Project 2

NFL Data Science - 4th Down Conversion Rate

Matthew Graeff mkg2223

Introduction

The datasets I have chosen provide data on every passing play in the 2018 regular season of the NFL. The data can be found on Kaggle (https://www.kaggle.com/code/aryashah2k/sports-analytics-visualization/notebook (https://www.kaggle.com/code/aryashah2k/sports-analytics-visualization/notebook)). I chose to use the datasets games.csv and plays.csv.

- Games.csv has data on the week, date, start time, home and away team, and a unique game ID.
- Plays.csv has data on the play such as the game ID, week, quarter, down, game clock, yards to go, yards earned, possession team, and more.

I am interested in this data because I wanted to see for myself how successful teams are at getting a conversion on 4th down because that has become a bigger talking point in NFL media in the last year. *A conversion just means that the team either got the yards they needed in order to gain a new set of downs (attempts) or they scored; if a team fails to convert, then they surrender the ball to the other team.* In this data, field goals are ignored, so scoring is limited to touchdowns. Additionally, I originally thought this data had records for running plays, too, but it is strictly passing plays. Thus, this analysis is incomplete for my original interest in testing the success of any 4th down attempt, but it is still useful for understanding the success of passing on 4th down.

a.) Load data

This code chunk loads the two datasets in and also loads the libraries needed.

```
# load games dataset from local computer
games <- read.csv("C:/SDS 322E Data Science/Project/games.csv")
# load plays dataset from local computer
plays <- read.csv("C:/SDS 322E Data Science/Project/plays.csv")
# load libraries
library(tidyverse)
library(ggplot2)
library(GGally)
library(cluster)
library(factoextra)
library(plotROC)
```

These datasets are already tidy, and any kind of pivoting would make the data less tidy. So, I will proceed to join them without pivoting them.

b.) Join data

Games has 253 records (It seems like 3 games are missing. I do not know why these games are missing, but they are insignificant in the aggregate.) and 6 columns. Plays has 19,239 records and 27 columns. Joining the two on game ID yields a dataset called games_plays that has 19,239 records and 32 columns. Joining them in this way added the 5 non-ID columns of games to every record in plays.

```
# count rows, columns of games
dim(games)
```

```
## [1] 253   6
```

```
# count rows, columns of plays
dim(plays)
```

```
## [1] 19239    27
```

```
# join games to plays on gameId
games_plays <- left_join(plays, games, by = "gameId")
# count rows, columns of games_plays
dim(games_plays)
```

```
## [1] 19239    32
```

c.) Manipulate dataset

In this code chunk I first filter by 4th down. Then, I create new columns based on existing data. The first new column is a Boolean that shows if the home team has possession, the next two show the scores based on which team has possession of the ball, the fourth shows how many more points the team with the ball has than the defending team, the fifth is a Boolean that shows if the possession team has the point lead, and the sixth is a Boolean that shows if the play resulted in a conversion (1st down or touchdown). Then, I select only the most relevant 6 columns, which are the quarter that the play occurred it, the yardsToGo until a conversion, and four of the newly created columns. Lastly, I drop the rows with NA values.

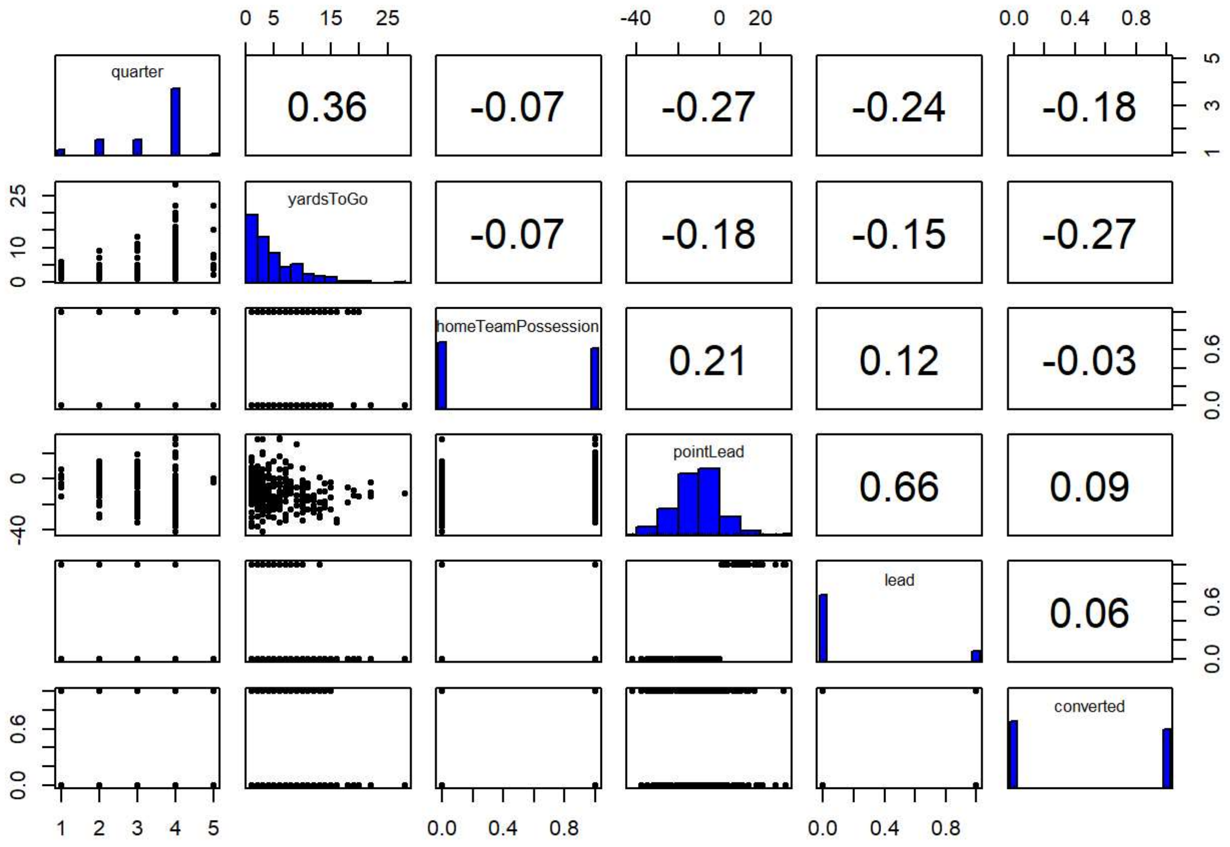
```
# filter by 4th down, create new columns, select 6 important columns, drop na's, and save new da
taframe
games_plays <- games_plays %>%
  filter(down==4) %>%
  mutate(homeTeamPossession = ifelse(possessionTeam == homeTeamAbbr, 1, 0),
         preSnapPossessionScore = ifelse(homeTeamPossession, preSnapHomeScore, preSnapVisitorSco
re),
         preSnapDefenderScore = ifelse(homeTeamPossession, preSnapVisitorScore, preSnapHomeScor
e),
         pointLead = preSnapPossessionScore - preSnapDefenderScore,
         lead = ifelse(pointLead > 0, 1, 0),
         converted = ifelse(offensePlayResult >= yardsToGo, 1, 0)) %>%
  select(quarter, yardsToGo, homeTeamPossession, pointLead, lead, converted) %>%
  drop_na()

# count rows, columns of games_plays
dim(games_plays)
```

```
## [1] 339   6
```

