NFL Play by Play

Beveryn Tucker, Gilbert Guyah, Kyle Welch

5/1/2020

Introduction

```
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 3.6.3
```

```
## -- Attaching packages ------------------------------------ tidymodels 0.1.0 --
```

```
## v broom     0.5.6      v recipes   0.1.12
## v dials     0.0.6      v rsample   0.0.6
## v dplyr     0.8.5      v tibble    3.0.1
## v ggplot2   3.3.0      v tune      0.1.0
## v infer     0.5.1      v workflows 0.1.1
## v parsnip   0.1.1      v yardstick 0.0.6
## v purrr     0.3.4
```

```
## Warning: package 'broom' was built under R version 3.6.3
```

```
## Warning: package 'dials' was built under R version 3.6.3
```

```
## Warning: package 'scales' was built under R version 3.6.3
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
## Warning: package 'infer' was built under R version 3.6.3
```

```
## Warning: package 'parsnip' was built under R version 3.6.3
```

```
## Warning: package 'purrr' was built under R version 3.6.3
```

```
## Warning: package 'recipes' was built under R version 3.6.3
```

```
## Warning: package 'rsample' was built under R version 3.6.3
```

```
## Warning: package 'tibble' was built under R version 3.6.3
```

```
## Warning: package 'tune' was built under R version 3.6.3
```

```
## Warning: package 'workflows' was built under R version 3.6.3
```

```
## Warning: package 'yardstick' was built under R version 3.6.3
```

```
## -- Conflicts --------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x ggplot2::margin() masks dials::margin()
## x recipes::step()   masks stats::step()
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.0 --
```

```
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
## Warning: package 'stringr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::margin()   masks dials::margin()
## x readr::spec()       masks yardstick::spec()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity
```

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

```
library(ggplot2)
library(lattice)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
## The following object is masked from 'package:dials':
##
##     margin
```

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 3.6.3
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.3
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.6.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

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

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(dplyr)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.6.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 3.6.3
```

```
## Package 'mclust' version 5.4.6
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

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

```
library(cluster)
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.6.3
```

```
##
## Attaching package: 'ranger'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(rsample)
library(dendextend)
```

```
## Warning: package 'dendextend' was built under R version 3.6.3
```

```
##
## ---------------------
## Welcome to dendextend version 1.13.4
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issu
es
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:dials':
##
##     prune
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
NFL <- read_csv("F:/1. Syracuse/6.IST707_DATA_ANALYTICS/Project/NFL Play by Play  2016.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Date = col_character(),
##   SideofField = col_character(),
##   posteam = col_character(),
##   DefensiveTeam = col_character(),
##   Conference = col_character(),
##   HomeTeam = col_character(),
##   AwayTeam = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

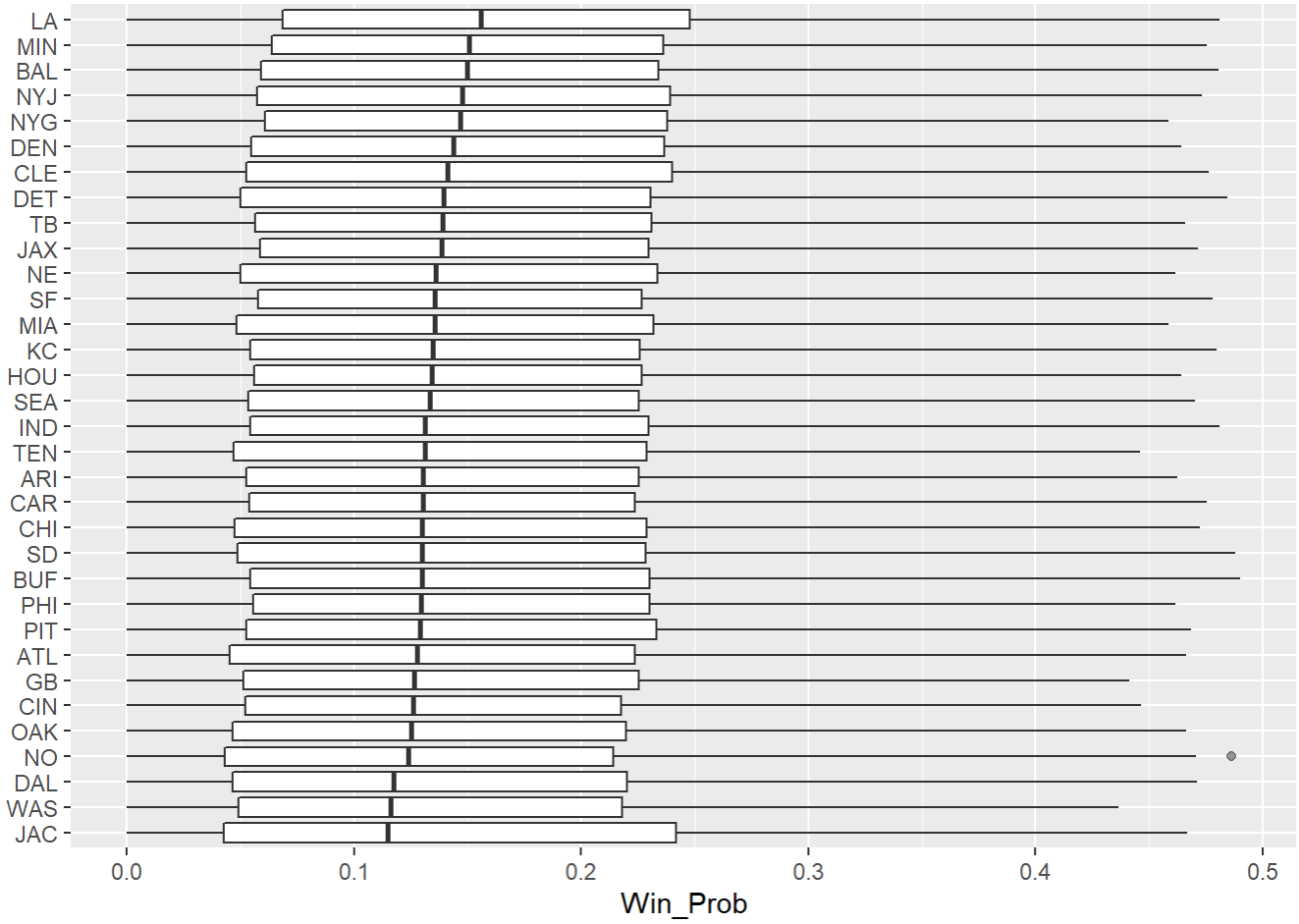
```
head(NFL)
```

```
## # A tibble: 6 x 29
##   Date  GameID Drive   qtr SideofField yrdln yrdline100 ydstogo ydsnet posteam
##   <chr>  <dbl> <dbl> <dbl> <chr>       <dbl>      <dbl>   <dbl>  <dbl> <chr>
## 1 1/3/~ 2.02e9     1     1 BUF            35         35       0      0 NYJ
## 2 1/3/~ 2.02e9     1     1 NYJ            20         80      10      0 NYJ
## 3 1/3/~ 2.02e9     1     1 NYJ            20         80      10      0 NYJ
## 4 1/3/~ 2.02e9     1     1 NYJ            20         80      10     -5 NYJ
## 5 1/3/~ 2.02e9     1     1 NYJ            15         85      15      1 NYJ
## 6 1/3/~ 2.02e9     1     1 NYJ            21         79       9      1 NYJ
## # ... with 19 more variables: DefensiveTeam <chr>, Conference <chr>,
## #   Yards.Gained <dbl>, HomeTeam <chr>, AwayTeam <chr>, No_Score_Prob <dbl>,
## #   Opp_Field_Goal_Prob <dbl>, Opp_Safety_Prob <dbl>, Opp_Touchdown_Prob <dbl>,
## #   Field_Goal_Prob <dbl>, Safety_Prob <dbl>, Touchdown_Prob <dbl>,
## #   Home_WP_pre <dbl>, Away_WP_pre <dbl>, Home_WP_post <dbl>,
## #   Away_WP_post <dbl>, Win_Prob <dbl>, WPA <dbl>, Season <dbl>
```

##Data Cleaning removing NA's

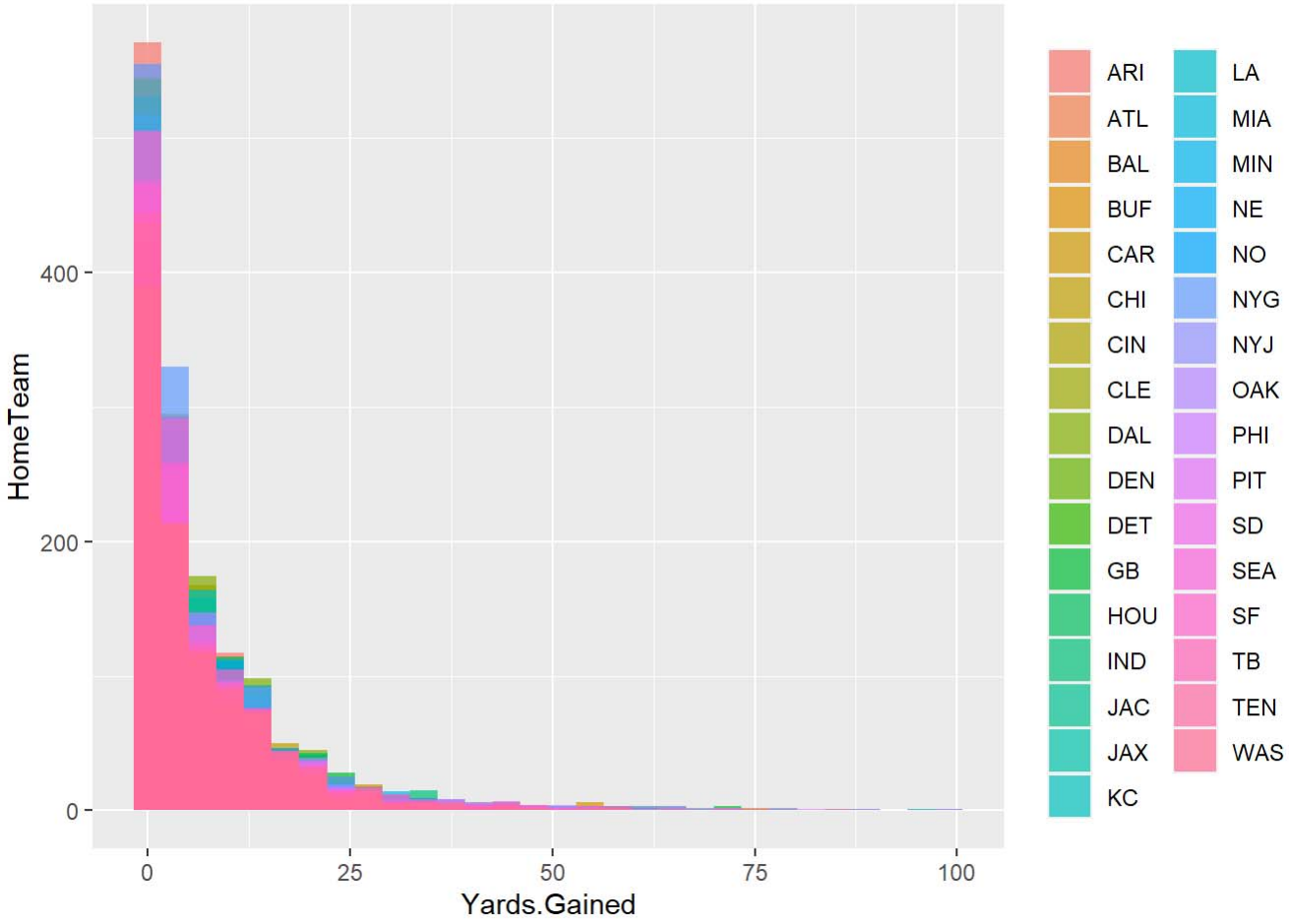
```
NFL_clean <- NFL %>%
drop_na()
```

```
Hteam_NFL <- NFL_clean %>%

  ggplot(aes(fct_reorder(HomeTeam, Opp_Touchdown_Prob),
    Opp_Touchdown_Prob,
    fill = Win_Prob
  )) +
  geom_boxplot(outlier.alpha = 0.5) +
  coord_flip() +
  labs(
    fill = NULL, x = NULL,
    y = "Win_Prob"
  )
Hteam_NFL
```



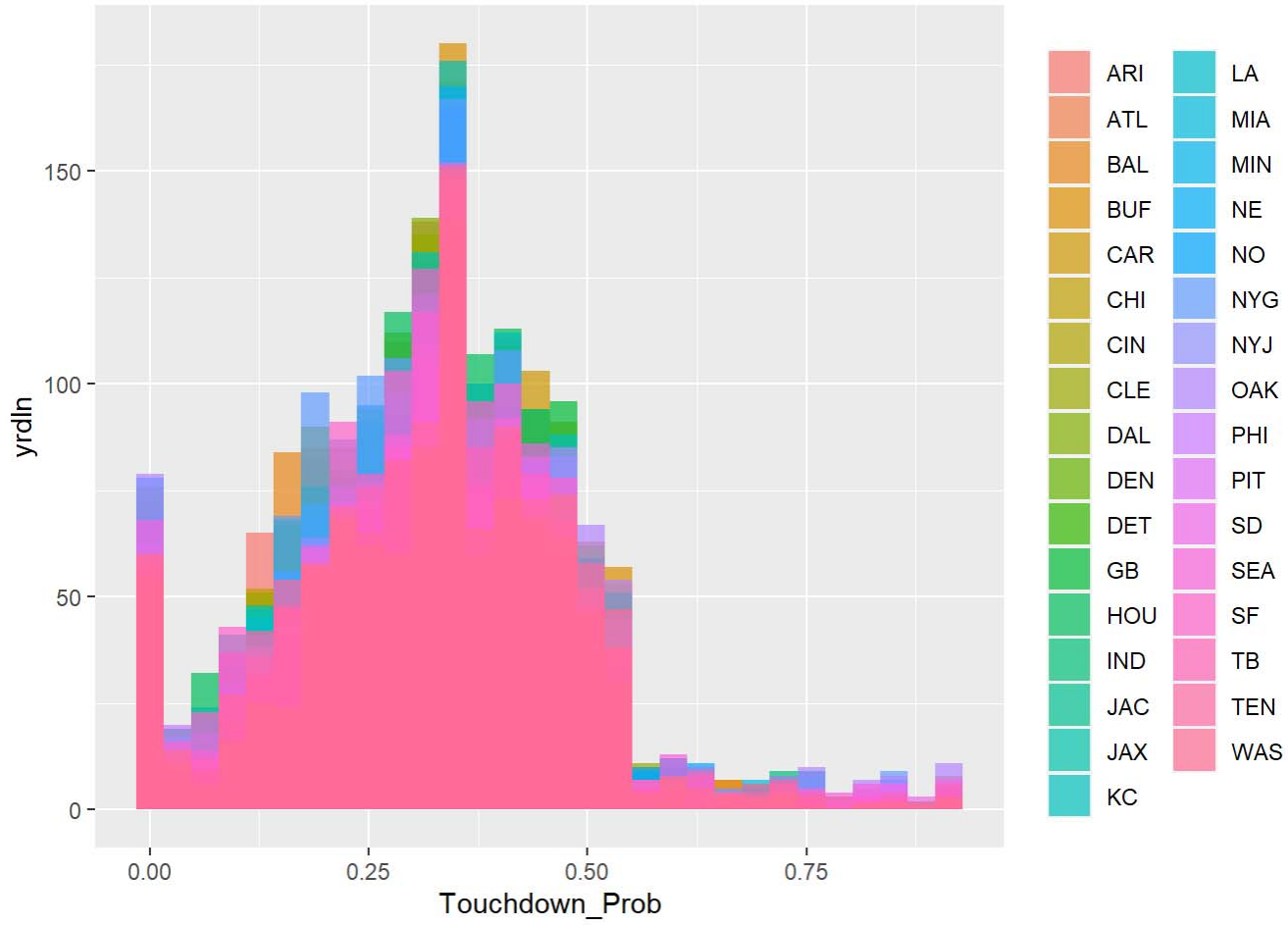

