NFL Yardage Predictor Report

Jake Berberian, Bilal Gilani, JP Zamanillo

4/28/2020

Intro

The average length of a NFL head coach's career is about 4.3 years which is actually longer than the average careers of coaches in other professional sports leagues such as the NBA, MLB, NHL, and the English Premier League. The NFL head coaches are responsible for their respective team's play calls during a game. As with any responsibility, comes criticism if such play calls go awry. Depending on the situation, a coach must decide on what kind of play to run, and this was our initial research question. Can we create a model that suggests what kind of play to use that is dependent on the situation?

Roles

For the project, the following tasks were completed by the following people: - **Initial data review**: All three, each taking 85 columns from the original data set to explore. - **Model**: JP wrote the randomForest model, while Jake wrote the two quick linear models. - **Shiny app**: Bilal wrote the Shiny app, with Jake touching up the presentation of the app. - **Report**: Written by Jake.

Data Review

This data was is from Kaggle and stored on GitHub in a truncated version. The edited version only includes six of the 185 variables of the original data set. This is primarily due to the size of the data set and uploading to Git Hub (for reproducibility) and to shinyapps.io (for their size requirement). The data contains detailed play-by-play from the 2009 through 2018 NFL regular seasons.

```r
nfl <- suppressMessages(
  read_csv(
    "https://raw.githubusercontent.com/jakeberberian/STAT-613/master/nfl_play.csv"))
```

```r
glimpse(nfl)
```

```
## Observations: 449,371
## Variables: 6
## $ down          <dbl> NA, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 1, 2, 3, 4, 1, 1, 1...
## $ yards_gained  <dbl> 0, 5, -3, 0, 0, 0, 4, -2, 0, 3, 10, -1, 9, -19, 0, 32...
## $ play_type     <chr> "kickoff", "pass", "run", "pass", "punt", "run", "pas...
## $ ydstogo       <dbl> 0, 10, 5, 8, 8, 10, 10, 6, 8, 10, 7, 10, 11, 2, 21, 1...
## $ side_of_field <chr> "TEN", "PIT", "PIT", "PIT", "PIT", "TEN", "TEN", "TEN...
## $ defteam       <chr> "TEN", "TEN", "TEN", "TEN", "TEN", "PIT", "PIT", "PIT...
```

As we see there are 449371 observations of 6 variables. These columns are self-explanatory, sans *yds_to_go*, which indicates yards to first down. Our data contains 94528 NAs, which will need to be cleaned up. As prior mentioned, some of the data clean included truncating the data and this was done before. However, we will want to take care of the NAs and create factors. We also will remove "explosive plays" which we define as a run play over 15 yards and a pass play over 35 yards. Lots of sports websites have different opinions about what amounts to an "explosive play," but we chose 35 and 15 for pass and run, respectively. This will tune our model to the typical yardage gain and therefore won't be skewed in the direction of these larger plays. Effectively, we're removing outliers. We also don't need any other plays besides runs and passes, so those will be removed as well. Finally, since some teams have changed their location (i.e. the St. Louis Rams moving to Los Angeles), we'll correct these by updating all instances to their most current three-letter abbreviations.

```r
nfl1 <- nfl %>%
  filter(play_type == "pass" & yards_gained < 35 |
           play_type == "run" & yards_gained < 15) %>%
  drop_na() %>%
  mutate(side_of_field = case_when(defteam == side_of_field ~ "opposing",
                                   TRUE ~ "own")) %>%
  mutate(defteam = str_replace_all(defteam, "STL", "LA"),
         defteam = str_replace_all(defteam, "^LA$", "LAR"),
         defteam = str_replace_all(defteam, "JAC", "JAX"),
         defteam = str_replace_all(defteam, "SD", "LAC"))%>%
  mutate(down = factor(down),
         play_type = factor(play_type),
         side_of_field = factor(side_of_field),
         defteam = factor(defteam))

glimpse(nfl1)
```

```
## Observations: 307,708
## Variables: 6
## $ down          <fct> 1, 2, 3, 1, 2, 3, 1, 2, 1, 2, 3, 1, 1, 2, 3, 1, 2, 3,...
## $ yards_gained  <dbl> 5, -3, 0, 0, 4, -2, 3, 10, -1, 9, -19, 20, 3, 0, 0, 1...
## $ play_type     <fct> pass, run, pass, run, pass, run, pass, pass, run, pas...
## $ ydstogo       <dbl> 10, 5, 8, 10, 10, 6, 10, 7, 10, 11, 2, 10, 10, 7, 7, ...
## $ side_of_field <fct> own, own, own, own, own, own, opposing, opposing, opp...
## $ defteam       <fct> TEN, TEN, TEN, PIT, PIT, PIT, TEN, TEN, TEN, TEN, TEN...
```

```r
nlevels(nfl1$defteam)
```