Exploratory Data Analysis

In this section I build a correlation matrix that shows univariate graphs along the diagonal, bivariate graphs in the bottom left half and correlation coefficients in the upper right half.

```
# psych is a package for building a correlation matrix with univariate and bivariate graphs and
 correlation coefficients
library(psych)

# create correlation matrix
pairs.panels(games_plays,
             method = "pearson", # correlation coefficient method
             hist.col = "blue", # color of histogram
             smooth = FALSE, density = FALSE, ellipses = FALSE)
```



As expected (per results from Project 1), the average 4th down conversion rate was just slightly under half of the time. Something to note is that the correlation coeffiecients between the converted variable and the other variables are very low. Combined, these two observations indicate that no matter what the circumstances are, the likelihood of converting on 4th down is similar to a coin flip. The biggest correlation is with the yardsToGo variable. The bivariate graph indicates that successful conversions have a smaller average yards to go.

The biggest correlation is between the point_lead and the lead. This makes sense because lead is just a Boolean that describes if point_lead is greater than 0. Other than that, the data is largely uncorrelated.

Clustering

a.) Prepare the data

In this section I prepare the data for clustering by removing the target variable, converted. Additionally, I scale the data by subtracting each value by the column's mean value and then dividing by its standard deviation.

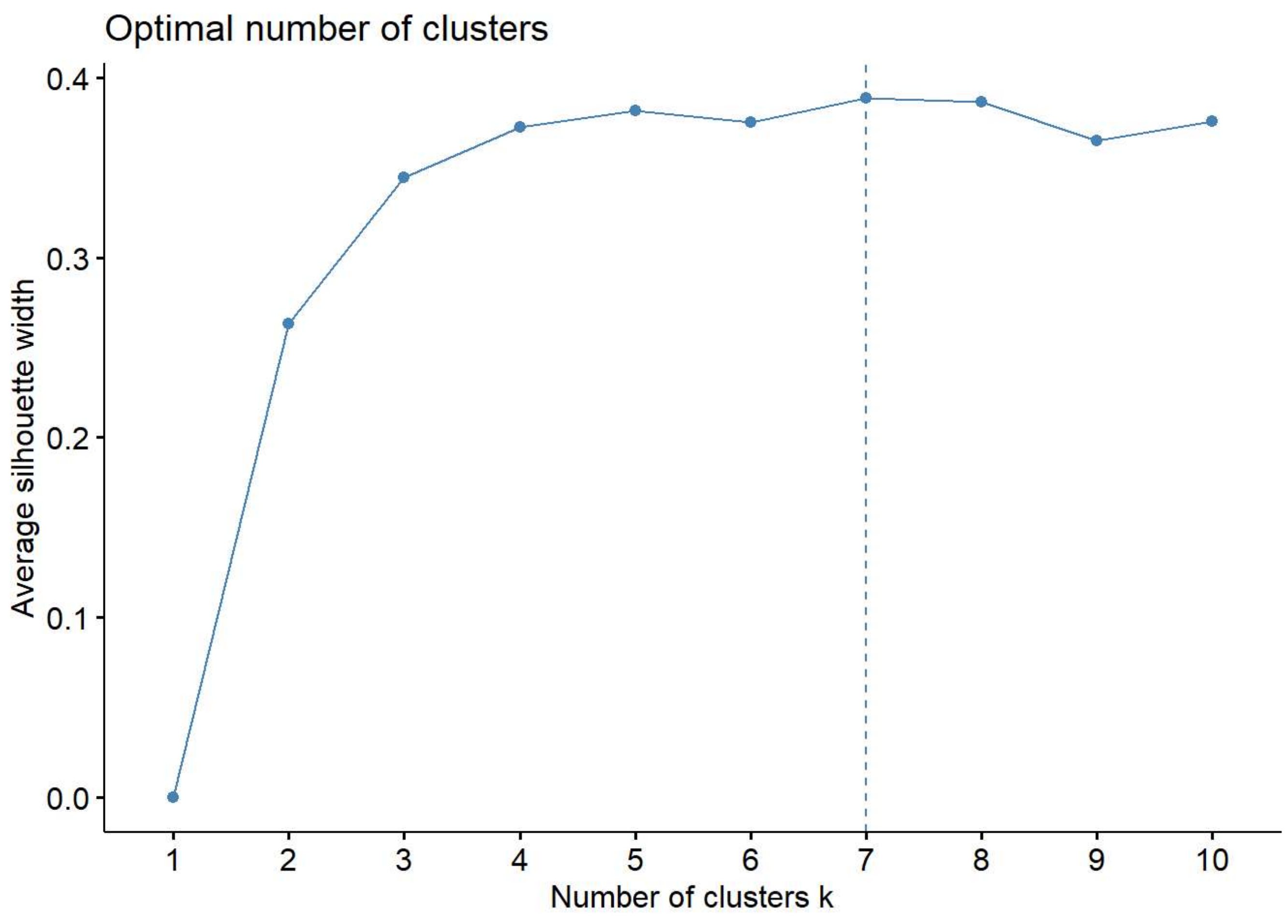
```
# create new dataset scaled and without converted variable
games_plays_cluster <- games_plays %>%
  select(-converted) %>%
  scale

# check first 6 rows of clustering dataset
head(games_plays_cluster)
```

```
##         quarter   yardsToGo homeTeamPossession   pointLead      lead
## [1,]  0.6327975 -0.08058041          1.0499184 -0.39605738 -0.405539
## [2,] -2.4317504 -0.51417976         -0.9496453  1.01987233  2.458580
## [3,] -1.4102344 -0.94777911          1.0499184  1.10316231  2.458580
## [4,]  0.6327975  0.35301894         -0.9496453 -0.06289745 -0.405539
## [5,] -0.3887185 -0.08058041         -0.9496453 -0.47934736 -0.405539
## [6,]  0.6327975  0.13621927          1.0499184  0.35355246 -0.405539
```