```
Yards.Gained <-NFL %>%
  distinct(HomeTeam, Yards.Gained, Touchdown_Prob) %>%
  ggplot(aes(Yards.Gained, fill = HomeTeam)) +
  geom_histogram(position = "identity", alpha = 0.7) +
  labs(
    x = "Yards.Gained",
    y = "HomeTeam",
    fill = NULL
  )
Yards.Gained
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
NFL_clean %>%
  distinct(HomeTeam, yrdln, Touchdown_Prob, Win_Prob) %>%
  ggplot(aes(Touchdown_Prob, fill = HomeTeam)) +
  geom_histogram(position = "identity", alpha = 0.7) +
  labs(
    x = "Touchdown_Prob",
    y = "yrdln",
    fill = NULL
  )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



##Select data set to use for clustering

```
df_NFL<- NFL_clean[,c(4,9,13:14,22,25:27)]
```

```
summary(df_NFL)
```

```
##       qtr            ydsnet        Yards.Gained      HomeTeam
##  Min.   :1.000   Min.   :-26.00   Min.   : 0.000   Length:39375
##  1st Qu.:2.000   1st Qu.:  6.00   1st Qu.: 0.000   Class :character
##  Median :3.000   Median : 21.00   Median : 3.000   Mode  :character
##  Mean   :2.553   Mean   : 27.51   Mean   : 5.881
##  3rd Qu.:4.000   3rd Qu.: 45.00   3rd Qu.: 8.000
##  Max.   :5.000   Max.   : 99.00   Max.   :99.000
##  Touchdown_Prob    Home_WP_post     Away_WP_post       Win_Prob
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2210   1st Qu.:0.3644   1st Qu.:0.2105   1st Qu.:0.2766
##  Median :0.3300   Median :0.5588   Median :0.4412   Median :0.5010
##  Mean   :0.3186   Mean   :0.5557   Mean   :0.4443   Mean   :0.4965
##  3rd Qu.:0.4154   3rd Qu.:0.7895   3rd Qu.:0.6356   3rd Qu.:0.7130
##  Max.   :0.9130   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
str(df_NFL)
```

```
## tibble [39,375 x 8] (S3: tbl_df/tbl/data.frame)
##  $ qtr           : num [1:39375] 1 1 1 1 1 1 1 1 1 ...
##  $ ydsnet        : num [1:39375] 0 0 0 -5 1 1 0 0 0 11 ...
##  $ Yards.Gained  : num [1:39375] 0 0 0 11 6 16 0 0 0 13 ...
##  $ HomeTeam      : chr [1:39375] "BUF" "BUF" "BUF" "BUF" ...
##  $ Touchdown_Prob: num [1:39375] 0.314 0.314 0.267 0.219 0.181 ...
##  $ Home_WP_post  : num [1:39375] 0.514 0.534 0.563 0.581 0.597 ...
##  $ Away_WP_post  : num [1:39375] 0.486 0.466 0.437 0.419 0.403 ...
##  $ Win_Prob      : num [1:39375] 0.486 0.486 0.466 0.437 0.419 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Date = col_character(),
##   ..   GameID = col_double(),
##   ..   Drive = col_double(),
##   ..   qtr = col_double(),
##   ..   SideofField = col_character(),
##   ..   yrdln = col_double(),
##   ..   yrdline100 = col_double(),
##   ..   ydstogo = col_double(),
##   ..   ydsnet = col_double(),
##   ..   posteam = col_character(),
##   ..   DefensiveTeam = col_character(),
##   ..   Conference = col_character(),
##   ..   Yards.Gained = col_double(),
##   ..   HomeTeam = col_character(),
##   ..   AwayTeam = col_character(),
##   ..   No_Score_Prob = col_double(),
##   ..   Opp_Field_Goal_Prob = col_double(),
##   ..   Opp_Safety_Prob = col_double(),
##   ..   Opp_Touchdown_Prob = col_double(),
##   ..   Field_Goal_Prob = col_double(),
##   ..   Safety_Prob = col_double(),
##   ..   Touchdown_Prob = col_double(),
##   ..   Home_WP_pre = col_double(),
##   ..   Away_WP_pre = col_double(),
##   ..   Home_WP_post = col_double(),
##   ..   Away_WP_post = col_double(),
##   ..   Win_Prob = col_double(),
##   ..   WPA = col_double(),
##   ..   Season = col_double()
##   .. )
```

```
df_NFL %>%
mutate(NFC, H_Team =case_when(HomeTeam == "ARI" ~1,
                         HomeTeam == "ATL" ~2,
                         HomeTeam == "BAL" ~3,
                          HomeTeam == "BUF" ~4,
                          HomeTeam == "CAR" ~5,
                          HomeTeam == "CHI" ~6,
                          HomeTeam == "CIN" ~7,
                          HomeTeam == "DAL"~8,
                          HomeTeam == "DEN" ~9,
                          HomeTeam == "DET" ~10,
                          HomeTeam == "GB" ~11,
                          HomeTeam == "HOU" ~12,
                          HomeTeam == "IND" ~13,
                          HomeTeam == "JAC" ~14,
                          HomeTeam == "JAX" ~15,
                          HomeTeam == "KC" ~16,
                          HomeTeam == "LA" ~17,
                          HomeTeam == "MIA" ~18,
                          HomeTeam == "MIN" ~19,
                          HomeTeam == "NE" ~20,
                          HomeTeam == "NO" ~21,
                          HomeTeam == "NYG" ~22,
                          HomeTeam == "NYJ" ~23,
                          HomeTeam == "OAK" ~24,
                          HomeTeam == "PHI" ~25,
                          HomeTeam == "PIT" ~26,
                          HomeTeam == "SD" ~27,
                          HomeTeam == "SEA" ~28,
                          HomeTeam == "SF" ~29,
                         HomeTeam == "STL" ~30,
                         HomeTeam == "TB" ~31,
                         HomeTeam == "TEN" ~32,
                         HomeTeam == "WAS" ~33,
                         HomeTeam == "CLE" ~34,))->df_NFL
```

```
##REMOVE HOMETEAM
df_NFL = df_NFL[, -c(4)]
```

```
##rename to nfl
NFL = df_NFL
```

##skip

```
NFL = ddply(NFL, 'H_Team', numcolwise(sum))
NFL
```

```
##    H_Team  qtr ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post
## 1       1 3682  39187         8117      456.63445    755.31499    681.68501
## 2       2 2985  38243         7804      385.10237    692.69813    489.30187
## 3       3 3119  29936         6770      374.40962    808.66083    423.33917
## 4       4 3553  38991         8944      439.15960    679.73258    702.26743
## 5       5 3509  35776         7887      442.54351    859.11309    524.88691
## 6       6 3384  38112         8185      427.91676    622.80567    703.19433
## 7       7 3327  37716         7561      409.58422    791.09548    483.90452
## 8       8 3559  42092         8334      447.74842    851.80825    524.19175
## 9       9 3409  32438         6751      410.61910    630.20774    671.79226
## 10     10 2505  29808         5733      312.59343    582.06585    403.93415
## 11     11 3367  36685         8279      432.78692    812.13199    526.86801
## 12     12 3520  33395         8120      425.76977    787.66605    583.33395
## 13     13 3118  35764         7396      395.41963    565.92246    668.07754
## 14     14  389   4463          782       48.56507     58.88821     97.11179
## 15     15 2922  28209         5729      352.47047    510.68769    623.31231
## 16     16 3442  36357         8105      436.71930    862.85218    489.14782
## 17     17 2615  26067         5530      307.76251    473.97950    549.02050
## 18     18 3138  33077         7575      378.33198    708.27852    501.72148
## 19     19 2679  26982         5999      339.18939    599.06175    457.93825
## 20     20 3138  33571         7191      403.17104    870.68662    375.31338
## 21     21 3254  39188         8072      419.69062    686.38017    595.61983
## 22     22 3588  35935         8393      444.67298    733.39339    688.60661
## 23     23 2545  26408         6012      312.98229    436.48830    573.51170
## 24     24 3214  36462         7648      423.42604    693.11850    565.88150
## 25     25 2667  30010         6171      329.33820    666.05141    378.94859
## 26     26 2826  30695         6603      351.20995    694.20227    421.79773
## 27     27 2749  28124         6028      347.60618    708.51629    350.48371
## 28     28 3071  31134         7298      388.67163    803.52418    407.47582
## 29     29 3390  35544         7448      409.23059    618.22464    685.77536
## 30     31 2896  29628         6084      364.34432    583.03665    532.96335
## 31     32 2793  30766         6445      345.16570    705.28085    400.71915
## 32     33 2736  33985         6655      354.07463    550.12682    519.87317
## 33     34 3428  38316         7903      428.14488    479.07551    891.92449
##    Win_Prob
## 1  727.5743
## 2  587.2608
## 3  604.9552
## 4  686.2961
## 5  684.5624
## 6  668.0339
## 7  627.6124
## 8  662.2020
## 9  642.2399
## 10 491.9249
## 11 649.5789
## 12 699.9786
## 13 615.8921
## 14  79.2553
## 15 550.5308
## 16 690.9194
## 17 495.0105
## 18 596.9013
```

```
## 19 525.8290
## 20 637.3852
## 21 621.4181
## 22 706.9388
## 23 510.9061
## 24 645.8693
## 25 530.2664
## 26 550.8505
## 27 514.7282
## 28 579.9980
## 29 666.4753
## 30 553.2447
## 31 532.5979
## 32 528.1760
## 33 684.0921
```

```
Cln_NFL <-NFL
```

```
##Four Cluster by Author
set.seed((123))
model_r<- kmeans(NFL, centers =4, nstart =50)
model_r
```

```
## K-means clustering with 4 clusters of sizes 12, 1, 12, 8
##
## Cluster means:
##     H_Team      qtr   ydsnet Yards.Gained Touchdown_Prob Home_WP_post
## 1 21.33333 2782.250 28980.58     6200.167      343.81198    630.96296
## 2 14.00000  389.000  4463.00      782.000       48.56507     58.88821
## 3 17.66667 3297.417 34915.75     7620.667      413.06379    724.31017
## 4 10.37500 3396.500 38980.62     8115.000      426.74766    694.86385
##   Away_WP_post Win_Prob
## 1    460.28704 536.7368
## 2     97.11179  79.2553
## 3    566.77317 647.0764
## 4    634.01115 658.0612
##
## Clustering vector:
##  [1] 4 4 1 4 3 4 4 4 3 1 3 3 3 2 1 3 1 3 1 3 4 3 1 3 1 1 1 1 3 1 1 3 4
##
## Within cluster sum of squares by cluster:
## [1] 37714273        0 29458239 14867557
##  (between_SS / total_SS =  94.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
model_r$size
```