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

Removing the NAs, leaves us with 307708 observations, indicating that all 94528 NAs have been removed. In addition, 47135 explosive plays have been removed. Since our data set is now clean and tidied, we can look more into the data itself. A quick check of the number of levels in *defteam* yields only 32 factors- one for each of the NFL's 32 teams.

Exploratory Data Analysis

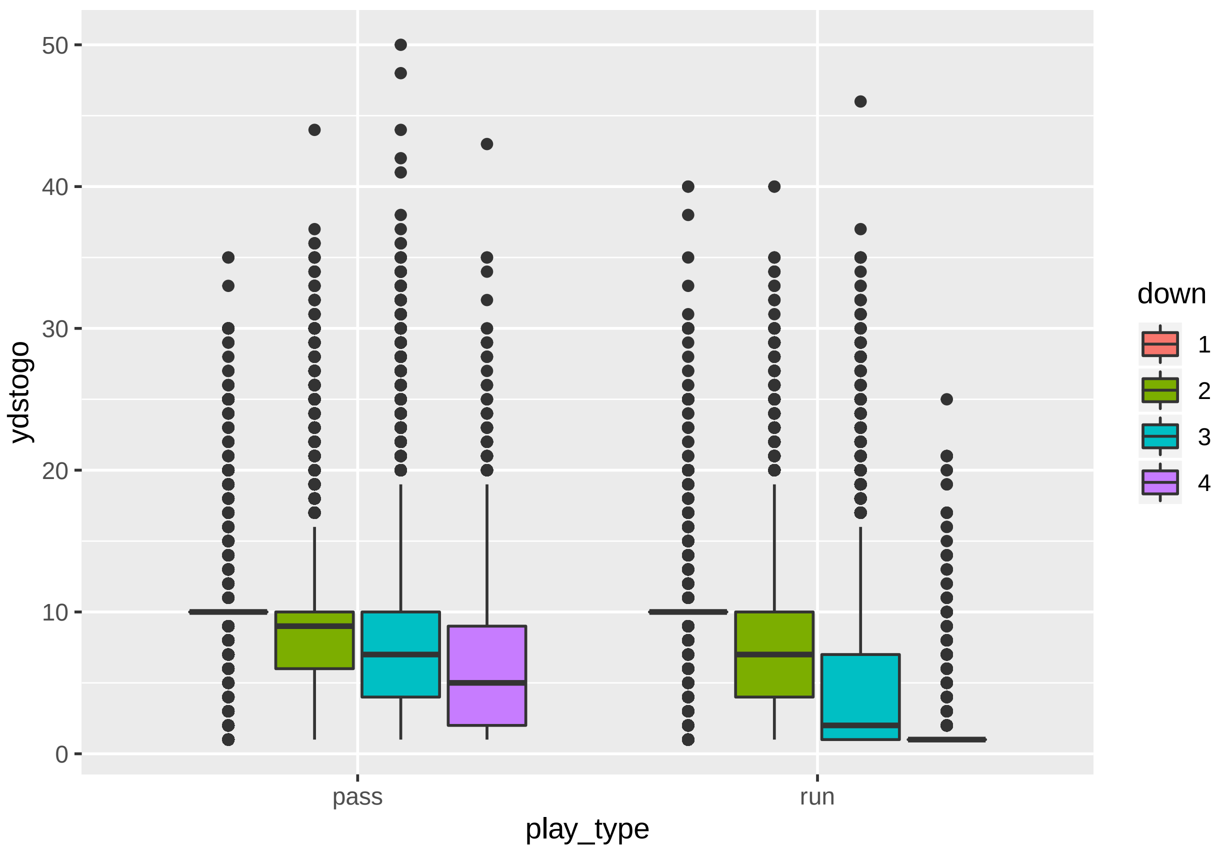
First we'll take a quick glance at the summary stats for each variable.

```r
summary(nfl1)
```

```
##  down        yards_gained      play_type         ydstogo        side_of_field
##  1:135673   Min.    :-38.000   pass:181983   Min.   : 1.000   opposing:128777
##  2:102752   1st Qu.:  0.000   run :125725   1st Qu.: 6.000   own     :178931
##  3: 64711   Median :  3.000                 Median :10.000
##  4:  4572   Mean   :  4.546                 Mean   : 8.627
##             3rd Qu.:  8.000                 3rd Qu.:10.000
##             Max.   : 34.000                 Max.   :50.000
##
##     defteam
##  TEN    :  9991
##  CLE    :  9914
##  NE     :  9847
##  CIN    :  9822
##  KC     :  9808
##  ARI    :  9801
##  (Other):248525
```