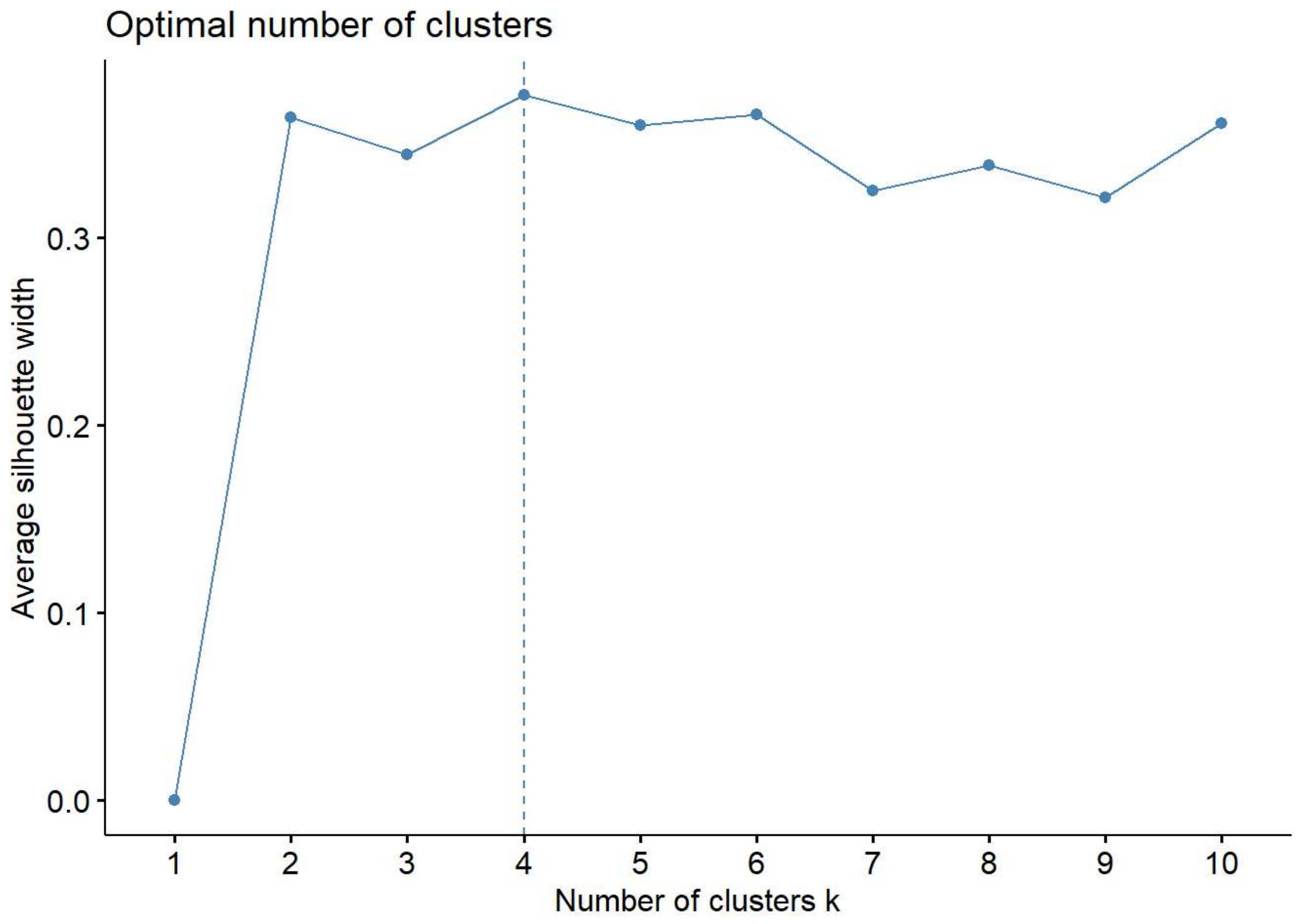
b.) Find optimal number of clusters based on silhouette width

In this section I plot the average silhouette width (a metric for determing the goodness of fit of clusters) to determine how many clusters would be optimal for my data. I plot this for both the PAM method and the kmeans method to get a rough idea of which one might perform better.

```
# plot average silhouette width vs number of clusters for PAM method
fviz_nbclust(games_plays_cluster, pam, method = "silhouette")
```



```
# plot average silhouette width vs number of clusters for kmeans method
fviz_nbclust(games_plays_cluster, kmeans, method = "silhouette")
```