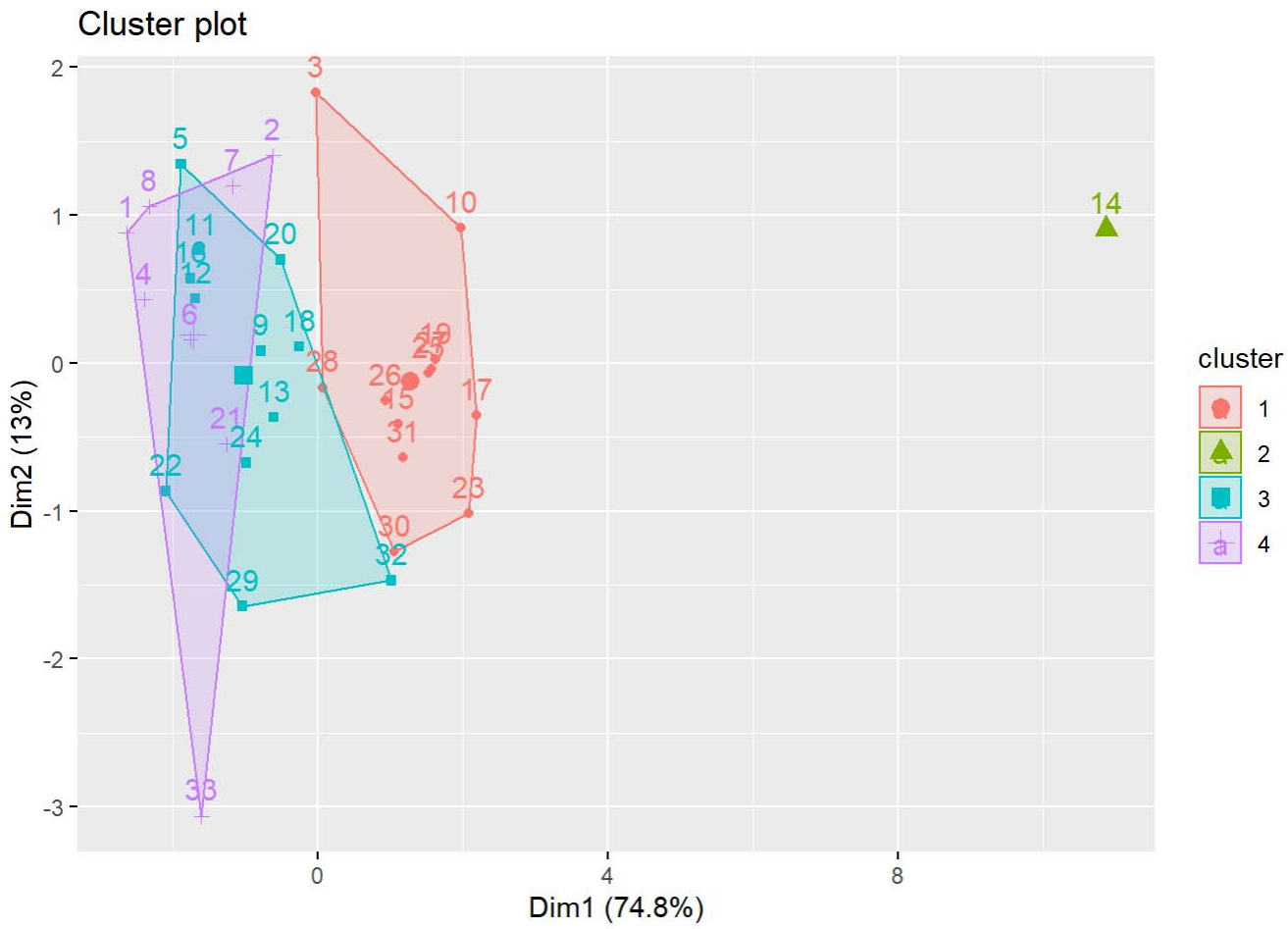
```
## [1] 12  1 12  8
```

```
table(NFL$H_Team, model_r$cluster)
```

```
##
##      1 2 3 4
##   1  0 0 0 1
##   2  0 0 0 1
##   3  1 0 0 0
##   4  0 0 0 1
##   5  0 0 1 0
##   6  0 0 0 1
##   7  0 0 0 1
##   8  0 0 0 1
##   9  0 0 1 0
##   10 1 0 0 0
##   11 0 0 1 0
##   12 0 0 1 0
##   13 0 0 1 0
##   14 0 1 0 0
##   15 1 0 0 0
##   16 0 0 1 0
##   17 1 0 0 0
##   18 0 0 1 0
##   19 1 0 0 0
##   20 0 0 1 0
##   21 0 0 0 1
##   22 0 0 1 0
##   23 1 0 0 0
##   24 0 0 1 0
##   25 1 0 0 0
##   26 1 0 0 0
##   27 1 0 0 0
##   28 1 0 0 0
##   29 0 0 1 0
##   31 1 0 0 0
##   32 1 0 0 0
##   33 0 0 1 0
##   34 0 0 0 1
```

```
fviz_cluster(model_r,data=NFL)
```

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4
```



##Select data set to use for clustering

{r} ###removing Drive and qrt to inspect at the data ##R_data = Clean_data[, -c(1:2)]

```
##TwoCluster by team
set.seed((123))
model_r1<- kmeans(NFL, centers =2, nstart =50)
model_r1
```

```
## K-means clustering with 2 clusters of sizes 1, 32
##
## Cluster means:
##     H_Team  qtr   ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post
## 1 14.00000  389  4463.00      782.000       48.56507     58.88821     97.11179
## 2 17.21875 3129 33706.28     7211.562      390.51533    681.94339    543.65036
##   Win_Prob
## 1  79.2553
## 2 608.4453
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]         0 613650664
##  (between_SS / total_SS =  58.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

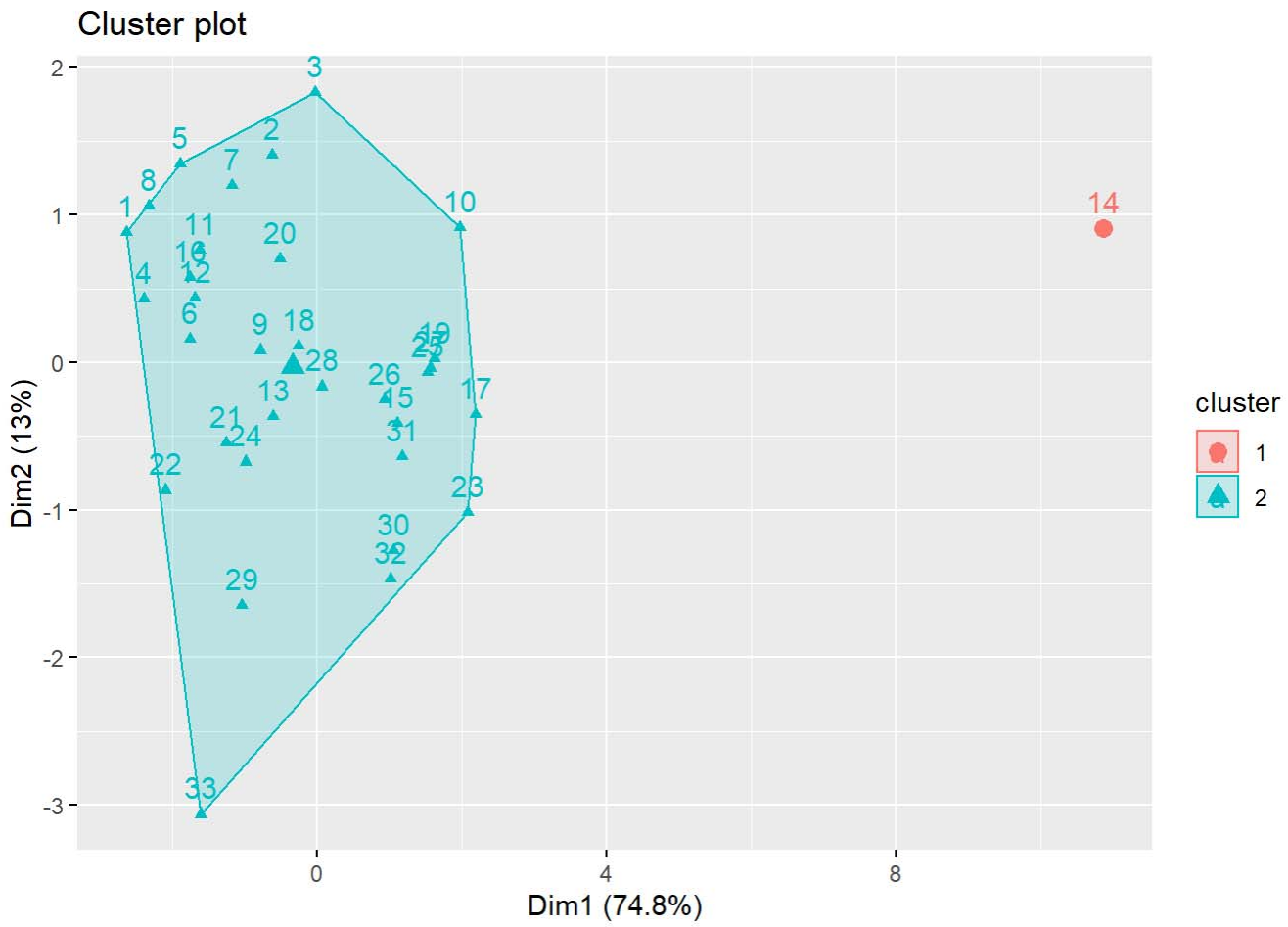
```
model_r1$size
```

```
## [1]  1 32
```

```
table(NFL$H_Team, model_r1$cluster)
```

```
##
##      1 2
##   1  0 1
##   2  0 1
##   3  0 1
##   4  0 1
##   5  0 1
##   6  0 1
##   7  0 1
##   8  0 1
##   9  0 1
##   10 0 1
##   11 0 1
##   12 0 1
##   13 0 1
##   14 1 0
##   15 0 1
##   16 0 1
##   17 0 1
##   18 0 1
##   19 0 1
##   20 0 1
##   21 0 1
##   22 0 1
##   23 0 1
##   24 0 1
##   25 0 1
##   26 0 1
##   27 0 1
##   28 0 1
##   29 0 1
##   31 0 1
##   32 0 1
##   33 0 1
##   34 0 1
```

```
fviz_cluster(model_r1,data=NFL)
```



What is the Best method

To determine best method to use

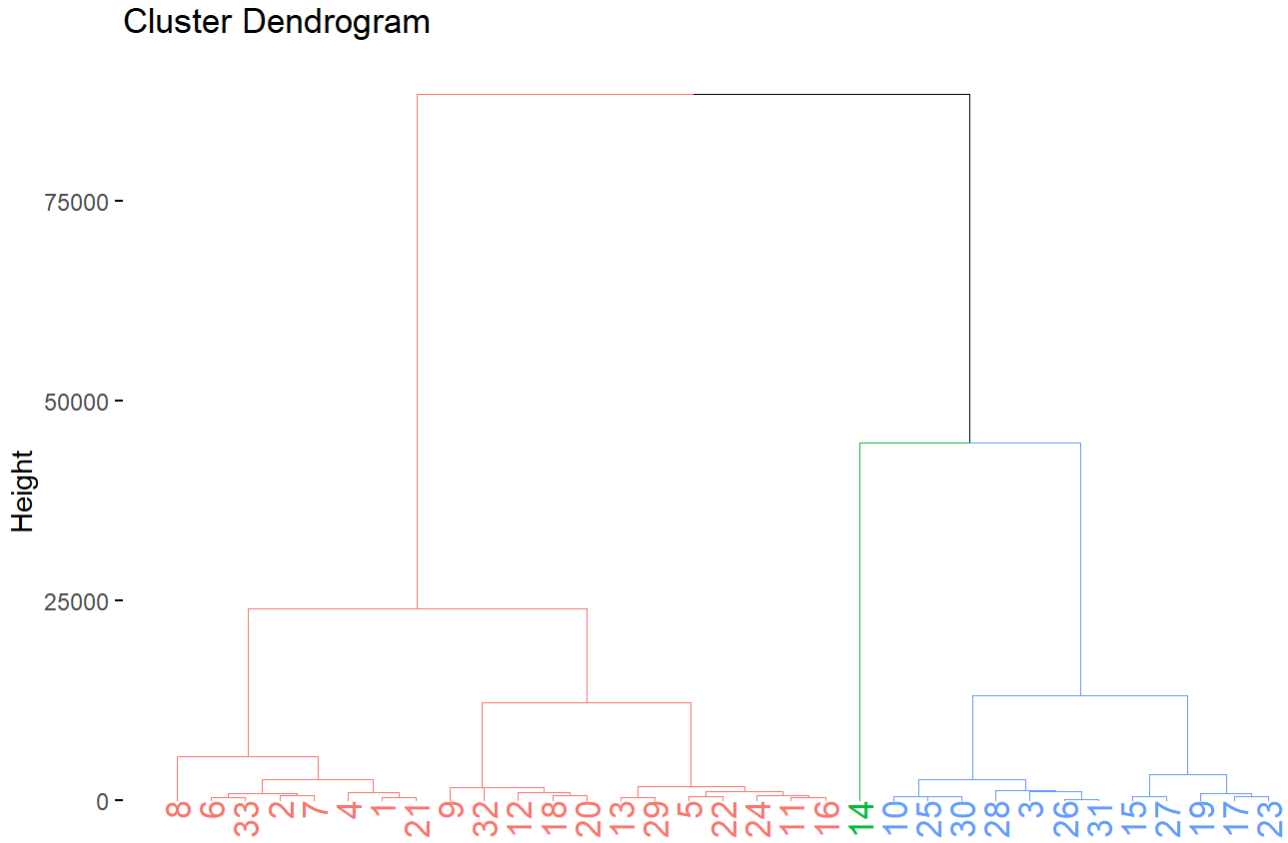
```
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward.D")

# function to compute coefficient
ac <- function(x) {
  agnes(NFL, method = x)$ac
}

map_dbl(m, ac)
```

```
##   average    single  complete    ward.D
## 0.9439623 0.9419217 0.9481225 0.9516177
```

```
set.seed(123)
d <- dist(NFL, method = "euclidean")
```

```
# Hierarchical clustering using Ward.D Linkage
hc1 <- hclust(d, method = "ward.D" )
```

```
###plot(hc1, cex = 0.6, hang = -1)
fviz_dend(x= hc1, cex = 0.90, hang = -1, lwd = 0.10, k=3)
```



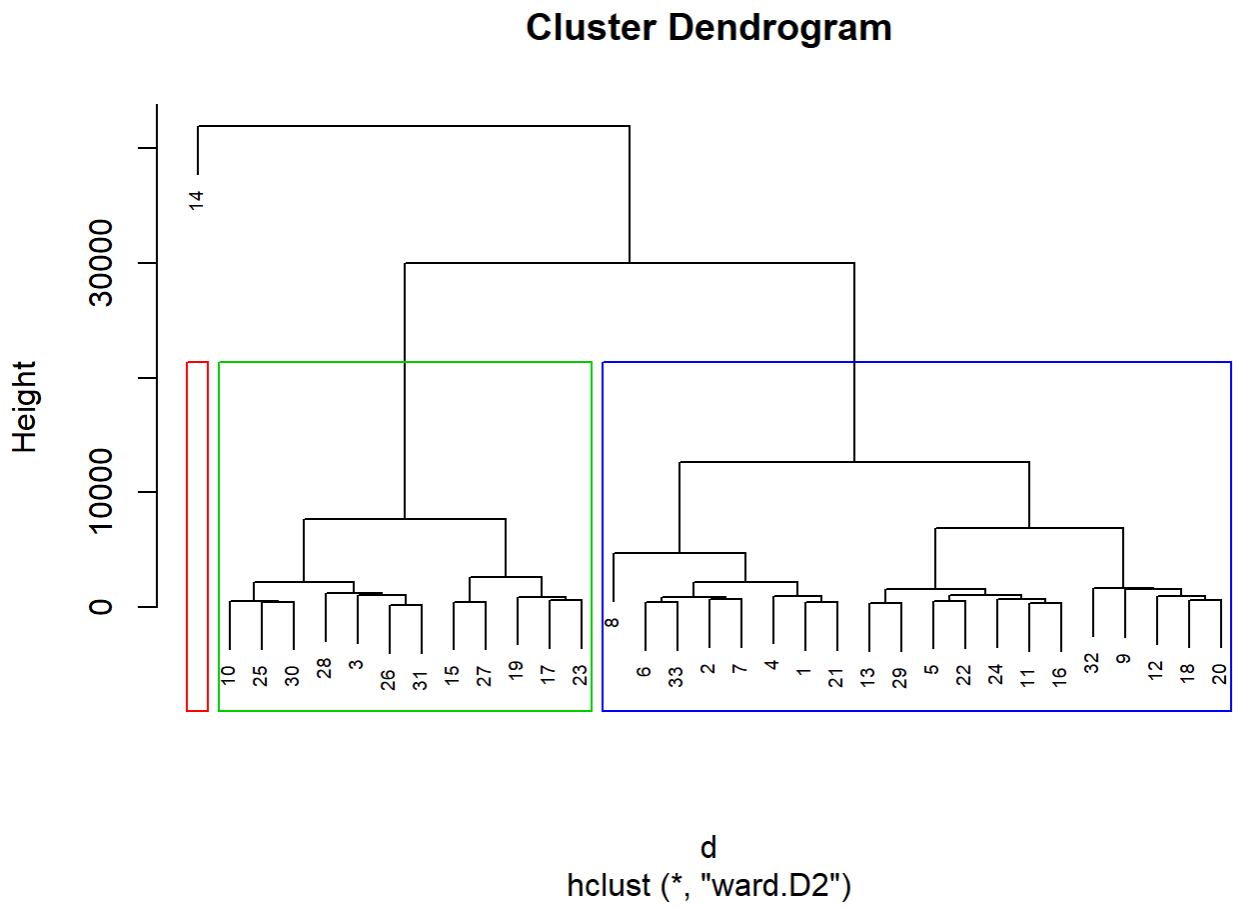

```
k_colors=c("read","green","blue","magenta")
```