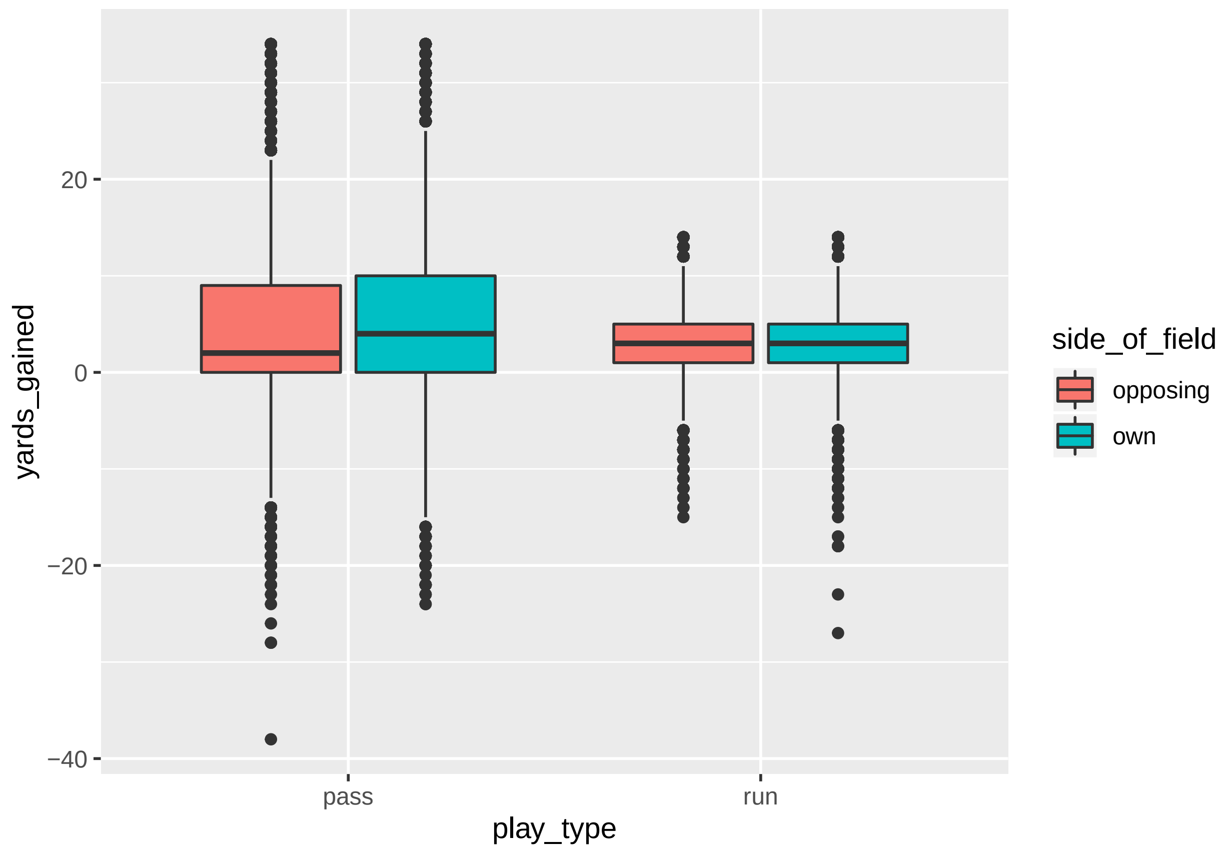
To help visualize these statistics, we'll look at some plots.

```r
nfl1 %>%
  ggplot(aes(play_type, ydstogo, fill = down)) +
  geom_boxplot()
```



