xgboostsetup

2025-02-17

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## Warning: package 'lubridate' was built under R version 4.4.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(nflfastR)
```

```
## Warning: package 'nflfastR' was built under R version 4.4.2
```

```
data <- load_pbp(2019:2024)
#data$play_type
```

FILTER DATA

```r
pbp <- data %>% filter(down==4,play_type!="no_play",play_type!="qb_kneel") %>%
  mutate(
    # Assign play types based on success & EPA:
    play_type_new = case_when(
      play_type == "run" | play_type == "pass" ~ "go_for_it",  # Converted on 4th down or made FG
      play_type == "punt" ~ "punt",       # Failed conversion or bad punt
      play_type == "field_goal" ~ "field_goal", # Rough estimate for FG makes
      TRUE ~ NA_character_
    ))
real_props <- pbp$play_type_new %>% table() %>% proportions()
real_props
```

```
## .
## field_goal  go_for_it       punt
##  0.2504436  0.1923532  0.5572032
```

Overall teams from 2019-2024 go for it about 19% of the time, Punt 56% and Attempt a Field Goal 25% on all 4th downs.

```r
#### PUNT WITHIN 10 DATA #####

offensive_plays <- c("run", "pass", "qb_spike", "qb_kneel")

punts <- pbp %>%
  filter(play_type == "punt") %>%
  select(game_id, play_id, posteam, yardline_100, season)

offensive_next_plays <- pbp %>%
  filter(play_type %in% offensive_plays) %>%
  select(game_id, play_id, posteam, yardline_100)

joined <- punts %>%
  left_join(offensive_next_plays, by = "game_id", suffix = c("_punt", "_next")) %>%
  filter(play_id_next > play_id_punt, posteam_next != posteam_punt) %>%
  group_by(game_id, play_id_punt) %>%
  slice_min(play_id_next) %>%
  ungroup()
```

```
## Warning in left_join(., offensive_next_plays, by = "game_id", suffix = c("_punt", : Detected an unexp
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
total_punts <- nrow(joined)
inside_10_count <- sum(joined$yardline_100_next >= 90, na.rm = TRUE)

joined <- joined %>%
  mutate(FieldPositionGroup = cut(
    yardline_100_punt,
    breaks = seq(30, 80),
```

```
    include.lowest = TRUE
  )) %>%
  filter(!is.na(FieldPositionGroup))

team_season_punt_stats <- joined %>%
  group_by(season, posteam_punt) %>%
  summarize(
    total_punts      = n(),
    inside_10_count  = sum(yardline_100_next >= 90, na.rm = TRUE),
    inside_10_pct    = round(100 * inside_10_count / total_punts, 1)
  ) %>%
  arrange(season, desc(-inside_10_pct))
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
print(team_season_punt_stats)
```

```
## # A tibble: 192 x 5
## # Groups:   season [6]
##    season posteam_punt total_punts inside_10_count inside_10_pct
##     <int> <chr>              <int>           <int>         <dbl>
##  1   2019 ATL                   15               0             0
##  2   2019 BAL                   21               0             0
##  3   2019 BUF                   31               0             0
##  4   2019 CAR                   22               0             0
##  5   2019 CHI                   32               0             0
##  6   2019 CIN                   17               0             0
##  7   2019 CLE                   23               0             0
##  8   2019 DAL                   25               0             0
##  9   2019 DEN                   26               0             0
## 10   2019 DET                   30               0             0
## # i 182 more rows
```

```
############ AGGRESSION ############

fourth_down_plays <- pbp %>%
  filter(
    down == 4,
    ydstogo <= 5,
    penalty == 0,
    !is.na(epa),
    !is.na(play_type),
    !play_type %in% c("no_play", "qb_kneel", "qb_spike")
  )

fourth_down_plays <- fourth_down_plays %>%
  mutate(go_for_it = as.integer(play_type %in% c("run", "pass")))

coach_aggressiveness <- fourth_down_plays %>%
  group_by(posteam, season) %>%
  summarise(
```

```
    total_4th_downs = n(),
    go_for_it_attempts = sum(go_for_it),
    coach_aggressiveness = mean(go_for_it)
  ) %>%
  arrange(season, desc(-coach_aggressiveness))
```

```
## `summarise()` has grouped output by 'posteam'. You can override using the
## `.groups` argument.
```

```
print(coach_aggressiveness)
```

```
## # A tibble: 192 x 5
## # Groups:   posteam [32]
##    posteam season total_4th_downs go_for_it_attempts coach_aggressiveness
##    <chr>    <int>           <int>              <int>                <dbl>
##  1 CHI       2019              44                  7                0.159
##  2 PIT       2019              43                  7                0.163
##  3 NE        2019              55                 10                0.182
##  4 DEN       2019              48                  9                0.188
##  5 DET       2019              42                  8                0.190
##  6 TEN       2019              41                  9                0.220
##  7 SF        2019              53                 12                0.226
##  8 NO        2019              44                 10                0.227
##  9 WAS       2019              44                 10                0.227
## 10 LA        2019              35                  8                0.229
## # i 182 more rows
```