The optimal number of clusters for the PAM method yields an average silhouette width of about 0.4, which is weak, but slightly better than the best possibility for the kmeans method. However, since I know that my converted column only has two outcomes, I would like to have two clusters. The kmeans method has a higher average silhouette width at two clusters (nearly equal to the peak value at four clusters) than the PAM method, so I will be using the kmeans method to cluster my data.

c.) Perform kmeans clustering

```
# cluster data using kmeans method and 2 clusters
kmeans_results <- games_plays_cluster %>%
  kmeans(2)

#calculate average silhouette width for 2 clusters
ss <- silhouette(kmeans_results$cluster, dist(games_plays_cluster))
mean(ss[,3])
```

```
## [1] 0.3637054
```

The average silhouette width is only about 0.36, which is weak and indicates that the clusters could be artificial. Given how uncorrelated things appeared in the correlation matrix, this does not surprise me.

d.) Evaluate performance

In the following two code chunks I will evaluate how the model performs for predicting the converted variable. The first section creates a confusion matrix and the second calculates the overall accuracy.

```
# add cluster labels to dataset with converted label
games_plays_kmeans <- games_plays %>%
  mutate(cluster = as.factor(kmeans_results$cluster))

# create confusion matrix with clusters (predicted) as rows and converted (actual) as columns
table(games_plays_kmeans$cluster, games_plays_kmeans$converted)
```

```
##
##       0   1
##   1  37  55
##   2 143 104
```

```
# calculate accuracy (correct predictions/total data points)
(55+143)/(339)
```

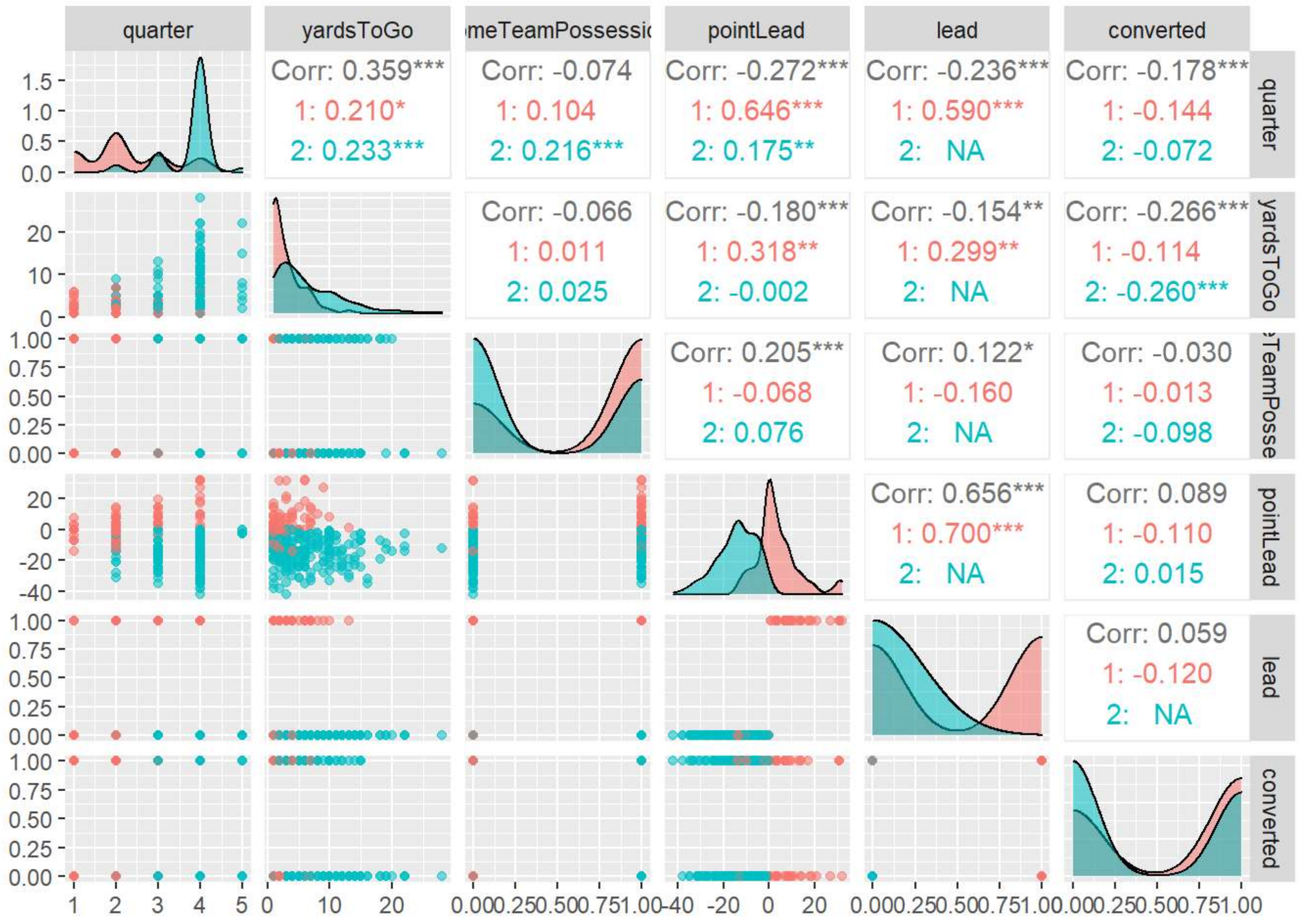
```
## [1] 0.5840708
```

This clustering method has an accuracy of 0.58, which is very low. It is only slightly better than a coin-toss. This is part of what makes football so exciting; no matter what the situation is, it always seems like teams have a chance to do something spectacular.

e.) Visualize clusters by pairwise combinations of variables

In this section I visualize the clusters with univariate graphs, bivariate graphs, and correlation coefficients. The first cluster (converted) is represented with red, and the second (not converted) is represented with blue.

```
# plot variables and pairwise combination of variables colored by cluster
ggpairs(games_plays_kmeans, columns = 1:(ncol(games_plays_kmeans)-1), aes(color = cluster, alpha
= 0.5))
```



One of the main findings from Project 1 is that going for 4th down in the first quarter with only a few yards to go is usually successful. I'm happy to see that this clustering method picked up on that also, as seen in the bivariate graph between quarter and yardsToGo. Similarly, in Project 1 I saw that the teams who are already successful and winning the game are more likely to be successful in 4th downs than those that are losing, which this also picks up on. Also as noted previously, homeTeamPossession does not actually matter as much as people might expect when it comes to 4th downs.