As we see,the first down box is so densely packed, because nearly every first down play is 1st and 10. In fact, there are only 1.3635×10^4 instances where it's first down and there are not 10 yards to go. This is only 10.05% of the first down plays. The other place we see numerous outliers is 4th down runs. This makes sense, as teams will occasionally run on 4th and 1, depending on the situation (weather, quarter, score, etc.). There are 1623 times where this occurred and only 14.97% of these fourth down runs occurred with more than 2 yards to go to fourth down. Other than those three densely concentrated play type-down combos, the data look reasonably spread. It's fair to wonder how a play can result in 3rd and 50, but this is moist likely due to a combination of prior plays going for a loss of yardage and penalties.

```
nfl1 %>%
  ggplot(aes(play_type, yards_gained, fill = side_of_field)) +
  geom_boxplot()
```



As we see, the distribution of run gains on both sides of the field are pretty evenly spread. It's important to remember that all explosive plays have been removed, so that influences the spread. Pass gains are spread higher in teams' own side of field than the opposing. This makes sense as defenses get stouter once they are within 50 yards or less of the end zone. Additionally, touchdown passes of less than 35 (remember, explosive plays removed) are generally completed in the opposing side and thus there is a max to the number of yard gained. For example, if I score a pass TD from the 7, I can obviously only gain 7 yards.

