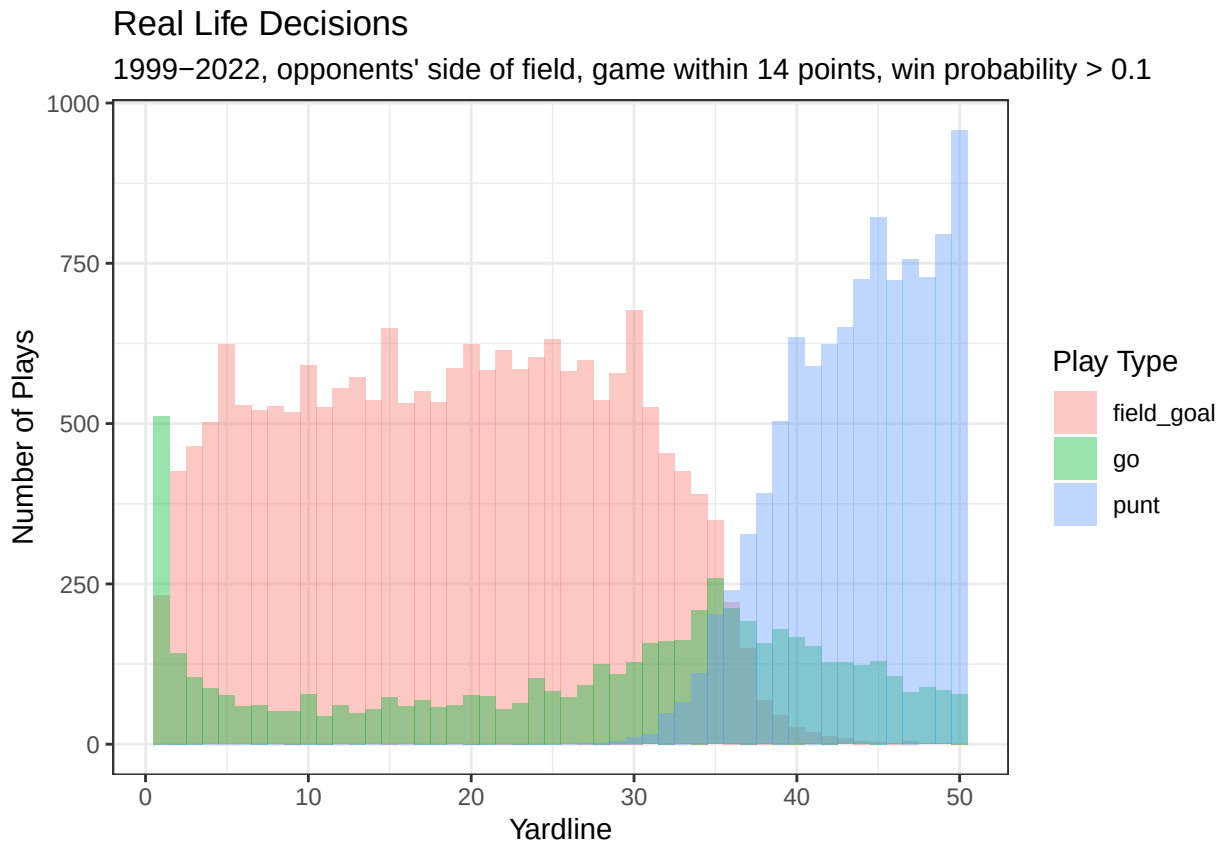

```
crucial_data %>%
  arrange(desc(year))
```

```
##                wpa yardline_100 game_seconds_remaining qtr ydstogo
##     1: -0.002238333           51                    3296   1       1
##     2: -0.019976735            6                    2899   1       3
##     3: -0.018679649           66                    2820   1       1
##     4: -0.095877826           27                    2455   2       4
##     5:  0.030603722           27                    1828   2       5
##    ---
## 46803:  0.028520644           11                    2369   2      10
## 46804: -0.044980898           10                    1819   2      10
## 46805: -0.073177233           29                    1800   3       5
## 46806:  0.060336888           57                     876   4       1
```

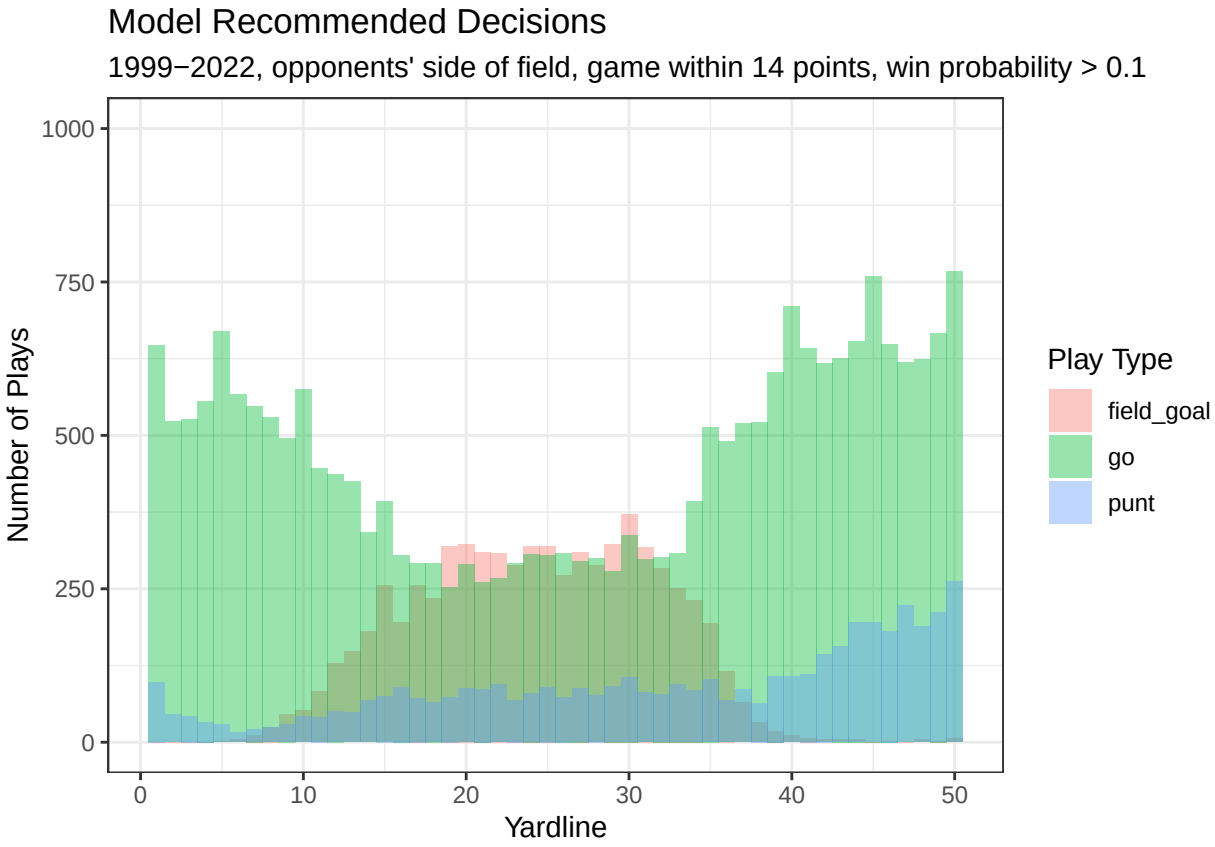
```
## 46807:  0.002091736          25                   377   4       7
##        score_differential no_score_prob opp_fg_prob opp_safety_prob
##     1:                  0   0.011813287 0.177440062    1.565513e-03
##     2:                  0   0.000751453 0.003594615    5.229356e-05
##     3:                 -3   0.046969537 0.196949929    3.269824e-03
##     4:                 -3   0.027033767 0.055807115    7.228962e-04
##     5:                -10   0.198925869 0.031771343    7.081460e-05
##    ---
## 46803:                  3   0.012635923 0.010552165    1.447756e-04
## 46804:                  6   0.058551728 0.001686924    1.439996e-05
## 46805:                 -9   0.002216538 0.084979071    1.469645e-03
## 46806:                -10   0.084862515 0.183874503    2.234099e-03
## 46807:                 -3   0.078442221 0.041841515    6.270266e-04
##         opp_td_prob   fg_prob  safety_prob      