Since the kmeans method does not use actual data points for its cluster centers, I will look at the univariate graphs to get an idea of what the centers of each cluster look like. The center of the first cluster, which represents a greater likelihood of converting, has a low quarter, a low yardsToGo, the homeTeamPossession, a positive pointLead, the lead or not the lead, and a conversion. The center of the second cluster, which represents a smaller likelihood of converting, has a high quarter, a higher yardsToGo, not the homeTeamPossession, a negative point_lead, not the lead, and not a conversion.

Dimensionality Reduction

a.) Prepare the data

Again, I will drop the converted column so the model does not get biased by it. Additionally, I need to scale the data as with the clustering dataset.

```
# create new data frame for dimensionality reduction by scaling the data
games_plays_dr <- games_plays %>%
  select(-converted) %>%
  scale %>%
  as.data.frame
```

b.) Perform PCA

In this section I look at the principal components and the percentage of variance explained by each one.

```
# compare to PCA performed with the function prcomp()
pca <- games_plays_dr %>%
  prcomp()

# look at percentage of variance explained for each PC in a table
get_eigenvalue(pca)
```

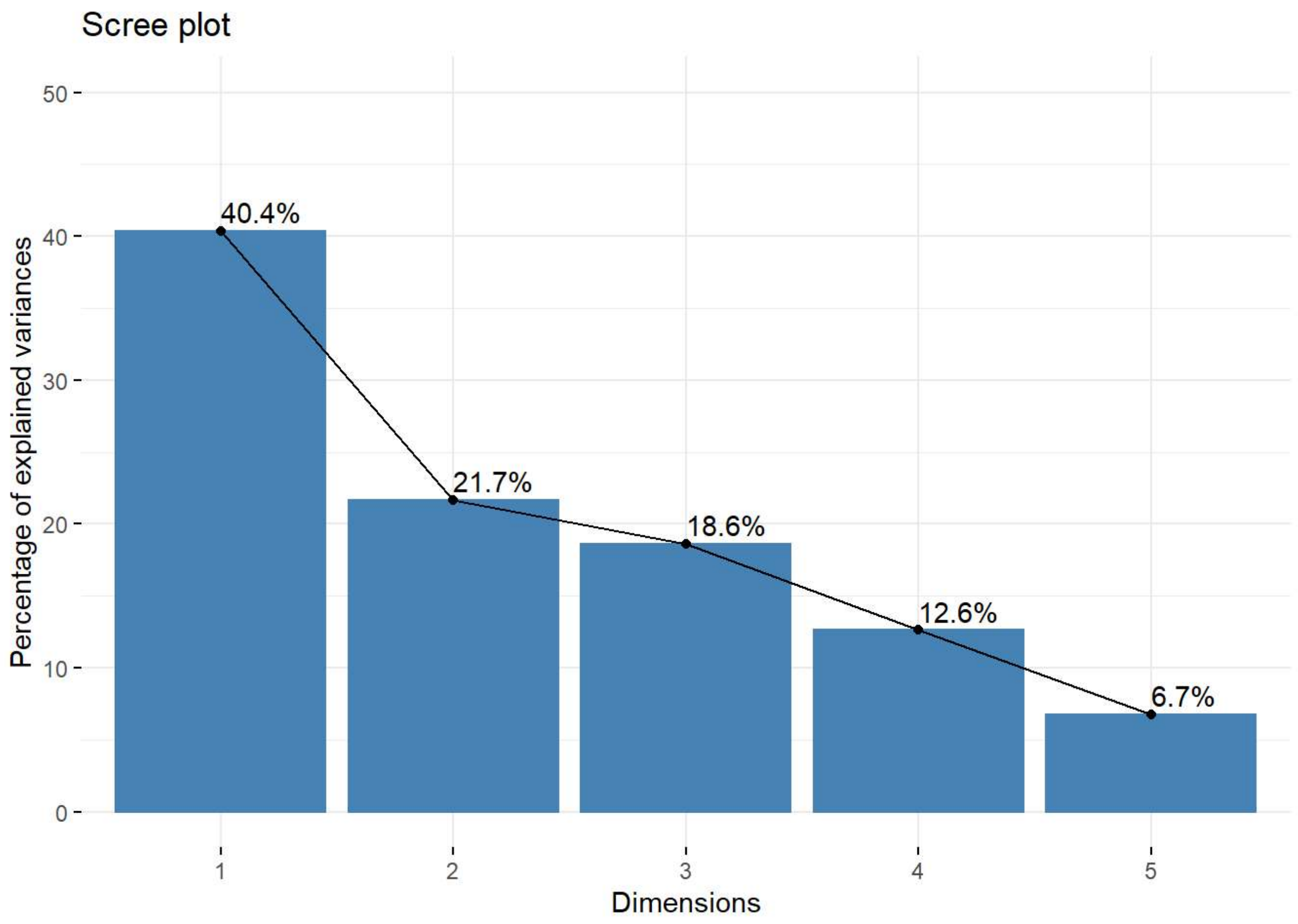
```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  2.0181106        40.362211                    40.36221
## Dim.2  1.0839704        21.679408                    62.04162
## Dim.3  0.9298024        18.596048                    80.63767
## Dim.4  0.6309253        12.618507                    93.25617
## Dim.5  0.3371913         6.743826                   100.00000
```

Only the first 3 of 5 dimensions are needed to get about 80% of the variance explanation.

c.) Scree plot

This scree plot is just a way to visualize what is going on in the table above.