```
######## FIELD GOAL PCT ######### (Excluding attempts within the 20 yardline)


field_goals <- pbp %>%
  filter(play_type == "field_goal", yardline_100 > 20)


team_fg_by_season <- field_goals %>%
  group_by(posteam, season) %>%
  summarize(
    total_fg_att  = n(),
    total_fg_made = sum(field_goal_result == "made", na.rm = TRUE),
    fg_percentage = round(total_fg_made / total_fg_att * 100, 1)
  ) %>%
  arrange(season, desc(-fg_percentage))
```

```
## `summarise()` has grouped output by 'posteam'. You can override using the
## `.groups` argument.
```

```
print(team_fg_by_season)
```

```
## # A tibble: 192 x 5
## # Groups:   posteam [32]
##    posteam season total_fg_att total_fg_made fg_percentage
```

```
##    <chr>    <int>        <int>         <int>         <dbl>
##  1 NYG       2019            5             1          20
##  2 TEN       2019           13             6          46.2
##  3 IND       2019           14             7          50
##  4 LV        2019           12             6          50
##  5 LA        2019           17             9          52.9
##  6 NYJ       2019           17             9          52.9
##  7 CHI       2019            7             4          57.1
##  8 DAL       2019           19            11          57.9
##  9 LAC       2019           20            12          60
## 10 NE        2019           13             8          61.5
## # i 182 more rows
```

```
######## OFFENSIVE YARDAGE #########

yardage <- pbp %>%
  filter(
    !is.na(yards_gained),
    !is.na(result),
    play_type %in% c("run", "pass"))

total_yardage_by_team_season <- yardage %>%
  group_by(season, posteam) %>%
  summarise(
    total_yardage = sum(yards_gained)
  ) %>%
  ungroup()%>%
  arrange(season, -total_yardage)
```

```
## `summarise()` has grouped output by 'season'. You can override using the
## `.groups` argument.
```

```
print(total_yardage_by_team_season, n = 32)
```

```
## # A tibble: 192 x 3
##    season posteam total_yardage
##     <int> <chr>           <dbl>
##  1   2019 BAL               211
##  2   2019 CIN               211
##  3   2019 MIA               176
##  4   2019 NYG               151
##  5   2019 ATL               137
##  6   2019 PHI               110
##  7   2019 IND               108
##  8   2019 LAC               107
##  9   2019 MIN               106
## 10   2019 SEA               104
## 11   2019 ARI                96
## 12   2019 LV                 95
## 13   2019 GB                 94
## 14   2019 DEN                88
## 15   2019 CHI                84
## 16   2019 HOU                83
```

```
## 17   2019 SF                83
## 18   2019 CLE               75
## 19   2019 LA                71
## 20   2019 JAX               60
## 21   2019 CAR               51
## 22   2019 WAS               51
## 23   2019 DAL               45
## 24   2019 NYJ               44
## 25   2019 KC                43
## 26   2019 NE                43
## 27   2019 TB                31
## 28   2019 TEN               27
## 29   2019 NO                19
## 30   2019 DET               18
## 31   2019 BUF                4
## 32   2019 PIT                0
## # i 160 more rows
```

```
##### TEAM RANKING AND STATS FOR EACH CATEGORY EACH SEASON ######


team_season_punt_stats <- team_season_punt_stats %>%
  group_by(season) %>%
  mutate(inside_10_pct_rank = dense_rank(desc(inside_10_pct))) %>%
  ungroup()

coach_aggressiveness <- coach_aggressiveness %>%
  group_by(season) %>%
  mutate(coach_aggressiveness_rank = dense_rank(desc(coach_aggressiveness))) %>%
  ungroup()

team_fg_by_season <- team_fg_by_season %>%
  group_by(season) %>%
  mutate(fg_percentage_rank = dense_rank(desc(fg_percentage))) %>%
  ungroup()

total_yardage_by_team_season <- total_yardage_by_team_season %>%
  group_by(season) %>%
  mutate(total_yardage_rank = dense_rank(desc(total_yardage))) %>%
  ungroup()

team_season_punt_stats <- team_season_punt_stats %>%
  rename(posteam = posteam_punt)

final_stats <- team_season_punt_stats %>%
  left_join(coach_aggressiveness,      by = c("season", "posteam")) %>%
  left_join(team_fg_by_season,        by = c("season", "posteam")) %>%
  left_join(total_yardage_by_team_season, by = c("season", "posteam"))

print(final_stats, n = 32)
```

```
## # A tibble: 192 x 16
##    season posteam total_punts inside_10_count inside_10_pct inside_10_pct_rank
##     <int> <chr>         <int>           <int>         <dbl>              <int>
```