Models

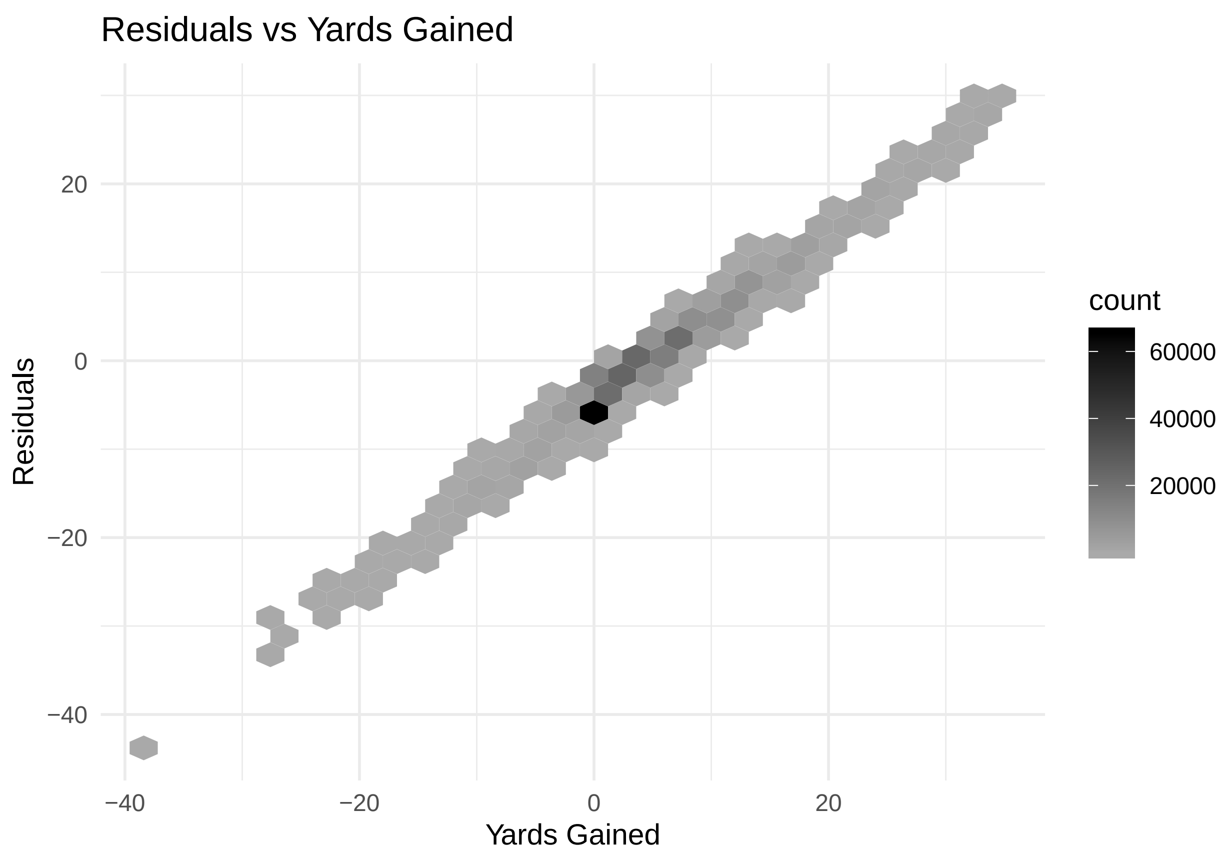
Linear Model

We decided to use a random Forest model to attempt to predict the number of yards gained given certain parameters. This was also the premise of our Shiny app. We first tried a linear regression, but that didn't work too well.

```r
mod1 <- lm(yards_gained ~ ., data = nfl1)
```

With an R^2 value of 0.0305329, this indicates that the model explains only 3.0533% of the variability in yard gained around its mean. This means that much of the variability in the data is left unexplained by this model.

```r
nfl2 <- nfl1 %>%
  add_residuals(mod1)

nfl2 %>%
  ggplot(aes(x = yards_gained, y = resid)) +
  geom_hex() +
  scale_fill_gradient(low = "darkgray", high = "black") +
  labs(title = "Residuals vs Yards Gained",
       x = "Yards Gained",
       y = "Residuals") +
  theme_minimal()
```



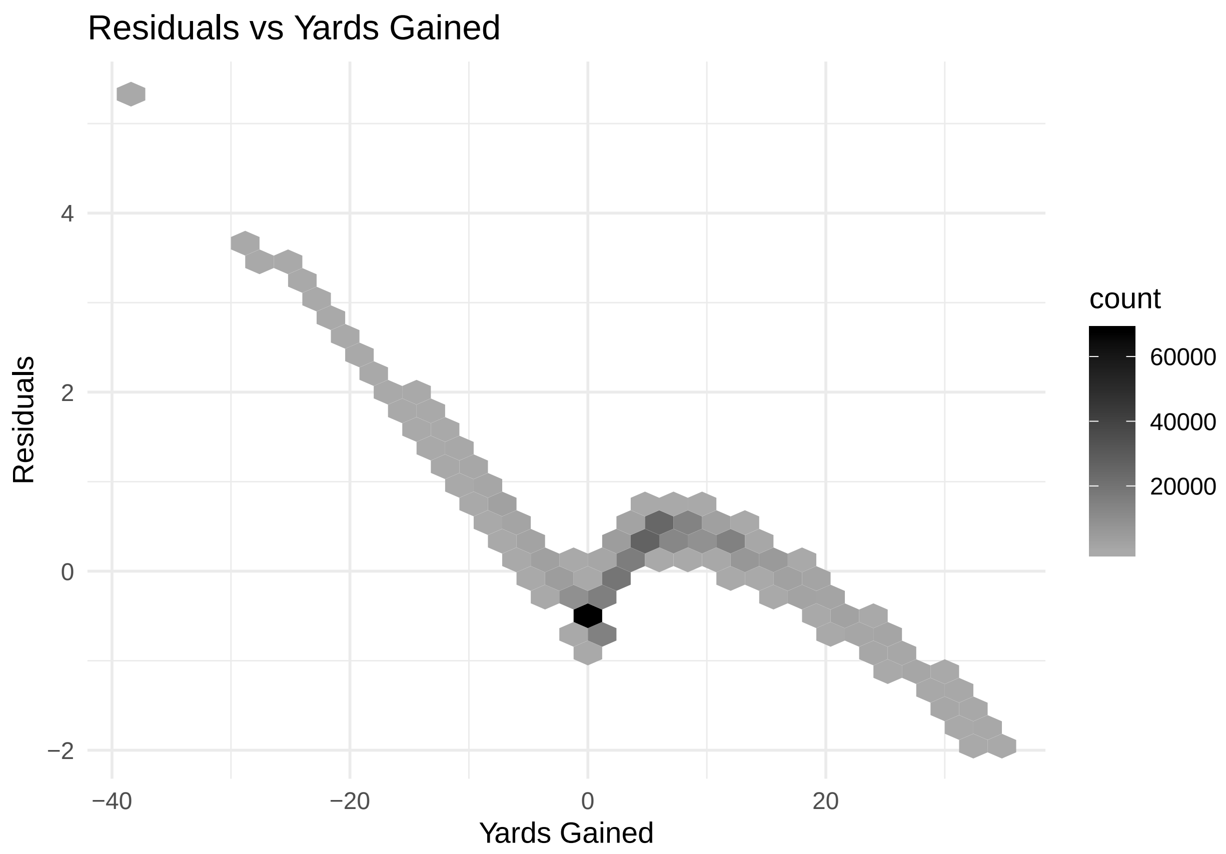
When we look at our residuals, there is a clear pattern. Some of this probably comes form the discrete nature of *yards_gained*. However, our linear pattern indicates a natural log transformation of yards_gained would better explain variability in our model. To do so, we need to transform our explanatory variable, *yards_gained* to $ln(yardsgained)$. Since this will return NaNs, NA's and -Inf's, we'll set those equal to 0. It is not perfect, because the NaNs came from negative yard gains, so they're not equivalent to 0.