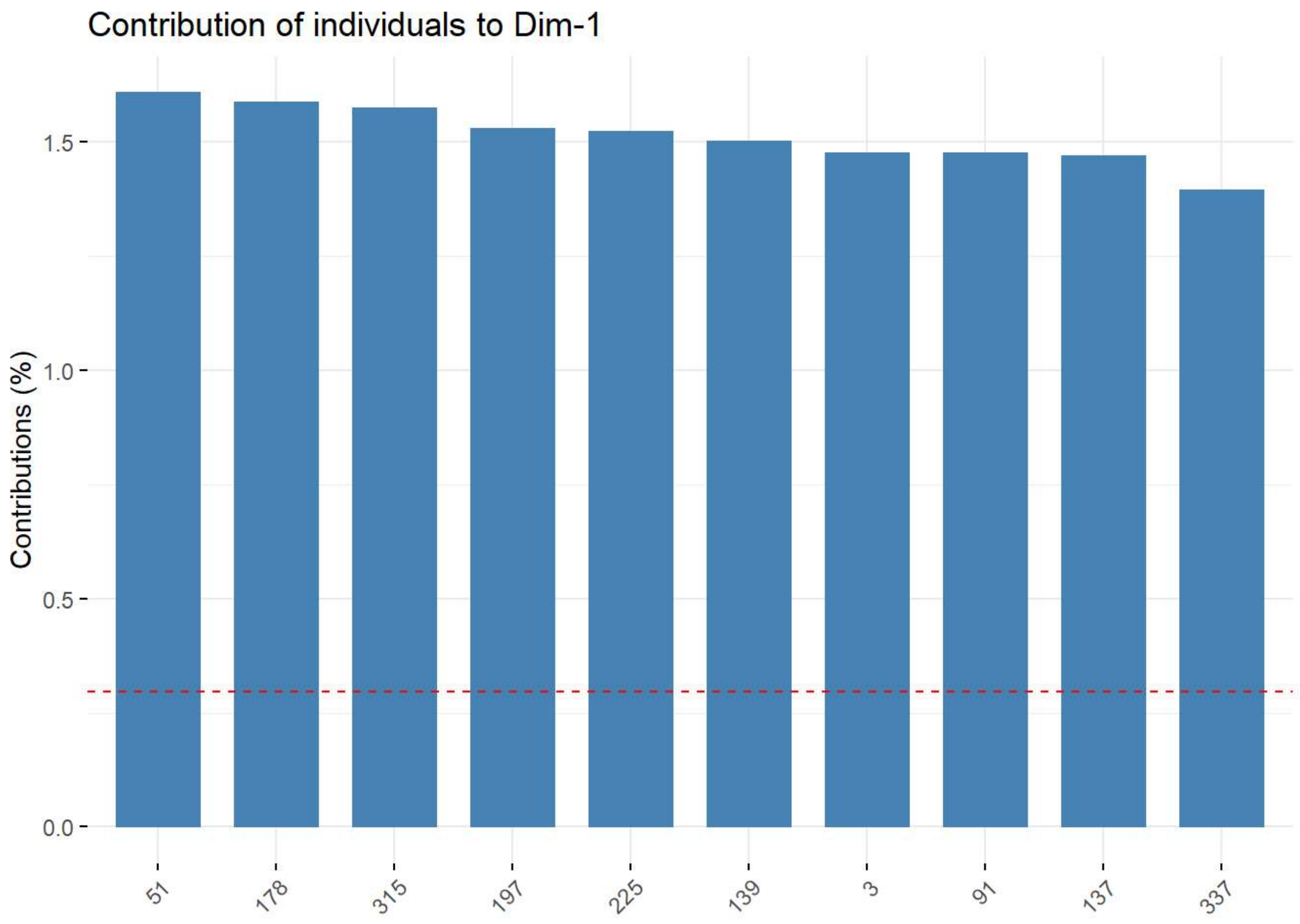
```
# visualize percentage of variance explained for each PC in a scree plot
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))
```