td_prob        wp
##     1: 0.3223033249 0.1836291 9.829091e-03 0.2934196293 0.4737365
##     2: 0.0052889088 0.9839867 4.810112e-05 0.0062778788 0.5815628
##     3: 0.2704848647 0.1542629 3.105036e-03 0.3249579072 0.4060214
##     4: 0.0902073976 0.7886681 3.824418e-04 0.0371782679 0.4443144
##     5: 0.0026942235 0.7643739 1.263318e-04 0.0020374833 0.2256564
##    ---
## 46803: 0.0191436318 0.9411797 3.276049e-04 0.0160162515 0.5929926
## 46804: 0.0005263558 0.9385480 1.056186e-04 0.0005670034 0.7959621
## 46805: 0.1227084289 0.7049640 9.205743e-04 0.0827417670 0.2314794
## 46806: 0.3207639158 0.1546367 6.023312e-03 0.2476048768 0.1292370
## 46807: 0.0705068382 0.7703825 5.302751e-04 0.0376695779 0.4555477
##        posteam_timeouts_remaining defteam_timeouts_remaining  play_type team
##     1:                          2                          3       punt  BAL
##     2:                          2                          3 field_goal  BAL
##     3:                          3                          2       punt  NYJ
##     4:                          3                          2 field_goal  NYJ
##     5:                          2                          0 field_goal  NYJ
##    ---
## 46803:                          3                          3 field_goal   LA
## 46804:                          1                          3 field_goal   LA
## 46805:                          3                          3 field_goal  TEN
## 46806:                          2                          3         go  TEN
## 46807:                          1                          3 field_goal  TEN
##        year    ewpa_punt     ewpa_kick      ewpa_go recommendation correct_call
##     1: 2022 -0.006766066 -0.0462302752  0.036177561             go           No
##     