```
##  1   2019 ATL              15                0           0                    4
##  2   2019 BAL              21                0           0                    4
##  3   2019 BUF              31                0           0                    4
##  4   2019 CAR              22                0           0                    4
##  5   2019 CHI              32                0           0                    4
##  6   2019 CIN              17                0           0                    4
##  7   2019 CLE              23                0           0                    4
##  8   2019 DAL              25                0           0                    4
##  9   2019 DEN              26                0           0                    4
## 10   2019 DET              30                0           0                    4
## 11   2019 GB               37                0           0                    4
## 12   2019 HOU              28                0           0                    4
## 13   2019 JAX              26                0           0                    4
## 14   2019 KC               45                0           0                    4
## 15   2019 LAC              19                0           0                    4
## 16   2019 LV               23                0           0                    4
## 17   2019 MIA              15                0           0                    4
## 18   2019 MIN              44                0           0                    4
## 19   2019 NE               47                0           0                    4
## 20   2019 NO               37                0           0                    4
## 21   2019 NYG              18                0           0                    4
## 22   2019 NYJ              19                0           0                    4
## 23   2019 PHI              27                0           0                    4
## 24   2019 PIT              28                0           0                    4
## 25   2019 SEA              32                0           0                    4
## 26   2019 SF               28                0           0                    4
## 27   2019 TB               40                0           0                    4
## 28   2019 TEN              51                0           0                    4
## 29   2019 WAS              19                0           0                    4
## 30   2019 LA               33                2           6.1                  3
## 31   2019 IND              25                3          12                    2
## 32   2019 ARI              37                6          16.2                  1
## # i 160 more rows
## # i 10 more variables: total_4th_downs <int>, go_for_it_attempts <int>,
## #   coach_aggressiveness <dbl>, coach_aggressiveness_rank <int>,
## #   total_fg_att <int>, total_fg_made <int>, fg_percentage <dbl>,
## #   fg_percentage_rank <int>, total_yardage <dbl>, total_yardage_rank <int>
```

```r
###### JUST TEAM RANKINGS #######

final_stats_rank <- team_season_punt_stats %>%
  left_join(coach_aggressiveness, by = c("season", "posteam")) %>%
  left_join(team_fg_by_season,    by = c("season", "posteam")) %>%
  left_join(total_yardage_by_team_season, by = c("season", "posteam")) %>%
  select(
    season,
    posteam,
    inside_10_pct_rank,
    coach_aggressiveness_rank,
    fg_percentage_rank,
    total_yardage_rank
  )

print(final_stats_rank, n = 150)
```

```
## # A tibble: 192 x 6
##     season posteam inside_10_pct_rank coach_aggressiveness_~1 fg_percentage_rank
##      <int> <chr>                <int>                   <int>               <int>
##   1   2019 ATL                      4                       2                  15
##   2   2019 BAL                      4                       1                   1
##   3   2019 BUF                      4                      14                   5
##   4   2019 CAR                      4                       3                  10
##   5   2019 CHI                      4                      28                  22
##   6   2019 CIN                      4                      11                  11
##   7   2019 CLE                      4                      15                  13
##   8   2019 DAL                      4                      19                  21
##   9   2019 DEN                      4                      25                  10
##  10   2019 DET                      4                      24                  14
##  11   2019 GB                       4                      16                   7
##  12   2019 HOU                      4                       2                  16
##  13   2019 JAX                      4                       7                   2
##  14   2019 KC                       4                      17                   3
##  15   2019 LAC                      4                      10                  20
##  16   2019 LV                       4                       6                  24
##  17   2019 MIA                      4                       4                  16
##  18   2019 MIN                      4                       8                   4
##  19   2019 NE                       4                      26                  19
##  20   2019 NO                       4                      21                   6
##  21   2019 NYG                      4                       5                  26
##  22   2019 NYJ                      4                      12                  23
##  23   2019 PHI                      4                       9                  17
##  24   2019 PIT                      4                      27                   5
##  25   2019 SEA                      4                      18                  19
##  26   2019 SF                       4                      22                  18
##  27   2019 TB                       4                      13                   9
##  28   2019 TEN                      4                      23                  25
##  29   2019 WAS                      4                      21                   8
##  30   2019 LA                       3                      20                  23
##  31   2019 IND                      2                       3                  24
##  32   2019 ARI                      1                      10                  12
##  33   2020 ARI                      2                       3                  16
##  34   2020 ATL                      2                       5                   5
##  35   2020 BAL                      2                      16                  10
##  36   2020 BUF                      2                      25                  11
##  37   2020 CAR                      2                       9                  21
##  38   2020 CHI                      2                      18                   8
##  39   2020 CIN                      2                      10                  21
##  40   2020 CLE                      2                       2                  12
##  41   2020 DAL                      2                       6                  20
##  42   2020 DEN                      2                      28                   9
##  43   2020 DET                      2                      12                  22
##  44   2020 GB                       2                       1                   1
##  45   2020 HOU                      2                      19                  18
##  46   2020 IND                      2                       8                   7
##  47   2020 JAX                      2                      14                  17
##  48   2020 KC                       2                      14                   2
##  49   2020 LA                       2                      22                  14
##  50   2020 LAC                      2                      11                  23
##  51   2020 LV                       2                       7                  11
```