The D is is euclidean

```
hc2 <- hclust(d, method = "ward.D2" )
sub_grp <- cutree(hc2, k = 4)
# Number of members in each cluster
table(sub_grp)
```

```
## sub_grp
##  1  2  3  4
##  8 12 12  1
```

```
plot(hc2, cex = 0.6)
rect.hclust(hc2, k = 3, border = 2:5)
```



##distance matrix Calculate the distance matrix

```
res.dist <- dist(NFL, method = "euclidean")

# Compute 2 hierarchical clusterings
hc1 <- hclust(res.dist, method = "ward.D")
hc1
```

```
##
## Call:
## hclust(d = res.dist, method = "ward.D")
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 33
```

```
hc2 <- hclust(res.dist, method = "ward.D2")
hc2
```

```
##
## Call:
## hclust(d = res.dist, method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 33
```

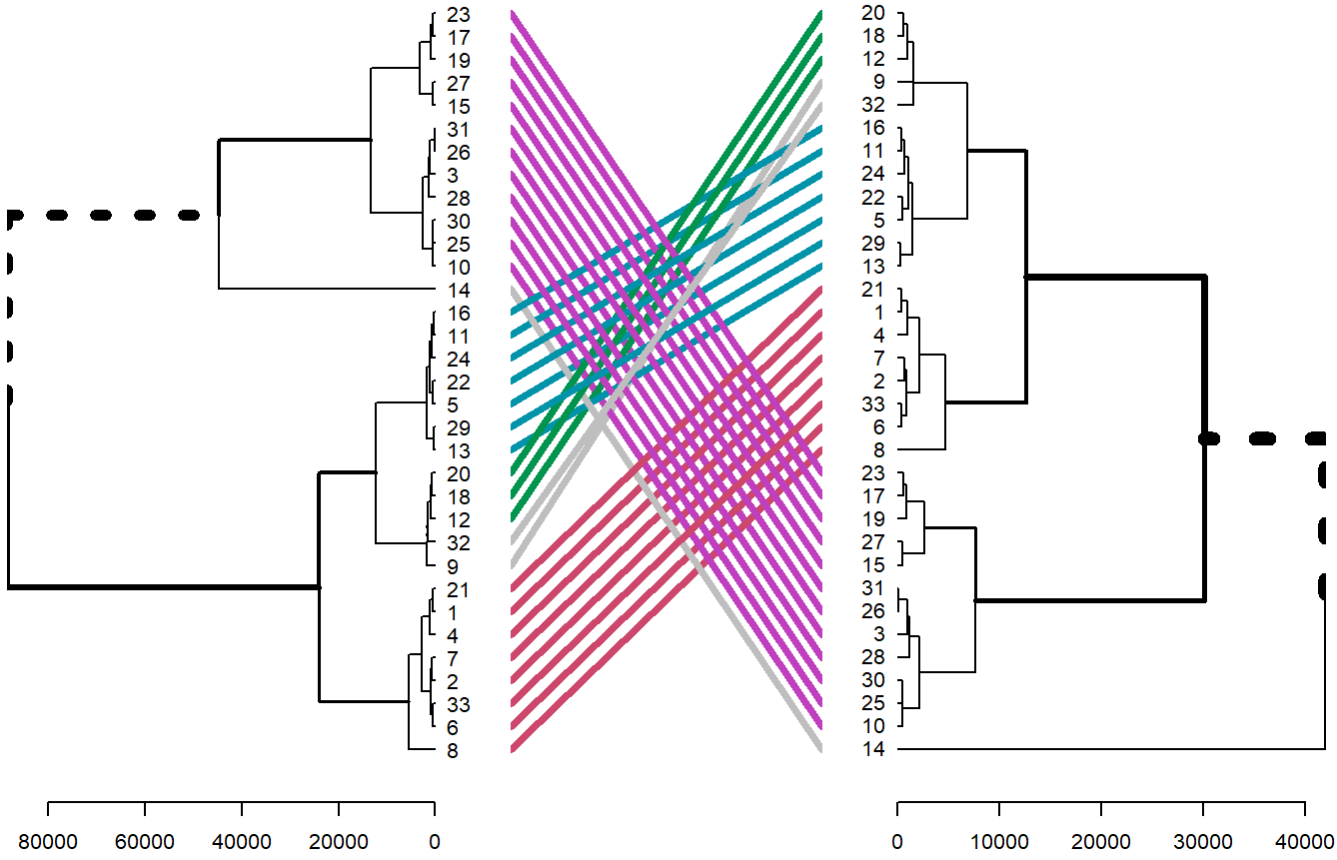
```
# Create two dendrograms
dend1 <- as.dendrogram (hc1)
dend1
```

```
## 'dendrogram' with 2 branches and 33 members total, at height 88397.5
```

```
dend2 <- as.dendrogram (hc2)
dend2
```

```
## 'dendrogram' with 2 branches and 33 members total, at height 41894.55
```

```
tanglegram(dend1, dend2)
```



```
Winning_Prob <- df_NFL
```

##Renaming df_NFL to Winning_Prob to save the data integrity above #Selecting the following columns for the model

```
Winning_Prob<- df_NFL %>%
  select(H_Team, ydsnet,Yards.Gained,Touchdown_Prob,Home_WP_post,Away_WP_post,Win_Prob

  )
```

##split data to 75% training and 25% testing

```
library(tidymodels)
set.seed(1234)
data_split <- Winning_Prob %>%
  initial_split(strata = H_Team)

nfl_train <- training(data_split)
nfl_test <- testing(data_split)
```

```
nfl_train
```

```
## # A tibble: 29,533 x 7
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
##     <dbl>  <dbl>        <dbl>          <dbl>        <dbl>        <dbl>    <dbl>
##  1      4      0            0          0.314        0.514        0.486    0.486
##  2      4      0            0          0.314        0.534        0.466    0.486
##  3      4      0            0          0.267        0.563        0.437    0.466
##  4      4     -5           11          0.219        0.581        0.419    0.437
##  5      4      1           16          0.171        0.560        0.440    0.403
##  6      4      0            0          0.437        0.540        0.460    0.560
##  7      4      0            0          0.309        0.466        0.534    0.513
##  8      4     11           13          0.202        0.498        0.502    0.438
##  9      4     12            1          0.323        0.515        0.485    0.502
## 10      4     15            3          0.285        0.527        0.473    0.485
## # ... with 29,523 more rows
```

```
nfl_test
```

```
## # A tibble: 9,842 x 7
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
##     <dbl>  <dbl>        <dbl>          <dbl>        <dbl>        <dbl>    <dbl>
##  1      4      1            6          0.181        0.597        0.403    0.419
##  2      4      0            0          0.199        0.533        0.467    0.466
##  3      4      0            0          0.391        0.520        0.480    0.542
##  4      4      0            0          0.340        0.492        0.508    0.520
##  5      4      0            0          0.279        0.446        0.554    0.492
##  6      4     25            4          0.526        0.624        0.376    0.628
##  7      4     25            0          0            0.728        0.272    0.732
##  8      4      0            0          0.320        0.728        0.272    0.272
##  9      4      1            0          0.177        0.776        0.224    0.190
## 10      4     20            2          0.453        0.804        0.196    0.805
## # ... with 9,832 more rows
```

```
nfl_train <- ddply(Winning_Prob, 'H_Team', numcolwise(sum))
nfl_train
```

```
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
## 1       1  39187         8117      456.63445    755.31499    681.68501 727.5743
## 2       2  38243         7804      385.10237    692.69813    489.30187 587.2608
## 3       3  29936         6770      374.40962    808.66083    423.33917 604.9552
## 4       4  38991         8944      439.15960    679.73258    702.26743 686.2961
## 5       5  35776         7887      442.54351    859.11309    524.88691 684.5624
## 6       6  38112         8185      427.91676    622.80567    703.19433 668.0339
## 7       7  37716         7561      409.58422    791.09548    483.90452 627.6124
## 8       8  42092         8334      447.74842    851.80825    524.19175 662.2020
## 9       9  32438         6751      410.61910    630.20774    671.79226 642.2399
## 10     10  29808         5733      312.59343    582.06585    403.93415 491.9249
## 11     11  36685         8279      432.78692    812.13199    526.86801 649.5789
## 12     12  33395         8120      425.76977    787.66605    583.33395 699.9786
## 13     13  35764         7396      395.41963    565.92246    668.07754 615.8921
## 14     14   4463          782       48.56507     58.88821     97.11179  79.2553
## 15     15  28209         5729      352.47047    510.68769    623.31231 550.5308
## 16     16  36357         8105      436.71930    862.85218    489.14782 690.9194
## 17     17  26067         5530      307.76251    473.97950    549.02050 495.0105
## 18     18  33077         7575      378.33198    708.27852    501.72148 596.9013
## 19     19  26982         5999      339.18939    599.06175    457.93825 525.8290
## 20     20  33571         7191      403.17104    870.68662    375.31338 637.3852
## 21     21  39188         8072      419.69062    686.38017    595.61983 621.4181
## 22     22  35935         8393      444.67298    733.39339    688.60661 706.9388
## 23     23  26408         6012      312.98229    436.48830    573.51170 510.9061
## 24     24  36462         7648      423.42604    693.11850    565.88150 645.8693
## 25     25  30010         6171      329.33820    666.05141    378.94859 530.2664
## 26     26  30695         6603      351.20995    694.20227    421.79773 550.8505
## 27     27  28124         6028      347.60618    708.51629    350.48371 514.7282
## 28     28  31134         7298      388.67163    803.52418    407.47582 579.9980
## 29     29  35544         7448      409.23059    618.22464    685.77536 666.4753
## 30     31  29628         6084      364.34432    583.03665    532.96335 553.2447
## 31     32  30766         6445      345.16570    705.28085    400.71915 532.5979
## 32     33  33985         6655      354.07463    550.12682    519.87317 528.1760
## 33     34  38316         7903      428.14488    479.07551    891.92449 684.0921
```

```
nfl_test <- ddply(Winning_Prob, 'H_Team', numcolwise(sum))
nfl_test
```

```
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
## 1       1  39187         8117      456.63445    755.31499    681.68501 727.5743
## 2       2  38243         7804      385.10237    692.69813    489.30187 587.2608
## 3       3  29936         6770      374.40962    808.66083    423.33917 604.9552
## 4       4  38991         8944      439.15960    679.73258    702.26743 686.2961
## 5       5  35776         7887      442.54351    859.11309    524.88691 684.5624
## 6       6  38112         8185      427.91676    622.80567    703.19433 668.0339
## 7       7  37716         7561      409.58422    791.09548    483.90452 627.6124
## 8       8  42092         8334      447.74842    851.80825    524.19175 662.2020
## 9       9  32438         6751      410.61910    630.20774    671.79226 642.2399
## 10     10  29808         5733      312.59343    582.06585    403.93415 491.9249
## 11     11  36685         8279      432.78692    812.13199    526.86801 649.5789
## 12     12  33395         8120      425.76977    787.66605    583.33395 699.9786
## 13     13  35764         7396      395.41963    565.92246    668.07754 615.8921
## 14     14   4463          782       48.56507     58.88821     97.11179  79.2553
## 15     15  28209         5729      352.47047    510.68769    623.31231 550.5308
## 16     16  36357         8105      436.71930    862.85218    489.14782 690.9194
## 17     17  26067         5530      307.76251    473.97950    549.02050 495.0105
## 18     18  33077         7575      378.33198    708.27852    501.72148 596.9013
## 19     19  26982         5999      339.18939    599.06175    457.93825 525.8290
## 20     20  33571         7191      403.17104    870.68662    375.31338 637.3852
## 21     21  39188         8072      419.69062    686.38017    595.61983 621.4181
## 22     22  35935         8393      444.67298    733.39339    688.60661 706.9388
## 23     23  26408         6012      312.98229    436.48830    573.51170 510.9061
## 24     24  36462         7648      423.42604    693.11850    565.88150 645.8693
## 25     25  30010         6171      329.33820    666.05141    378.94859 530.2664
## 26     26  30695         6603      351.20995    694.20227    421.79773 550.8505
## 27     27  28124         6028      347.60618    708.51629    350.48371 514.7282
## 28     28  31134         7298      388.67163    803.52418    407.47582 579.9980
## 29     29  35544         7448      409.23059    618.22464    685.77536 666.4753
## 30     31  29628         6084      364.34432    583.03665    532.96335 553.2447
## 31     32  30766         6445      345.16570    705.28085    400.71915 532.5979
## 32     33  33985         6655      354.07463    550.12682    519.87317 528.1760
## 33     34  38316         7903      428.14488    479.07551    891.92449 684.0921
```

```
nfold<-trainControl(method = "repeatedcv",
                        number=5,
                        repeats=3)
```

```
set.seed(123)

fit<-train(H_Team~.,
           data = nfl_train,
           methods ='knn',
           tuneLenght = 10,
           trControl=nfold,
           preProc=c("center", "scale"))## subtract mean from each value then/stv
fit
```

```
## Random Forest
##
## 33 samples
##  6 predictor
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 27, 27, 26, 26, 26, 26, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     10.05410  0.2303156  8.434631
##   4     10.19159  0.2302173  8.500743
##   6     10.37854  0.2072914  8.685976
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

#Model