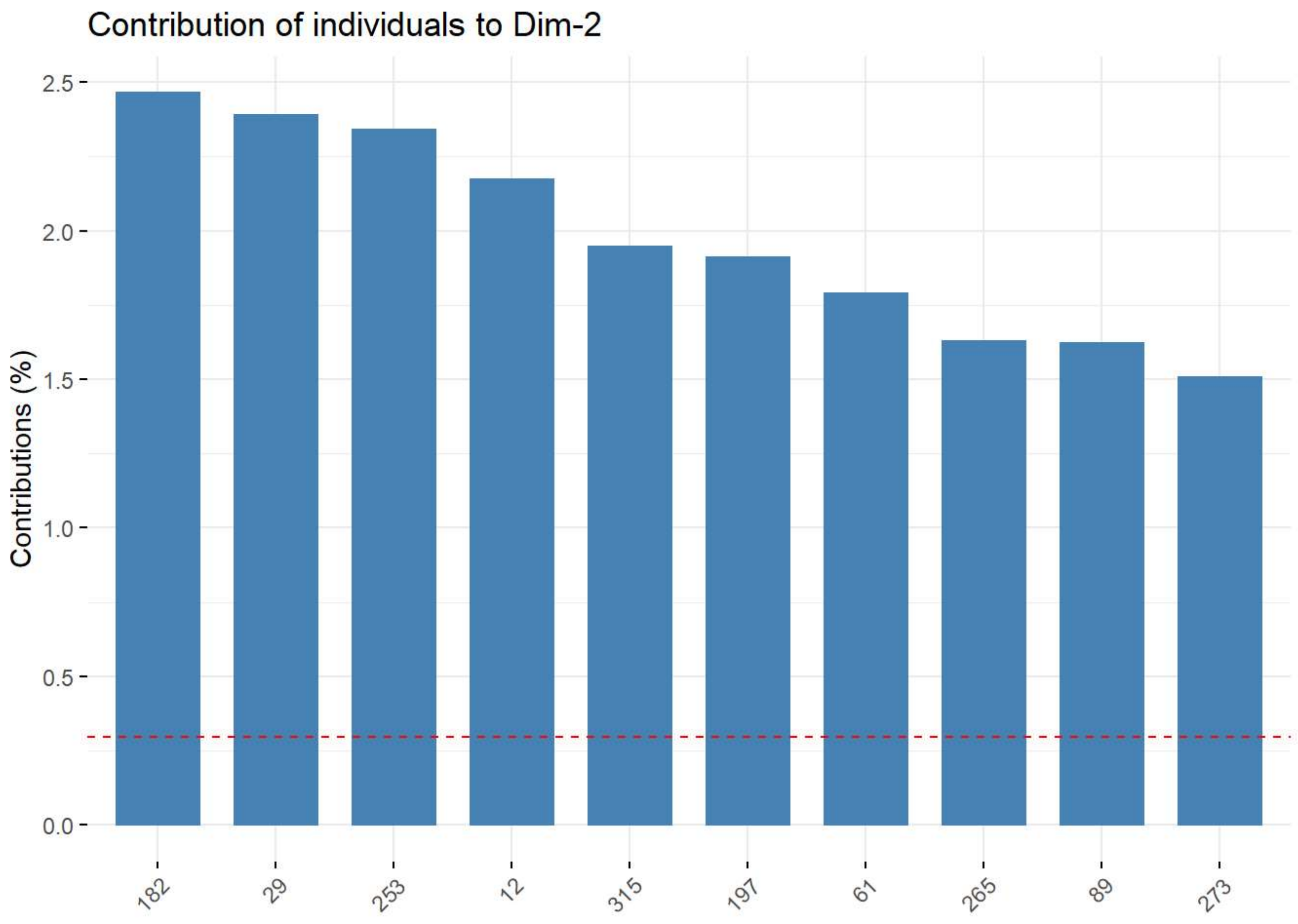
d.) Individual contributions

The following two plots show which rows in the data contribute the most to the first two dimensions. Even though it takes the first three dimensions to explain 80% of the variance, for ease of interpretation, I will only focus on plotting the first two.

```
# Note the red dash line indicates the average contribution
# plot contribution % vs index of row for dim 1
fviz_contrib(pca, choice = "ind", axes = 1, top = 10)
```



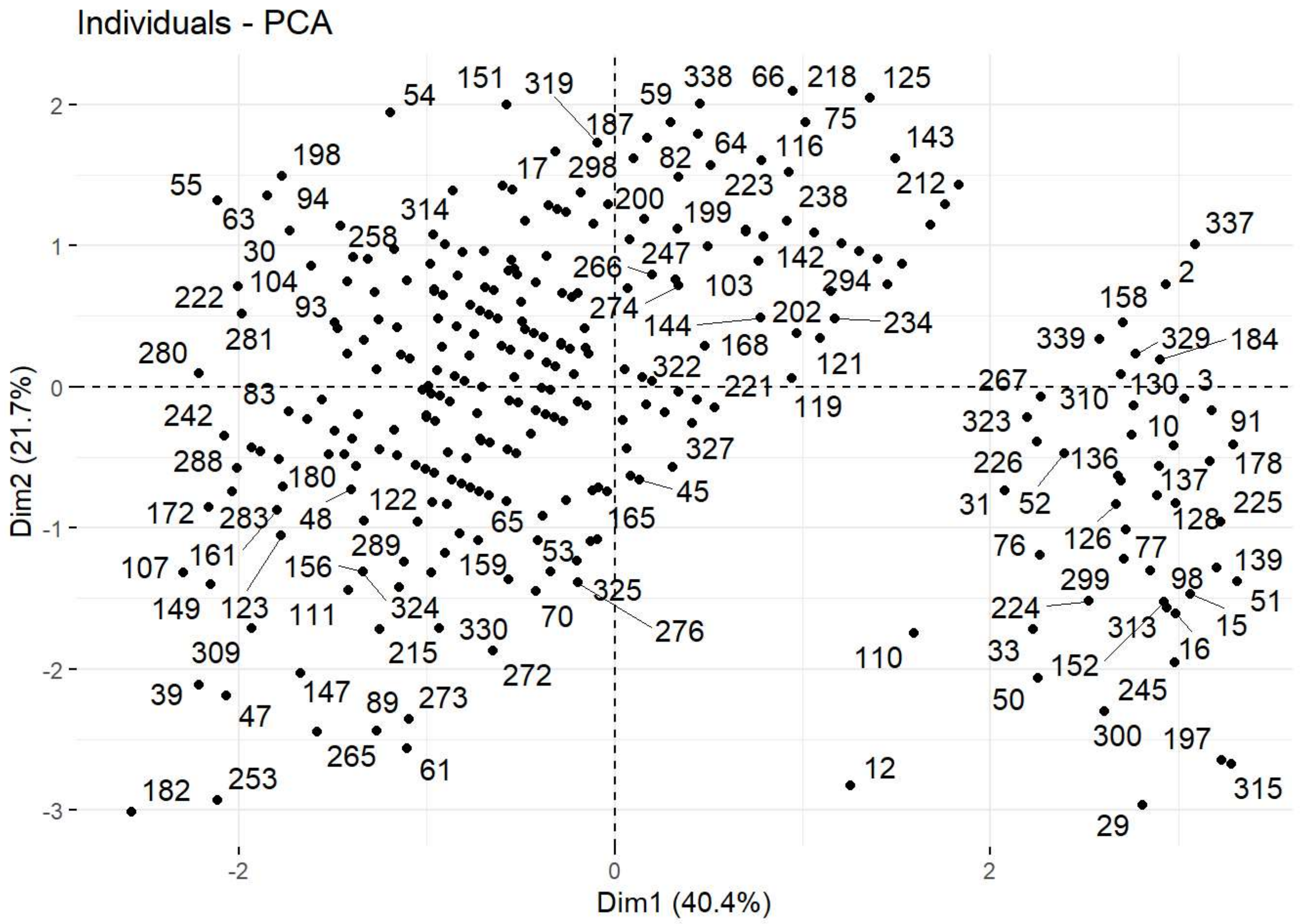
```
# plot contribution % vs index of row for dim 2
fviz_contrib(pca, choice = "ind", axes = 2, top = 10)
```



The data in these charts do not mean much on their own, but pay attention to a few of the top indexes for both dimensions. For example, 51, 178, and 315 in Dim-1 and 182, 29, and 253 in Dim-2.