```
##  52   2020 MIA                2                29                4
##  53   2020 MIN                2                10               23
##  54   2020 NE                 2                21                6
##  55   2020 NO                 2                24               22
##  56   2020 NYG                2                15                1
##  57   2020 NYJ                2                20               15
##  58   2020 PHI                2                 4               13
##  59   2020 PIT                2                17                3
##  60   2020 SEA                2                26                1
##  61   2020 SF                 2                23                7
##  62   2020 TEN                2                13               19
##  63   2020 WAS                2                24               19
##  64   2020 TB                 1                27               12
##  65   2021 ARI                7                 7               17
##  66   2021 ATL                7                15                5
##  67   2021 BAL                7                20                2
##  68   2021 BUF                7                11               18
##  69   2021 CAR                7                24                7
##  70   2021 CHI                7                 3               11
##  71   2021 CIN                7                13               10
##  72   2021 CLE                7                 6               19
##  73   2021 DAL                7                 9               17
##  74   2021 DEN                7                16               12
##  75   2021 DET                7                 1                9
##  76   2021 GB                 7                 5               15
##  77   2021 HOU                7                25               19
##  78   2021 JAX                7                21               20
##  79   2021 KC                 7                17                6
##  80   2021 LA                 7                29                7
##  81   2021 LV                 7                27                3
##  82   2021 NE                 7                23                2
##  83   2021 NYG                7                22                4
##  84   2021 NYJ                7                 8               22
##  85   2021 PHI                7                14                7
##  86   2021 PIT                7                30                1
##  87   2021 SF                 7                26                8
##  88   2021 TB                 7                28               13
##  89   2021 TEN                7                12               20
##  90   2021 WAS                7                10               14
##  91   2021 SEA                6                31               21
##  92   2021 IND                5                 2                7
##  93   2021 LAC                4                 4               12
##  94   2021 MIN                3                19               11
##  95   2021 MIA                2                20               16
##  96   2021 NO                 1                18               17
##  97   2022 ATL                5                 9                8
##  98   2022 BAL                5                17                9
##  99   2022 BUF                5                 6                4
## 100   2022 CAR                5                24                1
## 101   2022 CHI                5                15                6
## 102   2022 CIN                5                29               11
## 103   2022 CLE                5                 2               20
## 104   2022 DAL                5                12                3
## 105   2022 DEN                5                22                5
```

```
## 106   2022 DET                     5                      5                 16
## 107   2022 HOU                     5                     18                  3
## 108   2022 IND                     5                      8                 14
## 109   2022 JAX                     5                     10                  9
## 110   2022 LA                      5                     11                  7
## 111   2022 LAC                     5                     19                 15
## 112   2022 LV                      5                     16                  4
## 113   2022 MIA                     5                      7                 10
## 114   2022 MIN                     5                     20                 21
## 115   2022 NE                      5                     25                 17
## 116   2022 NO                      5                     30                 24
## 117   2022 NYG                     5                     23                  2
## 118   2022 NYJ                     5                     21                 18
## 119   2022 PHI                     5                      3                  2
## 120   2022 PIT                     5                     31                 22
## 121   2022 SEA                     5                     11                  6
## 122   2022 TB                      5                     14                 19
## 123   2022 TEN                     5                     28                 11
## 124   2022 WAS                     5                     13                 13
## 125   2022 ARI                     4                      1                 12
## 126   2022 KC                      3                     27                 25
## 127   2022 SF                      2                     26                  9
## 128   2022 GB                      1                      4                 23
## 129   2023 ARI                     2                      6                  9
## 130   2023 ATL                     2                     18                 15
## 131   2023 BAL                     2                     26                 12
## 132   2023 BUF                     2                      9                 20
## 133   2023 CAR                     2                      3                 17
## 134   2023 CHI                     2                     11                  7
## 135   2023 CIN                     2                     21                 13
## 136   2023 CLE                     2                     12                  8
## 137   2023 DAL                     2                      4                  1
## 138   2023 DEN                     2                     27                  6
## 139   2023 DET                     2                      1                  7
## 140   2023 GB                      2                      7                 25
## 141   2023 HOU                     2                     23                 21
## 142   2023 IND                     2                     16                 13
## 143   2023 JAX                     2                     20                 18
## 144   2023 LA                      2                     15                 23
## 145   2023 LAC                     2                     17                  6
## 146   2023 LV                      2                     29                 11
## 147   2023 MIA                     2                      5                 16
## 148   2023 MIN                     2                     13                 24
## 149   2023 NE                      2                     28                 26
## 150   2023 NO                      2                     22                 13
## # i 42 more rows
## # i abbreviated name: 1: coach_aggressiveness_rank
## # i 1 more variable: total_yardage_rank <int>
```