```
model <- lm(H_Team ~ ., data = nfl_train, usekernel = T)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'usekernel' will be disregarded
```

```
model
```

```
##
## Call:
## lm(formula = H_Team ~ ., data = nfl_train, usekernel = T)
##
## Coefficients:
##    (Intercept)          ydsnet    Yards.Gained  Touchdown_Prob    Home_WP_post
##     27.8447214      -0.0007220      -0.0006204       0.2875772      -0.0619589
##   Away_WP_post        Win_Prob
##     -0.0430845      -0.0473943
```

```
pred<- predict(model,nfl_test)
```

```
train_res<- table(predicted =pred, Actual=nfl_train$H_Team)
train_res
```

```
##                  Actual
## predicted          1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##   12.689604359268  0 0 1 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.0288293489905 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##   14.1025987156393 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.2747590708608 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##   14.3021450773587 0 0 0 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.307444638655  0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.3345633971436 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.6913248613985 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14.7643844158529 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
##   14.8774189243945 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##   15.1842801795187 1 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15.2959430839499 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15.539936328235  0 0 0 1 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15.8105903215607 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15.8810413633665 0 0 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15.9590240391654 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  1  0
##   16.1001772689244 0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   16.8785253625923 0 0 0 0 0 0 0 0 0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
##   17.5976691196778 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  1  0  0
##   17.6182312248852 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  1  0  0  0  0  0  0
##   17.7649241111657 0 0 0 0 0 1 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   17.7831343095084 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   17.8711461759375 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   18.1127425954764 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  0  0  0
##   19.0884769545851 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  1
##   19.4881471397091 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   19.8932912156928 0 0 0 0 0 0 0 0 1  0  0  0  0  0  0  0  0  0  0  0  0  0
##   20.3700704738778 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   20.4180009428678 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  1  0  0  0  0
##   20.6078821207692 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   20.6987754082752 0 0 0 0 0 0 0 0 0  0  0  0  0  1  0  0  0  0  0  0  0  0
##   22.1501002109921 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   26.5148172397051 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  0  0  0
##                  Actual
## predicted          24 25 26 27 28 29 31 32 33 34
##   12.689604359268   0  0  0  0  0  0  0  0  0  0
##   14.0288293489905  0  0  0  0  0  0  0  0  0  0
##   14.1025987156393  0  0  0  0  0  0  0  0  0  0
##   14.2747590708608  0  0  0  0  0  0  0  0  0  0
##   14.3021450773587  0  0  0  0  0  0  0  0  0  0
##   14.307444638655   0  0  0  0  0  0  0  0  0  0
##   14.3345633971436  0  1  0  0  0  0  0  0  0  0
##   14.6913248613985  0  0  0  0  0  0  0  1  0  0
##   14.7643844158529  0  0  0  0  0  0  0  0  0  0
##   14.8774189243945  0  0  0  0  0  0  0  0  0  0
##   15.1842801795187  0  0  0  0  0  0  0  0  0  0
##   15.2959430839499  0  0  1  0  0  0  0  0  0  0
##   15.539936328235   0  0  0  0  0  0  0  0  0  0
##   15.8105903215607  0  0  0  0  0  1  0  0  0  0
##   15.8810413633665  0  0  0  0  0  0  0  0  0  0
##   15.9590240391654  0  0  0  0  0  0  0  0  0  0
```

```
##   16.1001772689244  0  0  0  0  0  0  0  0  0  0
##   16.8785253625923  0  0  0  0  0  0  0  0  0  0
##   17.5976691196778  0  0  0  0  0  0  0  0  0  0
##   17.6182312248852  0  0  0  0  0  0  0  0  0  0
##   17.7649241111657  0  0  0  0  0  0  0  0  0  0
##   17.7831343095084  0  0  0  0  1  0  0  0  0  0
##   17.8711461759375  0  0  0  0  0  0  0  0  0  1
##   18.1127425954764  0  0  0  0  0  0  0  0  0  0
##   19.0884769545851  0  0  0  0  0  0  0  0  0  0
##   19.4881471397091  0  0  0  0  0  0  0  0  1  0
##   19.8932912156928  0  0  0  0  0  0  0  0  0  0
##   20.3700704738778  0  0  0  1  0  0  0  0  0  0
##   20.4180009428678  0  0  0  0  0  0  0  0  0  0
##   20.6078821207692  1  0  0  0  0  0  0  0  0  0
##   20.6987754082752  0  0  0  0  0  0  0  0  0  0
##   22.1501002109921  0  0  0  0  0  0  1  0  0  0
##   26.5148172397051  0  0  0  0  0  0  0  0  0  0
```

##Additional looking for collinearity

```
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.6.3
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:mclust':
##
##     sim
```

```
## The following object is masked from 'package:randomForest':
##
##     outlier
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

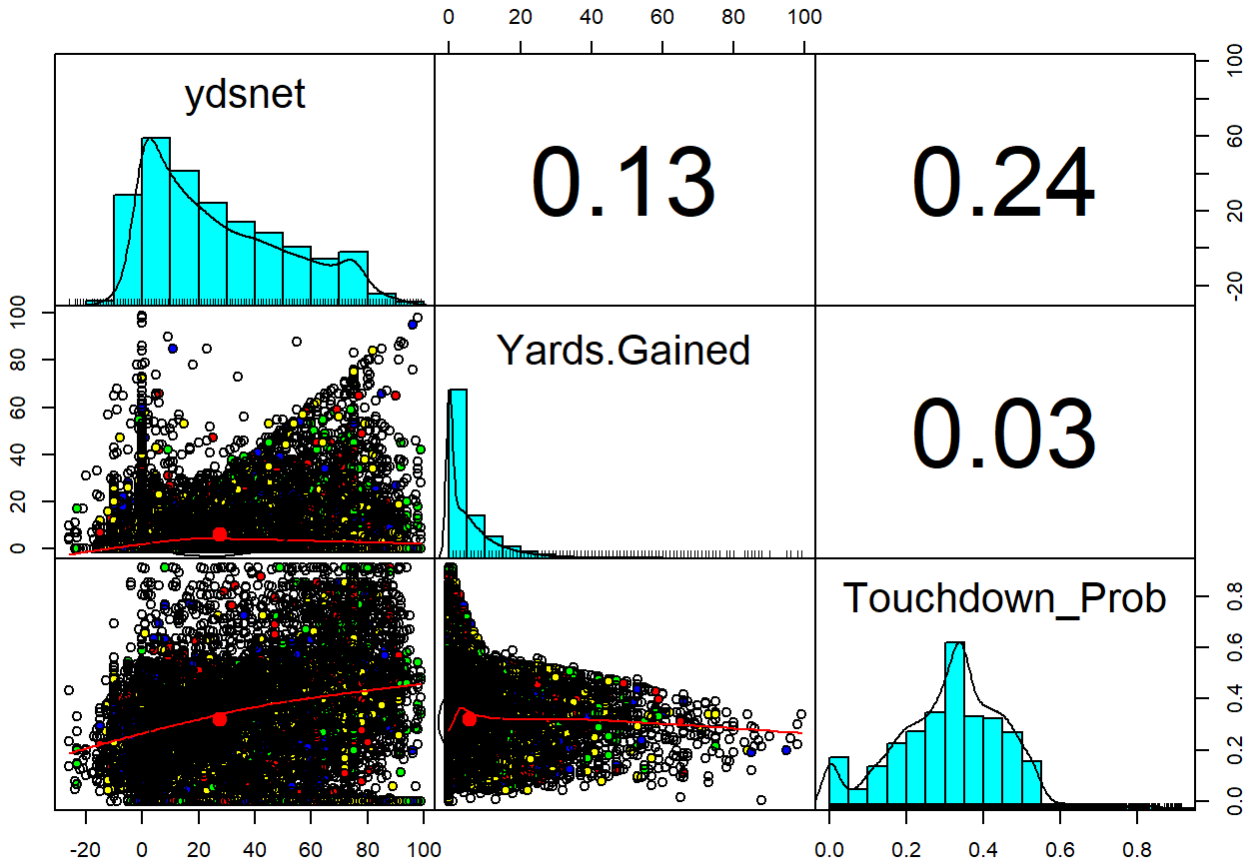
```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```
pairs.panels(Winning_Prob[2:4],
             gap=0, bg=c("red","green","blue","yellow")[Winning_Prob$H_Team],pch = 21)
```



##Decision Tree

```
library(caret)
library(knitr)
library(rpart)
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:dendextend':
##
##     prune
```

```
## The following object is masked from 'package:dials':
##
##     prune
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

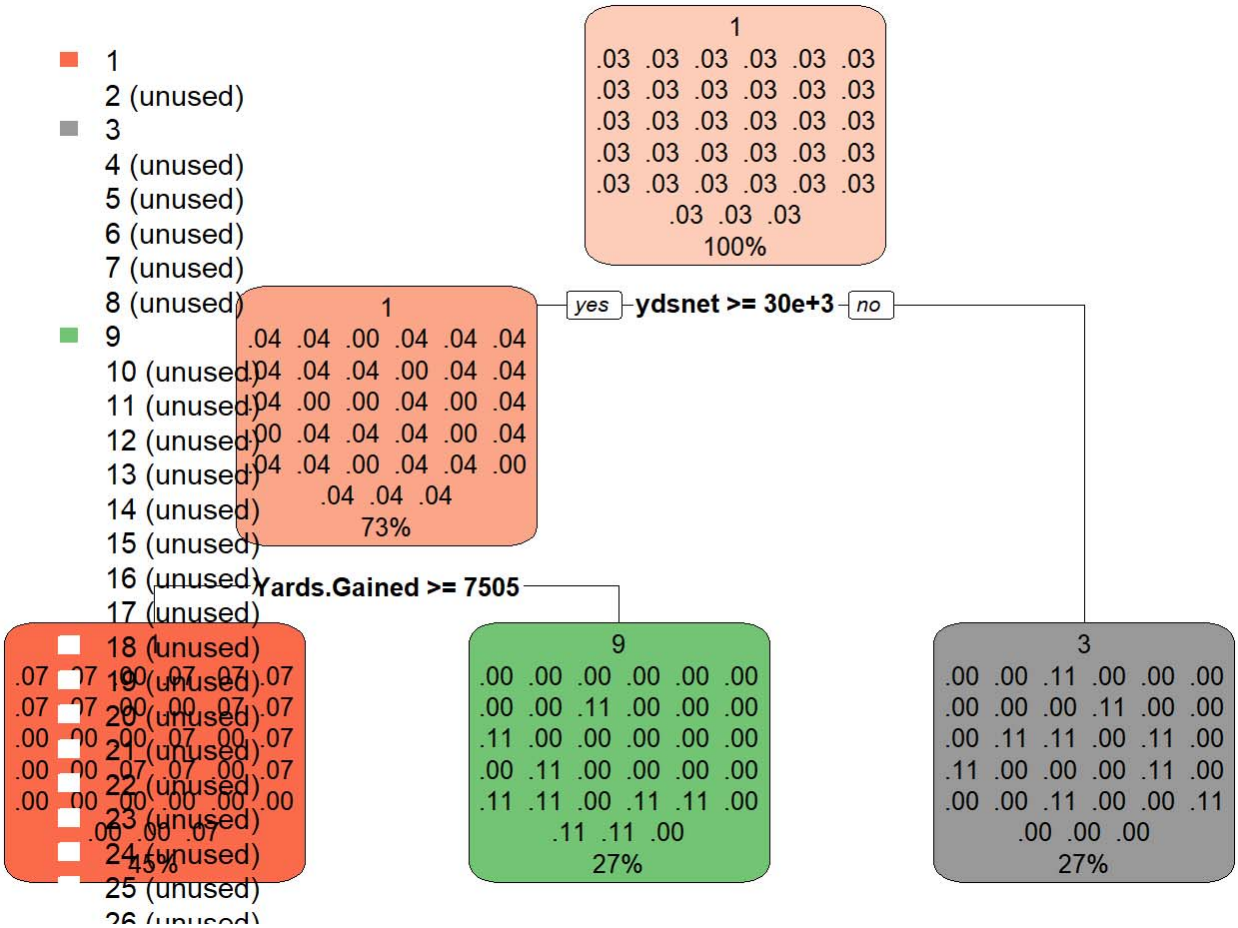
```
fit.default = rpart(

  H_Team ~ .,

  data = nfl_train,

  method = 'class'

)
head(fit.default)
```

```
## $frame
##            var  n wt dev yval complexity ncompete nsurrogate   yval2.V1
## 1       ydsnet 33 33  32    1    0.03125        4          5 1.00000000
## 2 Yards.Gained 24 24  23    1    0.03125        4          5 1.00000000
## 4       <leaf> 15 15  14    1    0.01000        0          0 1.00000000
## 5       <leaf>  9  9   8    9    0.01000        0          0 9.00000000
## 3       <leaf>  9  9   8    3    0.01000        0          0 3.00000000
##     yval2.V2   yval2.V3   yval2.V4   yval2.V5   yval2.V6   yval2.V7   yval2.V8
## 1 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 2 1.00000000 1.00000000 0.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 4 1.00000000 1.00000000 0.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 5 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 3 0.00000000 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##     yval2.V9  yval2.V10  yval2.V11  yval2.V12  yval2.V13  yval2.V14  yval2.V15
## 1 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 2 1.00000000 1.00000000 0.00000000 1.00000000 1.00000000 1.00000000 0.00000000
## 4 1.00000000 0.00000000 0.00000000 1.00000000 1.00000000 0.00000000 0.00000000
## 5 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 1.00000000 0.00000000
## 3 0.00000000 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 1.00000000
##    yval2.V16  yval2.V17  yval2.V18  yval2.V19  yval2.V20  yval2.V21  yval2.V22
## 1 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 2 0.00000000 1.00000000 0.00000000 1.00000000 0.00000000 1.00000000 1.00000000
## 4 0.00000000 1.00000000 0.00000000 1.00000000 0.00000000 0.00000000 1.00000000
## 5 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 1.00000000 0.00000000
## 3 1.00000000 0.00000000 1.00000000 0.00000000 1.00000000 0.00000000 0.00000000
##    yval2.V23  yval2.V24  yval2.V25  yval2.V26  yval2.V27  yval2.V28  yval2.V29
## 1 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## 2 1.00000000 0.00000000 1.00000000 1.00000000 1.00000000 0.00000000 1.00000000
## 4 1.00000000 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 5 0.00000000 0.00000000 0.00000000 1.00000000 1.00000000 0.00000000 1.00000000
## 3 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 1.00000000 0.00000000
##    yval2.V30  yval2.V31  yval2.V32  yval2.V33  yval2.V34  yval2.V35  yval2.V36
## 1 1.00000000 1.00000000 1.00000000 1.00000000 1.00000000 0.03030303 0.03030303
## 2 1.00000000 0.00000000 1.00000000 1.00000000 1.00000000 0.04166667 0.04166667
## 4 0.00000000 0.00000000 0.00000000 0.00000000 1.00000000 0.06666667 0.06666667
## 5 1.00000000 0.00000000 1.00000000 1.00000000 0.00000000 0.00000000 0.00000000
## 3 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##    yval2.V37  yval2.V38  yval2.V39  yval2.V40  yval2.V41  yval2.V42  yval2.V43
## 1 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303
## 2 0.00000000 0.04166667 0.04166667 0.04166667 0.04166667 0.04166667 0.04166667
## 4 0.00000000 0.06666667 0.06666667 0.06666667 0.06666667 0.06666667 0.00000000
## 5 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.11111111
## 3 0.11111111 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##    yval2.V44  yval2.V45  yval2.V46  yval2.V47  yval2.V48  yval2.V49  yval2.V50
## 1 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303
## 2 0.00000000 0.04166667 0.04166667 0.04166667 0.00000000 0.00000000 0.04166667
## 4 0.00000000 0.06666667 0.06666667 0.00000000 0.00000000 0.00000000 0.06666667
## 5 0.00000000 0.00000000 0.00000000 0.11111111 0.00000000 0.00000000 0.00000000
## 3 0.11111111 0.00000000 0.00000000 0.00000000 0.11111111 0.11111111 0.00000000
##    yval2.V51  yval2.V52  yval2.V53  yval2.V54  yval2.V55  yval2.V56  yval2.V57
## 1 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303
## 2 0.00000000 0.04166667 0.00000000 0.04166667 0.04166667 0.04166667 0.00000000
## 4 0.00000000 0.06666667 0.00000000 0.00000000 0.06666667 0.06666667 0.00000000
```