Natural Log Transformed Linear Model

```
ln_yards <- log(nfl1$yards_gained)

ln_yards[which(!is.finite(ln_yards))] <- 0

mod2 <- lm(ln_yards ~ ., data = nfl1)
```

In this model, our R^2 is 0.8282, indicating that now our model explains 82.82% of the variability in $ln(yardgained)$ around its mean.

```
nfl3 <- nfl1 %>%
  add_residuals(mod2)

nfl3 %>%
  ggplot(aes(x = yards_gained, y = resid)) +
  geom_hex() +
  scale_fill_gradient(low = "darkgray", high = "black") +
  labs(title = "Residuals vs Yards Gained",
       x = "Yards Gained",
       y = "Residuals") +
  theme_minimal()
```



So, even though we still have a high R^2 value, our residual plot still shows a large amount of pattern. Some of this definitely comes from transforming 107011 data points into zeroes. This is also perhaps why we have such a high R^2 value, because the model can accurately, but falsely, explain the data around 0 very well. Since our linear models didn't do a great job of prediction after a very base-level analysis of them, we'll try to use machine learning.

randomForest

Our first step is to split our data into two, one for pass plays and the other for run plays. We'll also split our data, 75% into training and the remaining 25% into our test data.

```r
run <- nfl1 %>%
  filter(play_type == "run")

pass <- nfl1 %>%
  filter(play_type == "pass")

set.seed(100)

sample <- sample.split(run$yards_gained, SplitRatio = .75)
train_run <- subset(run, sample == TRUE)
test_run <- subset(run, sample == FALSE)

sample <- sample.split(pass$yards_gained, SplitRatio = .75)
train_pass <- subset(pass, sample == TRUE)
test_pass <- subset(pass, sample == FALSE)
```

We'll run two randomForests- one for run plays, one for pass plays.

```r
rf_run <- randomForest(yards_gained ~ . -play_type, data = train_run,
                       ntree = 100, importance = TRUE)

rf_pass <- randomForest(yards_gained ~ . -play_type, data = train_pass,
                        ntree = 100, importance = TRUE)
rf_run
```

```
##
## Call:
##  randomForest(formula = yards_gained ~ . - play_type, data = train_run,      ntree = 100, importance
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 13.08066
##                     % Var explained: 2.38
```