2: 2022 -0.003692099 -0.0249907244  0.132701740             go           No
##     3: 2022 -0.016130652 -0.0469167158  0.022320151             go           No
##     4: 2022 -0.042633358 -0.0612212941  0.020426100             go           No
##     5: 2022  0.009661306  0.0225406438  0.158307135             go           No
##    ---
## 46803: 1999 -0.014442386  0.0231197570  0.108514041             go           No
## 46804: 1999 -0.010340491 -0.0057053478  0.099041216             go           No
## 46805: 1999 -0.012865994 -0.0595337227  0.007767885             go           No
## 46806: 1999 -0.003388926 -0.0826969445  0.026677499             go          Yes
## 46807: 1999 -0.079701655 -0.0002697437 -0.020439010     field_goal          Yes
##            ewpa_max     ewpa_play    execution ewpa_max_percentile
##     1:  0.0361775607 -0.0067660660 -0.038415894           0.7226911
##     2:  0.1327017397 -0.0249907244 -0.152678475           0.8926870
##     3:  0.0223201513 -0.0161306523 -0.040999800           0.5309035
##     4:  0.0204261001 -0.0612212941 -0.116303926           0.4971906
```

```
##     5:  0.1583071351  0.0225406438 -0.127703413           0.9269767
##    ---
## 46803:  0.1085140407  0.0231197570 -0.079993397           0.8544448
## 46804:  0.0990412161 -0.0057053478 -0.144022115           0.8418185
## 46805:  0.0077678850 -0.0595337227 -0.080945118           0.2660286
## 46806:  0.0266774986  0.0266774986  0.033659389           0.6137330
## 46807: -0.0002697437 -0.0002697437  0.002361479           0.1477557
##        ewpa_play_percentile  accuracy
##     1:          0.078428440  35.57374
##     2:          0.013929540  12.12425
##     3:          0.030914180  50.00107
##     4:          0.001837332  50.46467
##     5:          0.534813169  60.78364
##    ---
## 46803:          0.545409875  69.09650
## 46804:          0.087743286  24.59248
## 46805:          0.001986882  73.59583
## 46806:          0.613732989 100.00000
## 46807:          0.147755678 100.00000
```