```
#names(team_season_punt_stats)
pbp <- pbp %>% left_join(final_stats_rank, by = c("season", "posteam"))
#pbp %>% select(posteam,inside_10_pct_rank, coach_aggressiveness_rank, fg_percentage_rank, total_yardag
```

```r
pbp <- pbp %>%
  mutate(
    across(c(inside_10_pct_rank, coach_aggressiveness_rank, fg_percentage_rank, total_yardage_rank), as
  )
```

Create Expected EPA columns

```r
##### Group by the Play type and yardline on the field to average epa so that we can group similar play
pbp <- pbp %>%
  mutate(
    yardline_bin = cut(yardline_100,
                       breaks = seq(0, 100, by = 10),
                       labels = seq(10, 100, by = 10),
                       include.lowest = TRUE)
  ) %>%
  group_by(play_type, yardline_bin) %>%
  mutate(
    expected_epa = mean(epa, na.rm = TRUE) # Average EPA for similar plays in each yardline bin
  ) %>%
  ungroup() %>%
  mutate(
    punt_expected_epa = ifelse(play_type_new == "punt", expected_epa, NA),
    field_goal_expected_epa = ifelse(play_type_new == "field_goal", expected_epa, NA),
    go_for_it_expected_epa = ifelse(play_type_new == "go_for_it", expected_epa, NA)
  )
pbp$yardline_bin %>% table() %>% proportions()
```

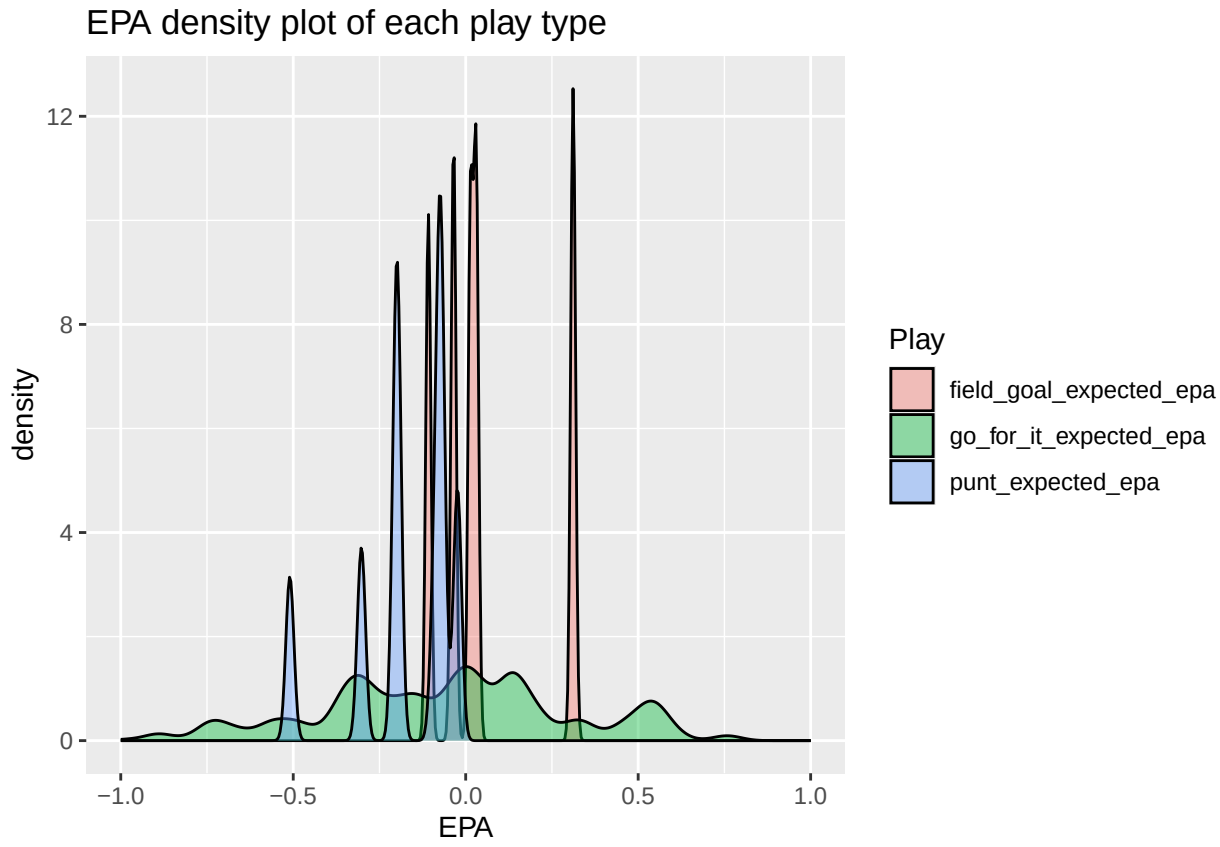
```
## .
##         10         20         30         40         50         60         70
## 0.08217153 0.08905788 0.09598648 0.10097169 0.10920997 0.13008027 0.16442755
##         80         90        100
## 0.14224757 0.06683566 0.01901141
```

The number of 4th down plays we have is scattered more evenly than I thought across all the different yardlines. But, the most are between 40 and 80 yards to go (Accounting for nearly 55% of the total 4th downs)