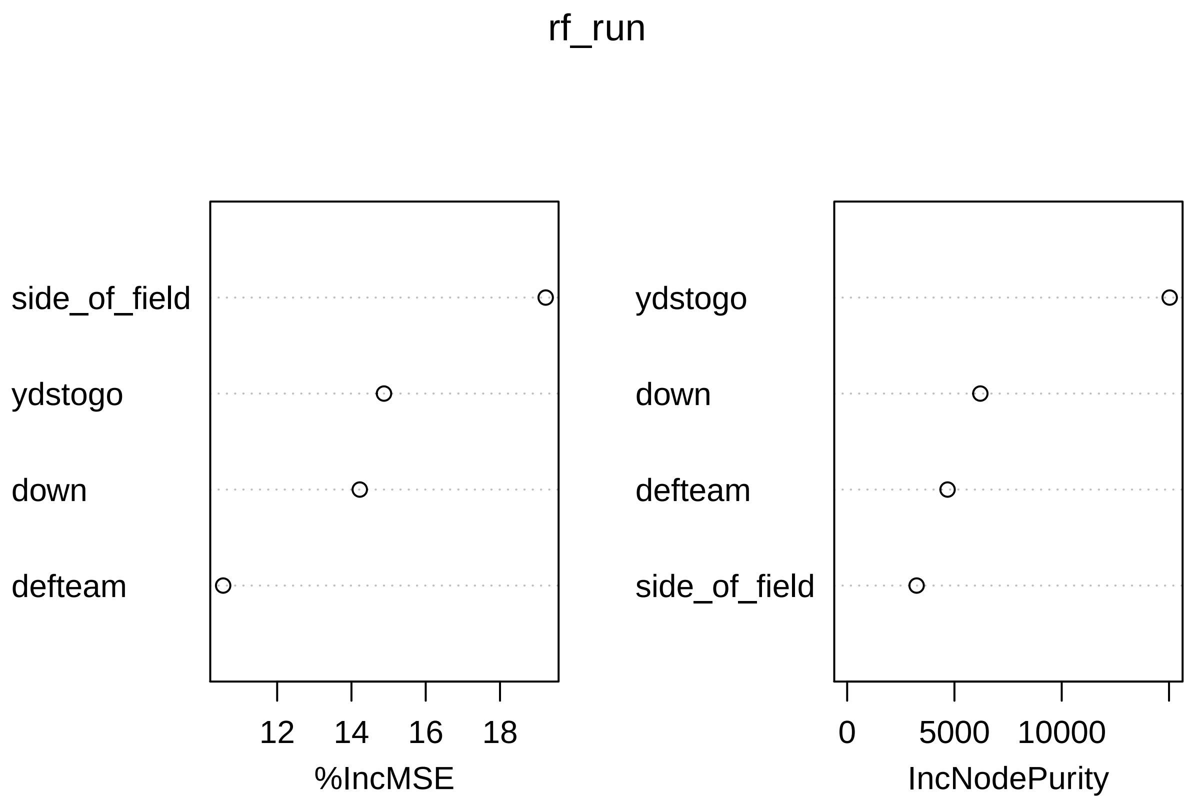
```r
rf_pass
```

```
##
## Call:
##  randomForest(formula = yards_gained ~ . - play_type, data = train_pass,      ntree = 100, importanc
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 1
```