```
## 5 0.00000000 0.00000000 0.00000000 0.11111111 0.00000000 0.00000000 0.00000000
## 3 0.11111111 0.00000000 0.11111111 0.00000000 0.00000000 0.00000000 0.11111111
##    yval2.V58  yval2.V59  yval2.V60  yval2.V61  yval2.V62  yval2.V63  yval2.V64
## 1 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303 0.03030303
## 2 0.04166667 0.04166667 0.04166667 0.00000000 0.04166667 0.04166667 0.00000000
## 4 0.06666667 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 5 0.00000000 0.11111111 0.11111111 0.00000000 0.11111111 0.11111111 0.00000000
## 3 0.00000000 0.00000000 0.00000000 0.11111111 0.00000000 0.00000000 0.11111111
##    yval2.V65  yval2.V66  yval2.V67 yval2.nodeprob
## 1 0.03030303 0.03030303 0.03030303     1.00000000
## 2 0.04166667 0.04166667 0.04166667     0.72727273
## 4 0.00000000 0.00000000 0.06666667     0.45454545
## 5 0.11111111 0.11111111 0.00000000     0.27272727
## 3 0.00000000 0.00000000 0.00000000     0.27272727
##
## $where
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  3  3  5  3  3  3  3  3  4  5  3  3  4  5  5  3  5  3  5  4  3  3  5  3  4  4
## 27 28 29 30 31 32 33
##  5  4  4  5  4  4  3
##
## $call
## rpart(formula = H_Team ~ ., data = nfl_train, method = "class")
##
## $terms
## H_Team ~ ydsnet + Yards.Gained + Touchdown_Prob + Home_WP_post +
##     Away_WP_post + Win_Prob
## attr(,"variables")
## list(H_Team, ydsnet, Yards.Gained, Touchdown_Prob, Home_WP_post,
##     Away_WP_post, Win_Prob)
## attr(,"factors")
##                ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post
## H_Team              0            0              0            0            0
## ydsnet              1            0              0            0            0
## Yards.Gained        0            1              0            0            0
## Touchdown_Prob      0            0              1            0            0
## Home_WP_post        0            0              0            1            0
## Away_WP_post        0            0              0            0            1
## Win_Prob            0            0              0            0            0
##                Win_Prob
## H_Team                0
## ydsnet                0
## Yards.Gained          0
## Touchdown_Prob        0
## Home_WP_post          0
## Away_WP_post          0
## Win_Prob              1
## attr(,"term.labels")
## [1] "ydsnet"         "Yards.Gained"   "Touchdown_Prob" "Home_WP_post"
## [5] "Away_WP_post"   "Win_Prob"
## attr(,"order")
## [1] 1 1 1 1 1 1
## attr(,"intercept")
## [1] 1
```

```
## attr(,"response")
## [1] 1
## attr(,".Environment")
## <environment: R_GlobalEnv>
## attr(,"predvars")
## list(H_Team, ydsnet, Yards.Gained, Touchdown_Prob, Home_WP_post,
##     Away_WP_post, Win_Prob)
## attr(,"dataClasses")
##         H_Team         ydsnet   Yards.Gained Touchdown_Prob   Home_WP_post
##      "numeric"      "numeric"      "numeric"      "numeric"      "numeric"
##   Away_WP_post       Win_Prob
##      "numeric"      "numeric"
##
## $cptable
##         CP nsplit rel error  xerror xstd
## 1 0.03125      0    1.0000 1.03125    0
## 2 0.01000      2    0.9375 1.03125    0
##
## $method
## [1] "class"
```

```
rpart.plot(fit.default)
```




```
printcp(fit.default)
```

```
##
## Classification tree:
## rpart(formula = H_Team ~ ., data = nfl_train, method = "class")
##
## Variables actually used in tree construction:
## [1] Yards.Gained ydsnet
##
## Root node error: 32/33 = 0.9697
##
## n= 33
##
##        CP nsplit rel error xerror xstd
## 1 0.03125      0    1.0000 1.0312    0
## 2 0.01000      2    0.9375 1.0312    0
```

```
fit.default.pred = table(predict(fit.default, type='class'), nfl_train$H_Team)

1-sum(diag(fit.default.pred))/sum(fit.default.pred)
```

```
## [1] 0.9090909
```

```
predict(fit.default, nfl_test, type = 'prob')
```

```
##             1          2         3          4          5          6          7
## 1  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 2  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 3  0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 4  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 5  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 6  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 7  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 8  0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 9  0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 10 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 11 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 12 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 13 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 14 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 15 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 16 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 17 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 18 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 19 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 20 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 21 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 22 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 23 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 24 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
## 25 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 26 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 27 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 28 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 29 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 30 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 31 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 32 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000 0.00000000
## 33 0.06666667 0.06666667 0.0000000 0.06666667 0.06666667 0.06666667 0.06666667
##             8         9        10         11         12        13        14
## 1  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 2  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 3  0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 4  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 5  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 6  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 7  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 8  0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 9  0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 10 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 11 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 12 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 13 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 14 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 15 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 16 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 17 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 18 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
```

```
## 19 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 20 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 21 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 22 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 23 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 24 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
## 25 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 26 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 27 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 28 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 29 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 30 0.00000000 0.0000000 0.1111111 0.00000000 0.00000000 0.0000000 0.1111111
## 31 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 32 0.00000000 0.1111111 0.0000000 0.00000000 0.00000000 0.1111111 0.0000000
## 33 0.06666667 0.0000000 0.0000000 0.06666667 0.06666667 0.0000000 0.0000000
##           15         16        17         18        19        20         21
## 1  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 2  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 3  0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 4  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 5  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 6  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 7  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 8  0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 9  0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 10 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 11 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 12 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 13 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 14 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 15 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 16 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 17 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 18 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 19 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 20 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 21 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 22 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 23 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 24 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
## 25 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 26 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 27 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 28 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 29 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 30 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.0000000 0.00000000
## 31 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 32 0.0000000 0.00000000 0.0000000 0.00000000 0.0000000 0.1111111 0.00000000
## 33 0.0000000 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.06666667
##           22        23         24        25        26        27        28
## 1  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 2  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 3  0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 4  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
```

```
## 5  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 6  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 7  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 8  0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 9  0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 10 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 11 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 12 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 13 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 14 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 15 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 16 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 17 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 18 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 19 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 20 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 21 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 22 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 23 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 24 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
## 25 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 26 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 27 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 28 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 29 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 30 0.00000000 0.1111111 0.00000000 0.0000000 0.0000000 0.1111111 0.0000000
## 31 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 32 0.00000000 0.0000000 0.00000000 0.1111111 0.1111111 0.0000000 0.1111111
## 33 0.06666667 0.0000000 0.06666667 0.0000000 0.0000000 0.0000000 0.0000000
##           29        31        32        33         34
## 1  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 2  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 3  0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 4  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 5  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 6  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 7  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 8  0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 9  0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 10 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 11 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 12 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 13 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 14 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 15 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 16 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 17 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 18 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 19 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 20 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 21 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 22 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
## 23 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 24 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
```

```
## 25 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 26 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 27 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 28 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 29 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 30 0.0000000 0.1111111 0.0000000 0.0000000 0.00000000
## 31 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 32 0.1111111 0.0000000 0.1111111 0.1111111 0.00000000
## 33 0.0000000 0.0000000 0.0000000 0.0000000 0.06666667
```

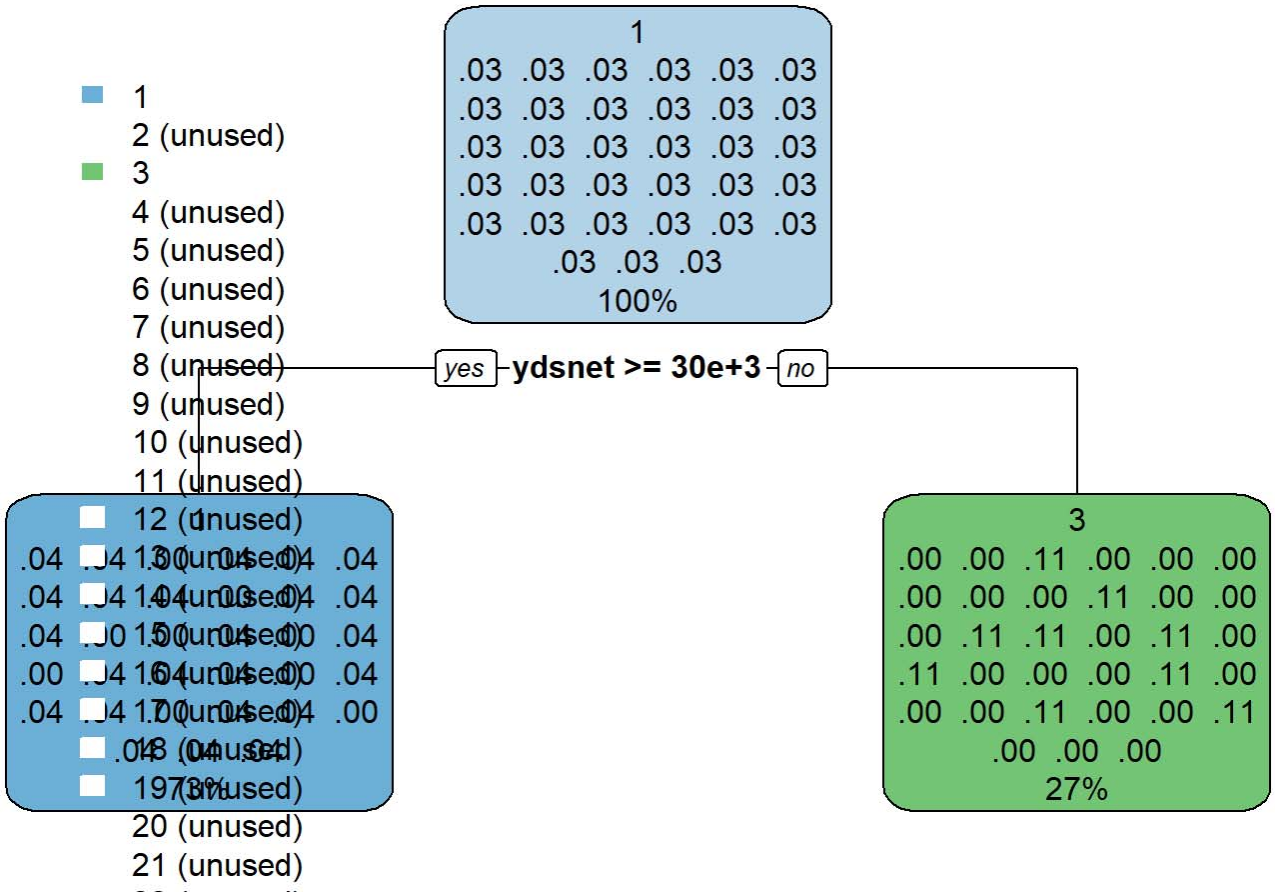
##Data showed Jay wrote the third paper.