```r
pbp <- pbp %>%
  group_by(yardline_100) %>%
  mutate(
    punt_expected_epa = ifelse(is.na(punt_expected_epa), mean(punt_expected_epa, na.rm = TRUE), punt_exp
    field_goal_expected_epa = ifelse(is.na(field_goal_expected_epa), mean(field_goal_expected_epa, na.rm
    go_for_it_expected_epa = ifelse(is.na(go_for_it_expected_epa), mean(go_for_it_expected_epa, na.rm =
  ) %>%
  ungroup()
pbp$go_for_it_expected_epa <- ifelse(is.na(pbp$go_for_it_expected_epa), mean(pbp$go_for_it_expected_epa
pbp$field_goal_expected_epa[is.na(pbp$field_goal_expected_epa)] = -10
for (i in 1:length(pbp$play_id)){
  pbp$punt_expected_epa[i] <- ifelse(is.na(pbp$punt_expected_epa[i]), mean(pbp$punt_expected_epa, na.rm
}
pbp_long <- pbp %>%
```

```
  pivot_longer(cols=c("punt_expected_epa","field_goal_expected_epa","go_for_it_expected_epa"), names_to
ggplot(pbp_long,aes(x=EPA,fill=Play)) + geom_density(alpha=0.4,color="black") + xlim(-1,1) + labs(title=
```

```
## Warning: Removed 14428 rows containing non-finite outside the scale range
## (`stat_density()`).
```




```
# Create Best Decision
pbp <- pbp %>%
  mutate(
    best_decision = case_when(
      go_for_it_expected_epa == pmax(field_goal_expected_epa, punt_expected_epa,go_for_it_expected_epa)
      field_goal_expected_epa == pmax(go_for_it_expected_epa, punt_expected_epa, field_goal_expected_epa
      punt_expected_epa == pmax(go_for_it_expected_epa, field_goal_expected_epa, punt_expected_epa) ~ "
    ) %>%
    as.factor()
  )
epa_props <- pbp$best_decision %>% table() %>% proportions()
epa_props
```