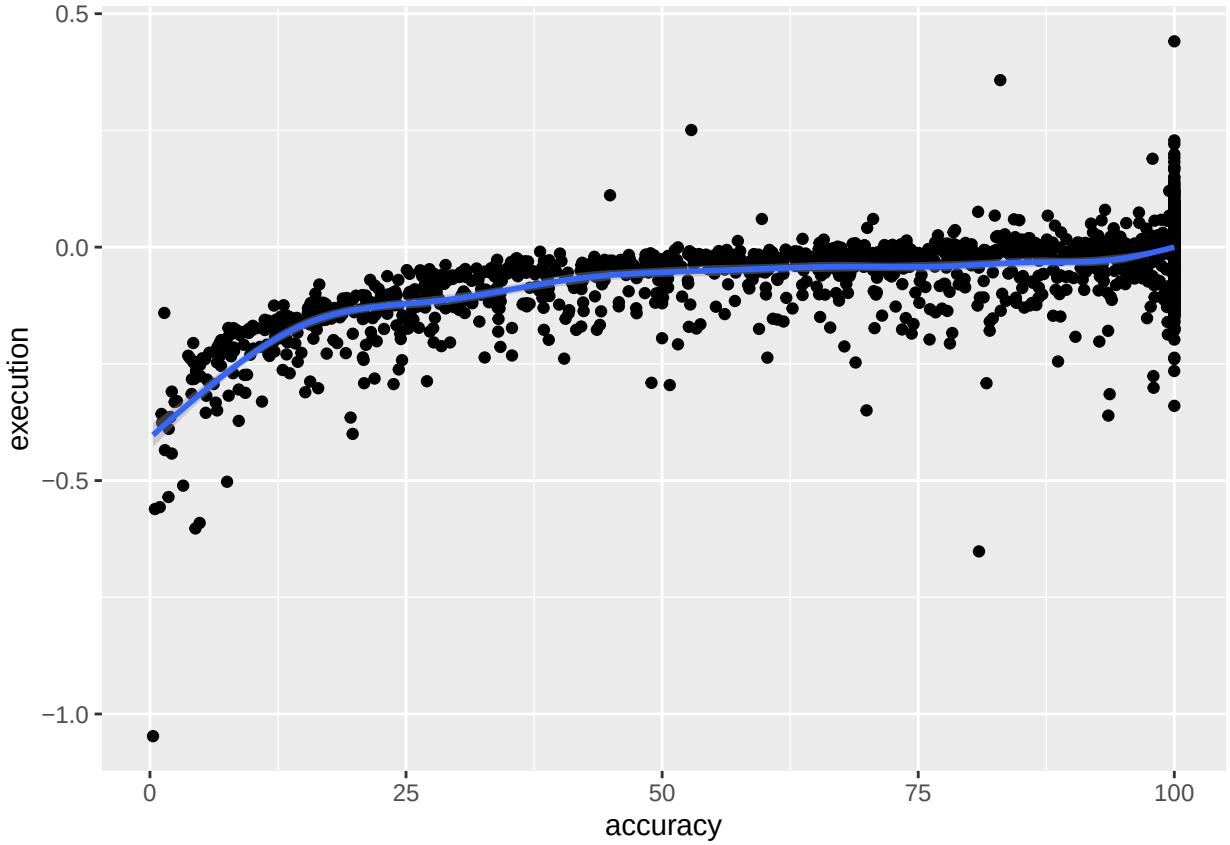
```
top_decisions <- crucial_data %>%
  arrange(desc(execution)) %>%
  select(ewpa_punt, ewpa_kick, ewpa_go, recommendation, play_type, wpa, execution, accuracy) %>%
  head(10)

crucial_data %>%
  filter(yardline_100<51) %>%
  ggplot(aes(x=yardline_100, fill = play_type)) + geom_histogram(binwidth = 1, alpha=.4, position="iden
  theme_bw()+
  labs(title = "Real Life Decisions", subtitle = "1999-2022, opponents' side of field, game within 14 p
```




```
crucial_data %>%
  filter(yardline_100<51) %>%
  ggplot(aes(x=yardline_100, fill = recommendation)) + geom_histogram(binwidth = 1, alpha=.4, position=
  theme_bw()+
  scale_y_continuous(limits = c(0,1000))+
  labs(title = "Model Recommended Decisions", subtitle = "1999-2022, opponents' side of field, game wit
```




```
ggplot(crucial_data[year == 2021], aes(x = accuracy, y = execution)) +
  geom_point() +
  geom_smooth()
```




```
top_decisions %>%
  kable(digits = 3)
```

ewpa_punt	ewpa_kick	ewpa_go	recommendation	play_type	wpa	execution	accuracy
-0.142	-0.091	0.014	go	go	0.532	0.518	100.000
0.013	-0.080	-0.003	punt	punt	0.529	0.516	100.000
-0.111	-0.258	0.002	go	go	0.481	0.479	100.000
-0.024	-0.194	-0.115	punt	go	0.450	0.474	98.494
-0.024	-0.312	0.115	go	go	0.586	0.471	100.000
0.071	-0.093	0.010	punt	go	0.521	0.450	48.681
-0.082	-0.247	0.033	go	go	0.474	0.441	100.000
-0.005	-0.336	-0.142	punt	go	0.435	0.441	90.826
-0.126	-0.145	0.061	go	go	0.474	0.413	100.000
-0.069	-0.361	0.012	go	go	0.418	0.406	100.000

Discussion and Conclusion