```
predict(fit.default, nfl_test, type = 'class')
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  1  1  3  1  1  1  1  1  9  3  1  1  9  3  3  1  3  1  3  9  1  1  3  1  9  9
## 27 28 29 30 31 32 33
##  3  9  9  3  9  9  1
## 33 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 34
```

```
fit.tuned = rpart(

  H_Team ~ .,

  data = nfl_train,

  method = 'class',

  control = list(maxdepth = 1)
)
```

```
rpart.plot(fit.tuned)
```




```
printcp(fit.tuned)
```

```
##
## Classification tree:
## rpart(formula = H_Team ~ ., data = nfl_train, method = "class",
##     control = list(maxdepth = 1))
##
## Variables actually used in tree construction:
## [1] ydsnet
##
## Root node error: 32/33 = 0.9697
##
## n= 33
##
##         CP nsplit rel error xerror xstd
## 1 0.03125      0   1.00000 1.0312    0
## 2 0.01000      1   0.96875 1.0312    0
```

tuned tree summary

```
printcp(fit.tuned)
```

```
##
## Classification tree:
## rpart(formula = H_Team ~ ., data = nfl_train, method = "class",
##     control = list(maxdepth = 1))
##
## Variables actually used in tree construction:
## [1] ydsnet
##
## Root node error: 32/33 = 0.9697
##
## n= 33
##
##        CP nsplit rel error xerror xstd
## 1 0.03125      0   1.00000 1.0312    0
## 2 0.01000      1   0.96875 1.0312    0
```

##Result Root node error is 0.31081 To measure the error rate

```
##Determine the predictive performance
fit.tuned.pred = table(predict(fit.tuned, type='class'), nfl_train$H_Team)

1-sum(diag(fit.tuned.pred))/sum(fit.tuned.pred)
```

```
## [1] 0.9393939
```

```
fit.tuned.pred
```

```
##
##      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
##   1  1 1 0 1 1 1 1 1 1  0  1  1  1  0  0  1  0  1  0  1  1  1  0  1  1  1  0  1
##   2  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   3  0 0 1 0 0 0 0 0 0  1  0  0  0  1  1  0  1  0  1  0  0  0  1  0  0  0  1  0
##   4  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   5  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   6  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   7  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   8  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   9  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   10 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   11 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   12 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   13 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   15 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   16 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   17 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   18 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   19 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   20 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   21 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   22 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   23 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   24 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   25 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   26 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   27 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   28 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   29 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   31 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   32 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   33 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   34 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
##      29 31 32 33 34
##   1   1  0  1  1  1
##   2   0  0  0  0  0
##   3   0  1  0  0  0
##   4   0  0  0  0  0
##   5   0  0  0  0  0
##   6   0  0  0  0  0
##   7   0  0  0  0  0
##   8   0  0  0  0  0
##   9   0  0  0  0  0
##   10  0  0  0  0  0
##   11  0  0  0  0  0
##   12  0  0  0  0  0
##   13  0  0  0  0  0
##   14  0  0  0  0  0
##   15  0  0  0  0  0
##   16  0  0  0  0  0
```

```
##   17  0  0  0  0  0
##   18  0  0  0  0  0
##   19  0  0  0  0  0
##   20  0  0  0  0  0
##   21  0  0  0  0  0
##   22  0  0  0  0  0
##   23  0  0  0  0  0
##   24  0  0  0  0  0
##   25  0  0  0  0  0
##   26  0  0  0  0  0
##   27  0  0  0  0  0
##   28  0  0  0  0  0
##   29  0  0  0  0  0
##   31  0  0  0  0  0
##   32  0  0  0  0  0
##   33  0  0  0  0  0
##   34  0  0  0  0  0
```

##Result predictive performance

Cross Validation

```
xpred.rpart(fit.tuned, xval=5)
```

```
##    0.51562500 0.01767767
## 1           2          2
## 2           1          1
## 3           1         10
## 4           1          1
## 5           2          2
## 6           1          1
## 7           1          1
## 8           1          1
## 9           2          2
## 10          1          3
## 11          1          1
## 12          1          1
## 13          1          1
## 14          1          3
## 15          1          3
## 16          1          1
## 17          1          3
## 18          2          2
## 19          2          3
## 20          1          1
## 21          1          1
## 22          1          1
## 23          1          3
## 24          1          1
## 25          1          3
## 26          1          3
## 27          1          3
## 28          1         10
## 29          1          1
## 30          1          3
## 31          1          3
## 32          2          2
## 33          1          1
```

```
predict(fit.tuned, nfl_test, type = 'prob')
```

```
##             1          2         3          4          5          6          7
## 1  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 2  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 3  0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 4  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 5  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 6  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 7  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 8  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 9  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 10 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 11 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 12 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 13 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 14 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 15 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 16 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 17 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 18 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 19 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 20 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 21 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 22 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 23 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 24 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 25 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 26 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 27 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 28 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 29 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 30 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.00000000
## 31 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 32 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
## 33 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.04166667
##             8          9         10         11         12         13         14
## 1  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 2  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 3  0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 4  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 5  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 6  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 7  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 8  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 9  0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 10 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 11 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 12 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 13 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 14 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 15 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 16 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 17 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 18 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
```

```
## 19 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 20 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 21 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 22 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 23 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 24 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 25 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 26 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 27 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 28 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 29 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 30 0.00000000 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111
## 31 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 32 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
## 33 0.04166667 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000
##           15         16        17         18        19         20         21
## 1  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 2  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 3  0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 4  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 5  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 6  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 7  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 8  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 9  0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 10 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 11 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 12 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 13 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 14 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 15 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 16 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 17 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 18 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 19 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 20 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 21 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 22 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 23 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 24 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 25 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 26 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 27 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 28 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 29 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 30 0.1111111 0.00000000 0.1111111 0.00000000 0.1111111 0.00000000 0.00000000
## 31 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 32 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
## 33 0.0000000 0.04166667 0.0000000 0.04166667 0.0000000 0.04166667 0.04166667
##           22         23         24         25         26         27         28
## 1  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 2  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 3  0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 4  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
```

```
## 5  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 6  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 7  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 8  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 9  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 10 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 11 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 12 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 13 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 14 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 15 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 16 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 17 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 18 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 19 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 20 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 21 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 22 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 23 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 24 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 25 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 26 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 27 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 28 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 29 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 30 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000 0.1111111 0.00000000
## 31 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 32 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
## 33 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667 0.0000000 0.04166667
##            29        31         32         33         34
## 1  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 2  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 3  0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 4  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 5  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 6  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 7  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 8  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 9  0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 10 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 11 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 12 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 13 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 14 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 15 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 16 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 17 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 18 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 19 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 20 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 21 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 22 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 23 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 24 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
```

```
## 25 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 26 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 27 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 28 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 29 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 30 0.00000000 0.1111111 0.00000000 0.00000000 0.00000000
## 31 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 32 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
## 33 0.04166667 0.0000000 0.04166667 0.04166667 0.04166667
```

```
predict(fit.tuned, nfl_test, type = 'class')
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  1  1  3  1  1  1  1  1  1  3  1  1  1  3  3  1  3  1  3  1  1  1  3  1  1  1
## 27 28 29 30 31 32 33
##  3  1  1  3  1  1  1
## 33 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 34
```

##Leniar Model and randomForest

```
Winning_Prob
```

```
## # A tibble: 39,375 x 7
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
##     <dbl>  <dbl>        <dbl>          <dbl>        <dbl>        <dbl>    <dbl>
##  1      4      0            0          0.314        0.514        0.486    0.486
##  2      4      0            0          0.314        0.534        0.466    0.486
##  3      4      0            0          0.267        0.563        0.437    0.466
##  4      4     -5           11          0.219        0.581        0.419    0.437
##  5      4      1            6          0.181        0.597        0.403    0.419
##  6      4      1           16          0.171        0.560        0.440    0.403
##  7      4      0            0          0.437        0.540        0.460    0.560
##  8      4      0            0          0.309        0.466        0.534    0.513
##  9      4      0            0          0.199        0.533        0.467    0.466
## 10      4     11           13          0.202        0.498        0.502    0.438
## # ... with 39,365 more rows
```

Split the data

```
library(tidymodels)
set.seed(1234)
data_split <- Winning_Prob %>%
  initial_split(strata = H_Team)
```

```
nfl_train <- training(data_split)
nfl_train
```

```
## # A tibble: 29,533 x 7
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
##     <dbl>  <dbl>        <dbl>          <dbl>        <dbl>        <dbl>    <dbl>
##  1      4      0            0          0.314        0.514        0.486    0.486
##  2      4      0            0          0.314        0.534        0.466    0.486
##  3      4      0            0          0.267        0.563        0.437    0.466
##  4      4     -5           11          0.219        0.581        0.419    0.437
##  5      4      1           16          0.171        0.560        0.440    0.403
##  6      4      0            0          0.437        0.540        0.460    0.560
##  7      4      0            0          0.309        0.466        0.534    0.513
##  8      4     11           13          0.202        0.498        0.502    0.438
##  9      4     12            1          0.323        0.515        0.485    0.502
## 10      4     15            3          0.285        0.527        0.473    0.485
## # ... with 29,523 more rows
```

```
nfl_test <- testing(data_split)
nfl_test
```

```
## # A tibble: 9,842 x 7
##    H_Team ydsnet Yards.Gained Touchdown_Prob Home_WP_post Away_WP_post Win_Prob
##     <dbl>  <dbl>        <dbl>          <dbl>        <dbl>        <dbl>    <dbl>
##  1      4      1            6          0.181        0.597        0.403    0.419
##  2      4      0            0          0.199        0.533        0.467    0.466
##  3      4      0            0          0.391        0.520        0.480    0.542
##  4      4      0            0          0.340        0.492        0.508    0.520
##  5      4      0            0          0.279        0.446        0.554    0.492
##  6      4     25            4          0.526        0.624        0.376    0.628
##  7      4     25            0          0            0.728        0.272    0.732
##  8      4      0            0          0.320        0.728        0.272    0.272
##  9      4      1            0          0.177        0.776        0.224    0.190
## 10      4     20            2          0.453        0.804        0.196    0.805
## # ... with 9,832 more rows
```

##Linearr Model Training data

```
Len_Model<- linear_reg() %>%
  set_engine(engine = 'lm')
```

```
Len_fit<- Len_Model %>%
  fit(H_Team~.,
    data=nfl_train)

Len_fit
```

```
## parsnip model object
##
## Fit time:  21ms
##
## Call:
## stats::lm(formula = formula, data = data)
##
## Coefficients:
##    (Intercept)          ydsnet    Yards.Gained  Touchdown_Prob    Home_WP_post
##      17.607749        0.000221       -0.010985        0.154356       -1.490772
##   Away_WP_post        Win_Prob
##             NA        0.013189
```

##Leniar Training new data result

```
trn.results<- Len_fit %>%
  predict(new_data=nfl_train) %>%
  mutate(truth=nfl_train$Win_Prob)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
trn.results
```

```
## # A tibble: 29,533 x 2
##    .pred truth
##    <dbl> <dbl>
##  1  16.9 0.486
##  2  16.9 0.486
##  3  16.8 0.466
##  4  16.7 0.437
##  5  16.6 0.403
##  6  16.9 0.560
##  7  17.0 0.513
##  8  16.8 0.438
##  9  16.9 0.502
## 10  16.8 0.485
## # ... with 29,523 more rows
```

##Leniar Training model

```
test.results<- Len_fit %>%
  predict(new_data=nfl_test) %>%
  mutate(truth=nfl_test$Win_Prob)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
test.results
```

```
## # A tibble: 9,842 x 2
##    .pred truth
##    <dbl> <dbl>
##  1  16.7 0.419
##  2  16.8 0.466
##  3  16.9 0.542
##  4  16.9 0.520
##  5  17.0 0.492
##  6  16.7 0.628
##  7  16.5 0.732
##  8  16.6 0.272
##  9  16.5 0.190
## 10  16.5 0.805
## # ... with 9,832 more rows
```

Random Forest Training data

```
rf_spec <- rand_forest(mode = "regression") %>%
  set_engine("ranger")
```

```
rf_fit <- rf_spec %>%
  fit(H_Team ~ .,
    data = nfl_train
  )

rf_fit
```

```
## parsnip model object
##
## Fit time:  56.1s
## Ranger result
##
## Call:
##  ranger::ranger(formula = formula, data = data, num.threads = 1,      verbose = FALSE, seed =
sample.int(10^5, 1))
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      29533
## Number of independent variables:  6
## Mtry:                             2
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       99.83063
## R squared (OOB):                  -0.01986728
```

##Random Forest Training model

```
trn_model<- rf_fit %>%
  predict(new_data=nfl_train) %>%
  mutate(truth=nfl_train$Win_Prob)

trn_model
```

```
## # A tibble: 29,533 x 2
##    .pred truth
##    <dbl> <dbl>
##  1  8.96 0.486
##  2 10.6  0.486
##  3 12.0  0.466
##  4  8.94 0.437
##  5  8.54 0.403
##  6  8.96 0.560
##  7  9.02 0.513
##  8  9.16 0.438
##  9  8.09 0.502
## 10  8.42 0.485
## # ... with 29,523 more rows
```

##Cbind the leniar model and randomForest model for new data

```
trn_model<- Len_fit %>%
  predict(new_data=nfl_train) %>%
  mutate(truth=nfl_train$Win_Prob,
         model="lm") %>%
  bind_rows(rf_fit %>%
              predict(new_data=nfl_train) %>%
              mutate(truth=nfl_train$Win_Prob,
                     model="rf"))
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
trn_model
```

```
## # A tibble: 59,066 x 3
##    .pred truth model
##    <dbl> <dbl> <chr>
##  1  16.9 0.486 lm
##  2  16.9 0.486 lm
##  3  16.8 0.466 lm
##  4  16.7 0.437 lm
##  5  16.6 0.403 lm
##  6  16.9 0.560 lm
##  7  17.0 0.513 lm
##  8  16.8 0.438 lm
##  9  16.9 0.502 lm
## 10  16.8 0.485 lm
## # ... with 59,056 more rows
```

##Random Forest Test model

```
test.results<- rf_fit %>%
  predict(new_data=nfl_test) %>%
  mutate(truth=nfl_test$Win_Prob)

test.results
```

```
## # A tibble: 9,842 x 2
##    .pred truth
##    <dbl> <dbl>
##  1  11.4 0.419
##  2  12.3 0.466
##  3  18.2 0.542
##  4  15.8 0.520
##  5  19.6 0.492
##  6  15.0 0.628
##  7  16.5 0.732
##  8  14.0 0.272
##  9  15.3 0.190
## 10  14.4 0.805
## # ... with 9,832 more rows
```

##Mutate Liniar model and randomforest model

```
test.results<- Len_fit %>%
  predict(new_data=nfl_test) %>%
  mutate(truth=nfl_test$Win_Prob,
         model="lm") %>%
  bind_rows(rf_fit %>%
              predict(new_data=nfl_test) %>%
              mutate(truth=nfl_test$Win_Prob,
                     model="rf"))
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
test.results
```

```
## # A tibble: 19,684 x 3
##    .pred truth model
##    <dbl> <dbl> <chr>
##  1  16.7 0.419 lm
##  2  16.8 0.466 lm
##  3  16.9 0.542 lm
##  4  16.9 0.520 lm
##  5  17.0 0.492 lm
##  6  16.7 0.628 lm
##  7  16.5 0.732 lm
##  8  16.6 0.272 lm
##  9  16.5 0.190 lm
## 10  16.5 0.805 lm
## # ... with 19,674 more rows
```

##For this regression model, let's look at the rmse for what we've done so far.