```
## .
## field_goal  go_for_it       punt
##  0.3211660  0.4188002  0.2600338
```

Here we have our best_decision coming from 0.42 for go_for_it and 0.321 for field_goal only 26% of time punting.

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Create Machine Learning Model
features <- pbp[,c("score_differential_post","yardline_100","wp","ydstogo","inside_10_pct_rank", "coach

# Convert to matrix for XGBoost
X <- as.matrix(features)
y <- as.factor(pbp$best_decision)

# Split data (80% train, 20% test)
set.seed(1234)
train_idx <- sample(length(pbp)*0.8,replace=FALSE)
X_train <- X[train_idx, ]
y_train <- y[train_idx]
X_test <- X[-train_idx, ]
y_test <- y[-train_idx]
as.numeric(y_train) - 1 %>% table() %>% proportions()
```

```
## Warning in as.numeric(y_train) - 1 %>% table() %>% proportions(): Recycling array of length 1 in vec
##   Use c() or as.vector() instead.

##   [1] 1 2 2 2 2 1 2 2 0 2 0 2 1 0 0 2 2 2 0 2 0 2 1 2 1 0 2 0 2 0 1 0 0 1 1 0 1
##  [38] 1 2 2 2 2 2 1 1 2 1 0 1 0 2 1 0 2 0 0 1 1 1 1 1 0 1 1 0 0 2 1 1 0 1 1 2 0
##  [75] 1 1 1 1 0 1 1 1 2 0 1 0 2 1 2 1 1 0 1 0 0 2 0 0 1 1 1 0 0 2 0 0 1 1 1 1 0
## [112] 1 2 2 0 1 2 1 2 2 2 2 2 1 2 1 2 1 2 2 2 0 2 1 2 0 1 1 2 1 0 2 1 1 2 0 2 0
## [149] 0 1 1 2 0 1 0 1 2 0 1 0 0 1 0 1 0 0 1 1 0 1 1 2 1 0 2 2 1 1 1 1 2 1 1 1 2
## [186] 1 0 0 2 1 2 2 1 1 0 0 1 1 1 1 2 1 1 1 1 1 2 1 1 1 0 1 0 1 2 0 0 1 2 1 1 0
## [223] 2 1 2 0 0 1 0 0 2 2 1 1 1 1 1 0 2 2 1 2 1 1 1 0 1 2 1 1 1 2 2 2 2 1 1 0 0
## [260] 1 1 2 1 1 1 1 0 2 1 1 2 1 2 0 2 0 1 2 0 1 1 1 1 0 1 0 0 0 1 2 1 1 2 1 0 2
## [297] 2 0 0 2 2 0 1 1 0 2
```

```r
# Train XGBoost model
xgb_model <- xgboost(
  data = X_train, label = as.numeric(y_train) - 1, # Convert to numeric
  nrounds = 100, objective = "multi:softprob",
  num_class = 3, eval_metric = "mlogloss"
)
```

```
## [1]  train-mlogloss:0.735776
## [2]  train-mlogloss:0.524639
## [3]  train-mlogloss:0.383447
## [4]  train-mlogloss:0.287132
## [5]  train-mlogloss:0.220558
## [6]  train-mlogloss:0.170709
## [7]  train-mlogloss:0.134367
## [8]  train-mlogloss:0.107981
## [9]  train-mlogloss:0.088099
## [10] train-mlogloss:0.072679
## [11] train-mlogloss:0.060157
## [12] train-mlogloss:0.050831
## [13] train-mlogloss:0.043411
## [14] train-mlogloss:0.038106
## [15] train-mlogloss:0.033494
## [16] train-mlogloss:0.029902
## [17] train-mlogloss:0.026689
## [18] train-mlogloss:0.024469
## [19] train-mlogloss:0.022553
## [20] train-mlogloss:0.020936
## [21] train-mlogloss:0.019577
## [22] train-mlogloss:0.018491
## [23] train-mlogloss:0.017585
## [24] train-mlogloss:0.016848
## [25] train-mlogloss:0.016230
## [26] train-mlogloss:0.015671
## [27] train-mlogloss:0.015257
## [28] train-mlogloss:0.014873
## [29] train-mlogloss:0.014562
## [30] train-mlogloss:0.014219
## [31] train-mlogloss:0.013973
## [32] train-mlogloss:0.013661
## [33] train-mlogloss:0.013396
## [34] train-mlogloss:0.013177
## [35] train-mlogloss:0.012907
## [36] train-mlogloss:0.012715
## [37] train-mlogloss:0.012456
## [38] train-mlogloss:0.012242
## [39] train-mlogloss:0.012078
## [40] train-mlogloss:0.011885
## [41] train-mlogloss:0.011696
## [42] train-mlogloss:0.011564
## [43] train-mlogloss:0.011314
## [44] train-mlogloss:0.011199
## [45] train-mlogloss:0.011028
## [46] train-mlogloss:0.010911
```

```
## [47] train-mlogloss:0.010776
## [48] train-mlogloss:0.010674
## [49] train-mlogloss:0.010539
## [50] train-mlogloss:0.010439
## [51] train-mlogloss:0.010343
## [52] train-mlogloss:0.010266
## [53] train-mlogloss:0.010176
## [54] train-mlogloss:0.010077
## [55] train-mlogloss:0.009987
## [56] train-mlogloss:0.009895
## [57] train-mlogloss:0.009820
## [58] train-mlogloss:0.009750
## [59] train-mlogloss:0.009683
## [60] train-mlogloss:0.009613
## [61] train-mlogloss:0.009550
## [62] train-mlogloss:0.009490
## [63] train-mlogloss:0.009426
## [64] train-mlogloss:0.009347
## [65] train-mlogloss:0.009288
## [66] train-mlogloss:0.009233
## [67] train-mlogloss:0.009160
## [68] train-mlogloss:0.009104
## [69] train-mlogloss:0.009053
## [70] train-mlogloss:0.009004
## [71] train-mlogloss:0.008953
## [72] train-mlogloss:0.008893
## [73] train-mlogloss:0.008848
## [74] train-mlogloss:0.008793
## [75] train-mlogloss:0.008750
## [76] train-mlogloss:0.008708
## [77] train-mlogloss:0.008618
## [78] train-mlogloss:0.008579
## [79] train-mlogloss:0.008539
## [80] train-mlogloss:0.008504
## [81] train-mlogloss:0.008480
## [82] train-mlogloss:0.008446
## [83] train-mlogloss:0.008415
## [84] train-mlogloss:0.008393
## [85] train-mlogloss:0.008364
## [86] train-mlogloss:0.008330
## [87] train-mlogloss:0.008302
## [88] train-mlogloss:0.008240
## [89] train-mlogloss:0.008221
## [90] train-mlogloss:0.008189
## [91] train-mlogloss:0.008173
## [92] train-mlogloss:0.008147
## [93] train-mlogloss:0.008118
## [94] train-mlogloss:0.008095
## [95] train-mlogloss:0.008079
## [96] train-mlogloss:0.008054
## [97] train-mlogloss:0.008015
## [98] train-mlogloss:0.007995
## [99] train-mlogloss:0.007943
## [100]    train-mlogloss:0.007928
```

```r
#Step 5: Define Dynamic
#k dynamically by comparing predicted decision confidence:
# Get model predictions
pred_probs <- predict(xgb_model, X_test)
pred_probs <- matrix(pred_probs, ncol = 3, byrow = TRUE)
# Assign k based on decision confidence
k_values <- apply(pred_probs, 1, function(probs) {
  confidence <- max(probs) # Confidence of best choice
  return( (1 - confidence) * 10 ) # Higher confidence -> lower k
})
summary(k_values)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002093 0.016159 0.040694 0.314727 0.120156 5.015536
```

```r
pbp$k_dynamic <- NA
pbp$k_dynamic[-train_idx] <- k_values
#pbp %>% filter(k_dynamic > 0.8)
```