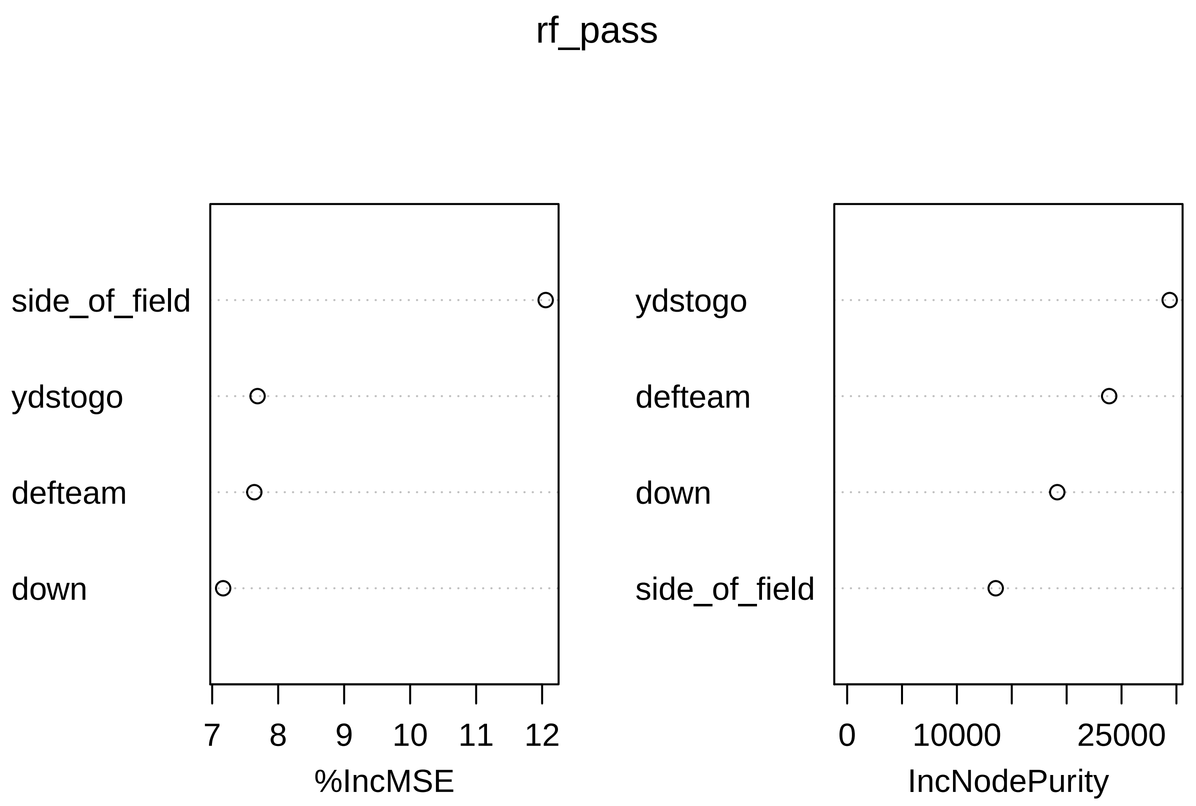
```
##
##           Mean of squared residuals: 62.48759
##                     % Var explained: 0.68
```

As we see, the output does not support any super convincing findings. With mean R^2 values of 0.0228665 and 0.0063588, this is not convincing that this model is superior to either of our regression models. However, we'll investigate a little further.

```r
varImpPlot(rf_run)
```



```r
varImpPlot(rf_pass)
```



We see that pass and run plays differ in variable importance, which makes sense. Naturally, a run play is going to be more influenced by yards to 1st down- if it's 3rd and 1, a team will most likely try to plow forward a yard or two, whereas on 1st and 10, a team will try to pick up a sizable chunk of yardage when running the ball. Meanwhile, for pass plays, the side of field holds a little bit more importance than yards to go. Similarly to run plays, yards to first down has a big impact but alike to the regression model, pass plays will differ based on if you're on your side of the oppositions side of the field.

```r
pred_run <- predict(rf_run, data = test_run[-5])

mean(pred_run[31433] - test_run$yards_gained)
```

```
## [1] 0.1149838
```

This, however, gives decent support that the predictions from our model are solid. We'll continue along with this model, as it is more flexible as the NFL evolves.

Shiny App

Our Shiny app used the randomForest model from above and combines that with a data tab. A user can select play_type (which will select one model of the other), defending team, yards to first down, and finally the side of the field they're on. It can be found using this link.

Conclusions

With more time, it would have been nice to explore more models and learned more in-depth about random-Forests. For the purpose of this project, machine learning didn't necessarily predict the most accurate but in the long run will be the best to match the evolution of the league. The app turned out very nice and we're proud of that, however, we wish the model was more fine tuned. Another change that could be made is finding a larger play-by-play data set (updated more recently, more historical data, etc.). There are a lot of adjustments that could be made with more time and more ML education. Overall, the model is reasonable and does a decent job at estimating.