```
trn_model %>%
  group_by(model) %>%
  rmse(truth = truth, estimate = .pred)
```

```
## # A tibble: 2 x 4
##   model .metric .estimator .estimate
##   <chr> <chr>   <chr>          <dbl>
## 1 lm    rmse    standard        16.3
## 2 rf    rmse    standard        17.2
```

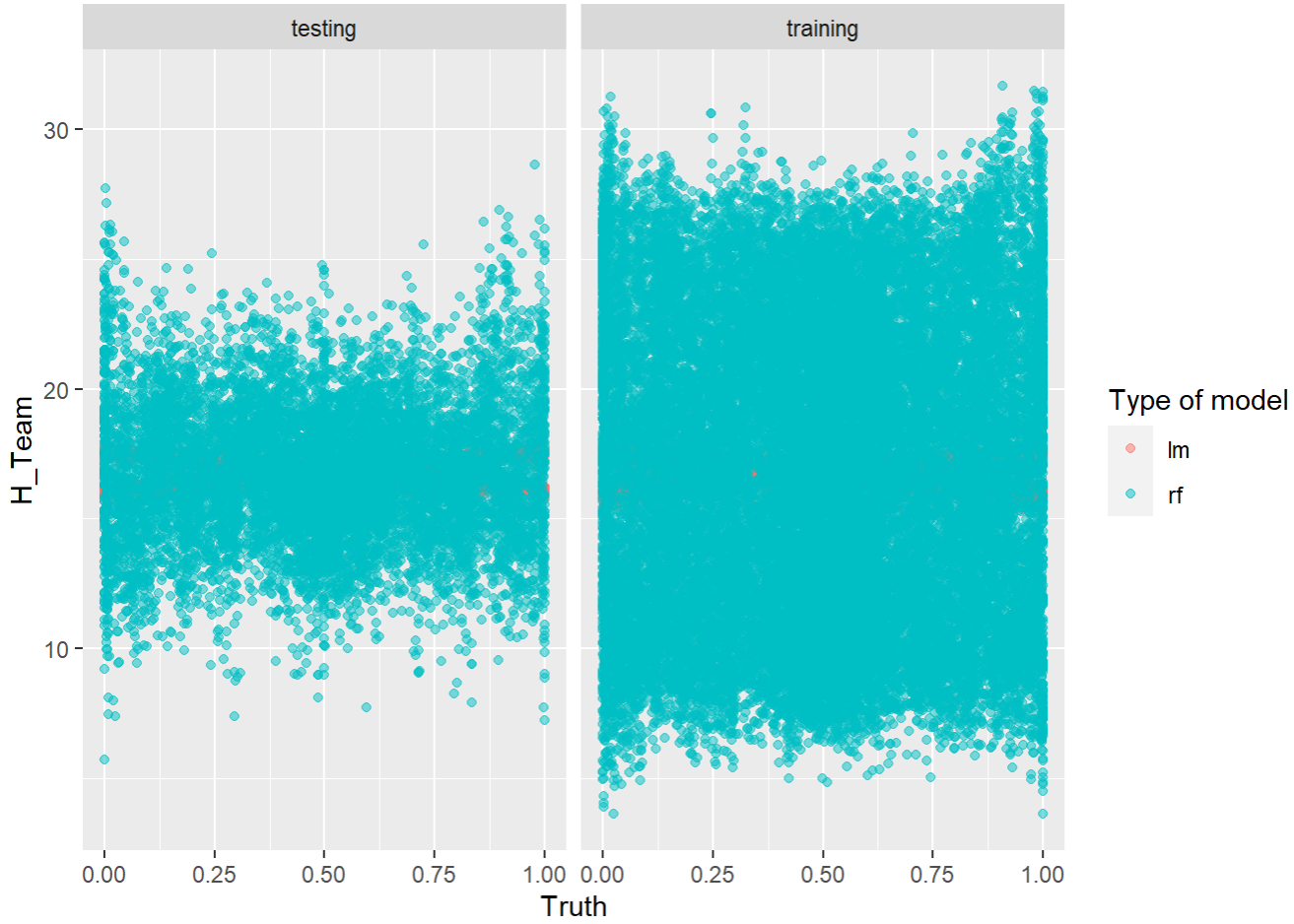
##Result on Test data is a lot lower that the train data

```
test.results %>%
  group_by(model) %>%
  rmse(truth = truth, estimate = .pred)
```

```
## # A tibble: 2 x 4
##   model .metric .estimator .estimate
##   <chr> <chr>   <chr>          <dbl>
## 1 lm    rmse    standard        16.3
## 2 rf    rmse    standard        16.5
```

##Looking at the visualization results

```
test.results %>%
  mutate(train = "testing") %>%
  bind_rows(trn_model %>%
    mutate(train = "training")) %>%
  ggplot(aes(truth, .pred, color = model)) +
  geom_abline(lty = 2, color = "gray80", size = 1.5) +
  geom_point(alpha = 0.5) +
  facet_wrap(~train) +
  labs(
    x = "Truth",
    y = "H_Team",
    color = "Type of model"
  )
```



##The result above is not so great so we are looking into the 10 fold

##Looking at the 10 fold default on tidymodel

```
set.seed(1234)
nfl_folds <- vfold_cv(nfl_train, strata = H_Team)
nfl_folds
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits             id
##    <named list>       <chr>
##  1 <split [26.6K/3K]> Fold01
##  2 <split [26.6K/3K]> Fold02
##  3 <split [26.6K/3K]> Fold03
##  4 <split [26.6K/3K]> Fold04
##  5 <split [26.6K/3K]> Fold05
##  6 <split [26.6K/3K]> Fold06
##  7 <split [26.6K/3K]> Fold07
##  8 <split [26.6K/3K]> Fold08
##  9 <split [26.6K/3K]> Fold09
## 10 <split [26.6K/3K]> Fold10
```

```
fit_resamples(
  H_Team ~ .,
  rf_spec,nfl_folds,
  control = control_resamples(save_pred = TRUE)
)
```

```
## Warning: `fit_resamples.formula()` is deprecated as of lifecycle 0.1.0.
## The first argument to `fit_resamples()` should be either a model or a workflow. In the futur
e, you can use:
## fit_resamples(rf_spec, H_Team ~ ., resamples = nfl_folds, control = control_resamples(save_pr
ed = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

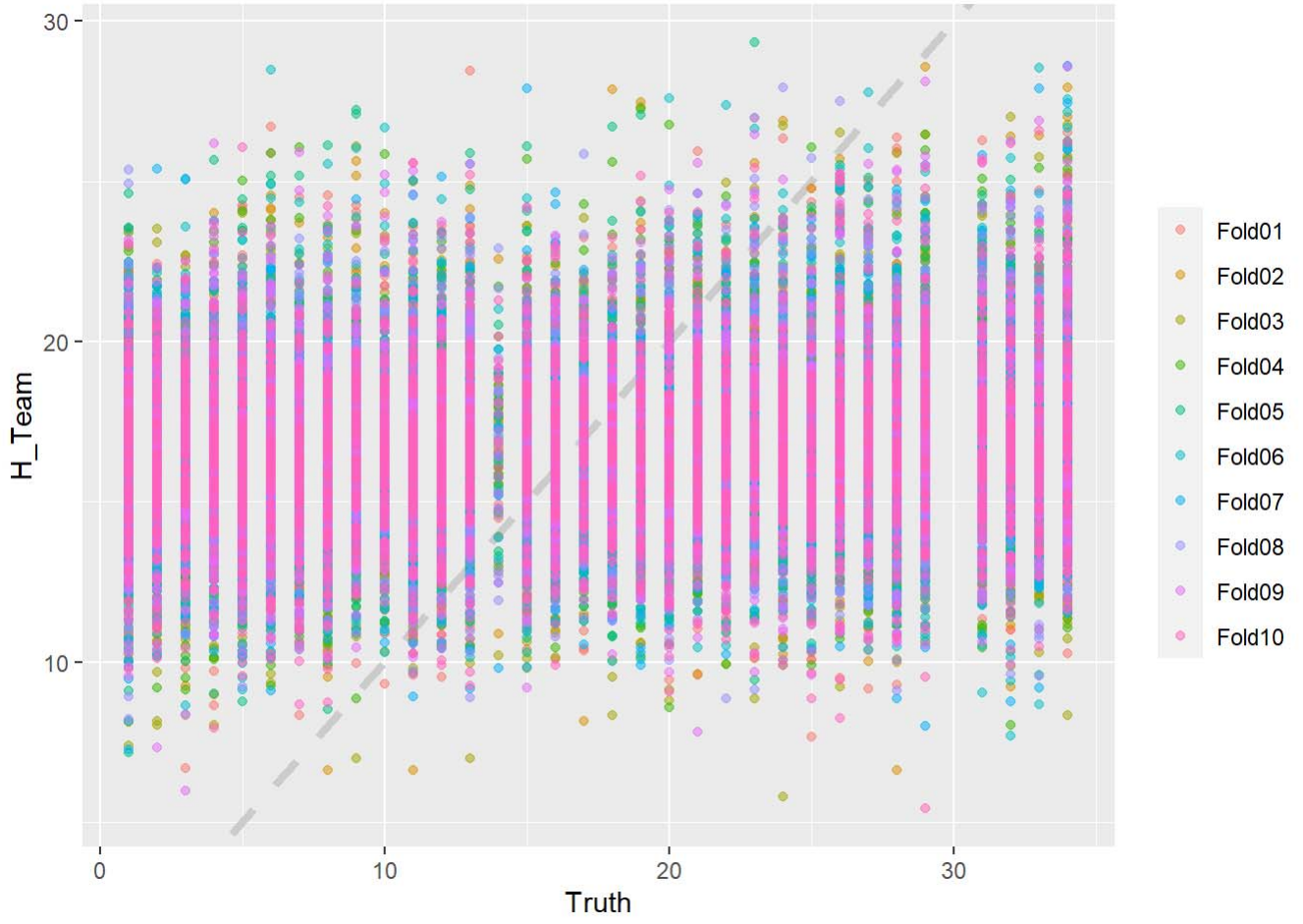
```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 5
##    splits             id     .metrics        .notes          .predictions
##    <list>             <chr>  <list>          <list>          <list>
##  1 <split [26.6K/3K]> Fold01 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,955 x 3~
##  2 <split [26.6K/3K]> Fold02 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,955 x 3~
##  3 <split [26.6K/3K]> Fold03 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,955 x 3~
##  4 <split [26.6K/3K]> Fold04 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,954 x 3~
##  5 <split [26.6K/3K]> Fold05 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,954 x 3~
##  6 <split [26.6K/3K]> Fold06 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,953 x 3~
##  7 <split [26.6K/3K]> Fold07 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,952 x 3~
##  8 <split [26.6K/3K]> Fold08 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,952 x 3~
##  9 <split [26.6K/3K]> Fold09 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,952 x 3~
## 10 <split [26.6K/3K]> Fold10 <tibble [2 x 3~ <tibble [0 x 1~ <tibble [2,951 x 3~
```

```
rf_res <- fit_resamples(
  H_Team ~ .,
  rf_spec,nfl_folds,
  control = control_resamples(save_pred = TRUE)
)
```

```
rf_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 5
##   .metric .estimator   mean     n  std_err
##   <chr>   <chr>       <dbl> <int>    <dbl>
## 1 rmse    standard   9.98      10 0.0150
## 2 rsq     standard   0.0112    10 0.000959
```

```
rf_res %>%
  unnest(.predictions) %>%
  ggplot(aes(H_Team, .pred, color = id)) +
  geom_abline(lty = 2, color = "gray80", size = 1.5) +
  geom_point(alpha = 0.5) +
  labs(
    x = "Truth",
    y = "H_Team",
    color = NULL
  )
```



looking at the 10 fold leniar

```
set.seed(1234)
nfl_folds1 <- vfold_cv(nfl_train, strata = H_Team)
nfl_folds1
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits             id
##    <named list>       <chr>
##  1 <split [26.6K/3K]> Fold01
##  2 <split [26.6K/3K]> Fold02
##  3 <split [26.6K/3K]> Fold03
##  4 <split [26.6K/3K]> Fold04
##  5 <split [26.6K/3K]> Fold05
##  6 <split [26.6K/3K]> Fold06
##  7 <split [26.6K/3K]> Fold07
##  8 <split [26.6K/3K]> Fold08
##  9 <split [26.6K/3K]> Fold09
## 10 <split [26.6K/3K]> Fold10
```

```
fit_resamples(
  H_Team ~ .,
  Len_Model,nfl_folds1,
  control1 = control_resamples(save_pred = TRUE)
)
```

```
## Warning: `fit_resamples.formula()` is deprecated as of lifecycle 0.1.0.
## The first argument to `fit_resamples()` should be either a model or a workflow. In the futur
e, you can use:
## fit_resamples(Len_Model, H_Team ~ ., resamples = nfl_folds1,
##     control1 = control_resamples(save_pred = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## Warning: The `...` are not used in this function but one or more objects were
## passed: 'control1'
```

```
## ! Fold01: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold02: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold03: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold04: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold05: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold06: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold07: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold08: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold09: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold10: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 4
##    splits             id     .metrics         .notes
##    <list>             <chr>  <list>           <list>
##  1 <split [26.6K/3K]> Fold01 <tibble [2 x 3]> <tibble [1 x 1]>
##  2 <split [26.6K/3K]> Fold02 <tibble [2 x 3]> <tibble [1 x 1]>
##  3 <split [26.6K/3K]> Fold03 <tibble [2 x 3]> <tibble [1 x 1]>
##  4 <split [26.6K/3K]> Fold04 <tibble [2 x 3]> <tibble [1 x 1]>
##  5 <split [26.6K/3K]> Fold05 <tibble [2 x 3]> <tibble [1 x 1]>
##  6 <split [26.6K/3K]> Fold06 <tibble [2 x 3]> <tibble [1 x 1]>
##  7 <split [26.6K/3K]> Fold07 <tibble [2 x 3]> <tibble [1 x 1]>
##  8 <split [26.6K/3K]> Fold08 <tibble [2 x 3]> <tibble [1 x 1]>
##  9 <split [26.6K/3K]> Fold09 <tibble [2 x 3]> <tibble [1 x 1]>
## 10 <split [26.6K/3K]> Fold10 <tibble [2 x 3]> <tibble [1 x 1]>
```

```
rf_res1 <- fit_resamples(
  H_Team ~ .,
  Len_Model,nfl_folds1,
  control1 = control_resamples(save_pred = TRUE)
)
```

```
## Warning: The `...` are not used in this function but one or more objects were
## passed: 'control1'
```

```
## ! Fold01: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold02: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold03: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold04: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold05: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold06: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold07: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold08: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold09: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
## ! Fold10: model (predictions): prediction from a rank-deficient fit may be misleading
```

```
rf_res1 %>%
  collect_metrics()
```

```
## # A tibble: 2 x 5
##   .metric .estimator    mean     n  std_err
##   <chr>   <chr>        <dbl> <int>    <dbl>
## 1 rmse    standard   9.89       10 0.0115
## 2 rsq     standard   0.00180    10 0.000455
```