We train the xg boost model with our features in the X and y is our best decision variable. But for xgboost to work we create a numerical value system for best decision where, - field_goal = 0 - go_for_it = 1 - punt = 2

```r
epa_fg <- pbp %>% filter(field_goal_attempt == 1)
epa_fg <- epa_fg$field_goal_expected_epa
epa_punt <- pbp$punt_expected_epa
epa_go <- pbp$go_for_it_expected_epa
only_fg_epa <- pbp$field_goal_expected_epa[pbp$field_goal_expected_epa != -10]
pbp <- pbp %>%
  mutate(only_fg_expected_epa = (field_goal_expected_epa[field_goal_expected_epa == -10] = NA)) %>%
  mutate(
    go_for_it_adjusted_epa = go_for_it_expected_epa - k_dynamic * sd(go_for_it_expected_epa),
    field_goal_adjusted_epa = field_goal_expected_epa - k_dynamic * sd(only_fg_epa),
    punt_adjusted_epa = punt_expected_epa - k_dynamic * sd(punt_expected_epa),
    final_decision = case_when(
      go_for_it_adjusted_epa > pmax(field_goal_adjusted_epa, punt_adjusted_epa) ~ "go_for_it",
      field_goal_adjusted_epa > pmax(go_for_it_adjusted_epa, punt_adjusted_epa) ~ "field_goal",
      punt_adjusted_epa > pmax(go_for_it_adjusted_epa, field_goal_adjusted_epa) ~ "punt"
    )
)
vec <- c(sd(pbp$punt_expected_epa),
sd(only_fg_epa),
sd(pbp$go_for_it_expected_epa))
final_props <- pbp$final_decision %>% table() %>% proportions()
sd_df <- data.frame(Plays=c("Punt SD","FG SD","Go For It SD"),Standard_Deviations = vec)
sd_df
```

```
##          Plays Standard_Deviations
## 1      Punt SD           0.2523796
## 2        FG SD           0.1475529
## 3 Go For It SD           0.4544383
```

```
decision_props_df <- data.frame(Play = c("Field Goal","Go For It","Punt"),Coach_decision = as.numeric(r
decision_props_df
```

```
##         Play Coach_decision EPA_decision XGBoost_decision
## 1 Field Goal      0.2504436    0.3211660        0.3494693
## 2  Go For It      0.1923532    0.4188002        0.3538350
## 3       Punt      0.5572032    0.2600338        0.2966958
```

Here we can see that XG Boost and EPA both favor field goals more than they are used in reality. Also XG Boost tends to split the difference on going for it and punts.

Our XGBoost model claims that teams should be going for it on 4th down about 16.2% more often than they have from 2019 to 2024. Also it claims there should be a reduction in the number of punts, actually decreasing punts by a shocking 26.2%.