In this code chunk I plot every row as a point on the Dim-1 and Dim-2 axes.

```
# visualize the individuals according to PC1 and PC2
fviz_pca_ind(pca,
             repel = TRUE) # avoid text overlapping
```



Notice how 51, 178, and 315 are the points most on the right. This is because those are the biggest contributors to Dim-1 and Dim-1 is encoded in the x-axis. Similarly, points 182, 29, and 253 are the points most on the bottom. This is because those are the biggest contributors to Dim-2, which is the y-axis.

e.) Variable contributions

In this section I show how each variable contributes to PC1 and PC2.

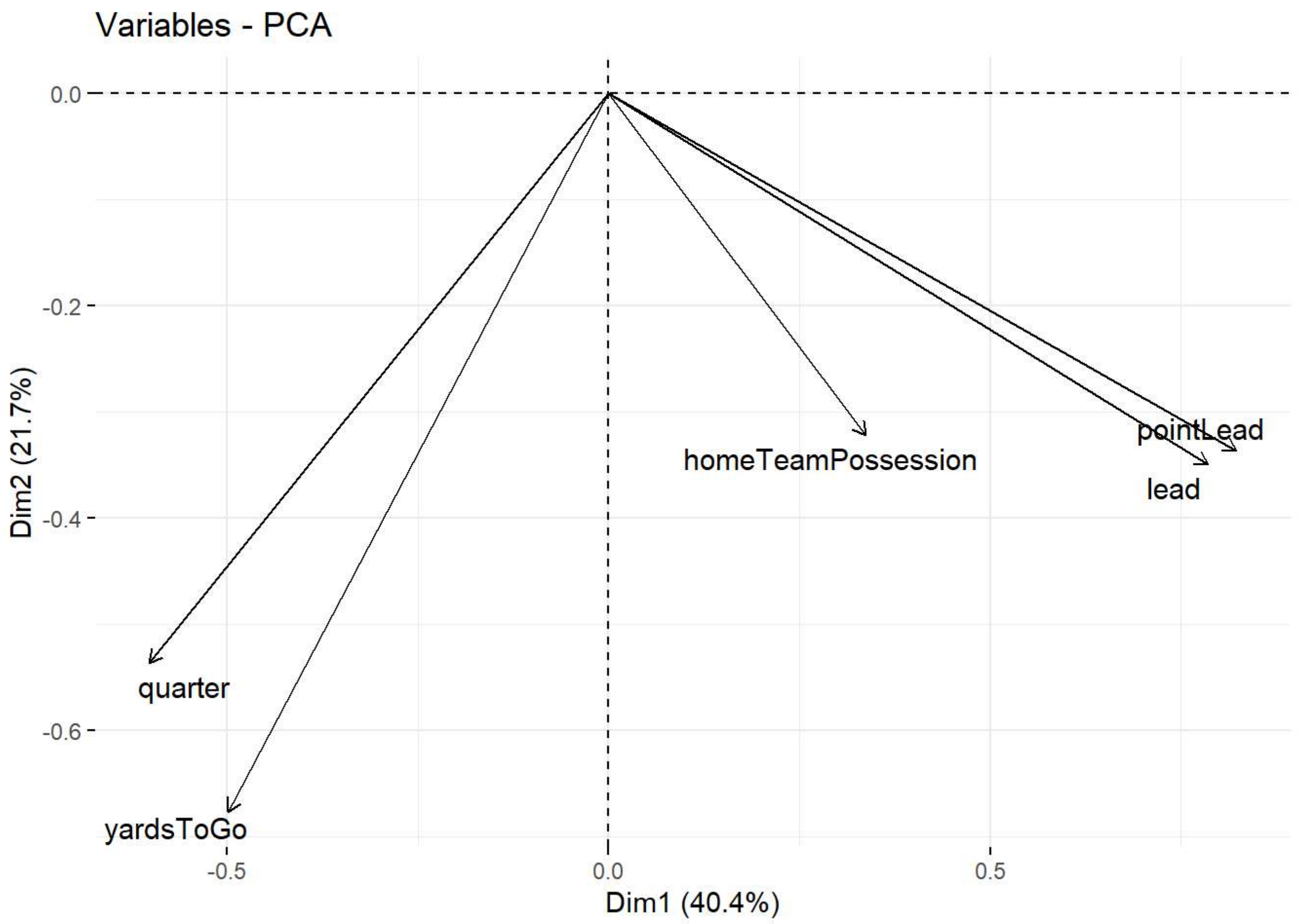
```
# visualize the contributions of the variables to the PCs in a table
get_pca_var(pca)$coord %>% as.data.frame
```

```
##                          Dim.1      Dim.2       Dim.3       Dim.4        Dim.5
## quarter             -0.6019862 -0.5362001  0.02859224 -0.59056910  0.022643265
## yardsToGo           -0.4984296 -0.6762835  0.14936290  0.52142424  0.004006663
## homeTeamPossession   0.3363772 -0.3217938 -0.88288475  0.02998388 -0.053987189
## pointLead            0.8227941 -0.3364916  0.17640428 -0.04846064  0.419900417
## lead                 0.7855898 -0.3497466  0.30995429 -0.08380192 -0.396776459
```

The important thing to take away from this table is the relative magnitudes and directions of each variable for the first two dimensions.

Next, I plot these magnitudes and directions on a Dim-1 vs Dim-2 plane.

```
# visualize the contributions of the variables to the PCs in a correlation circle
fviz_pca_var(pca, col.var = "black",
             repel = TRUE) # Avoid text overlapping
```

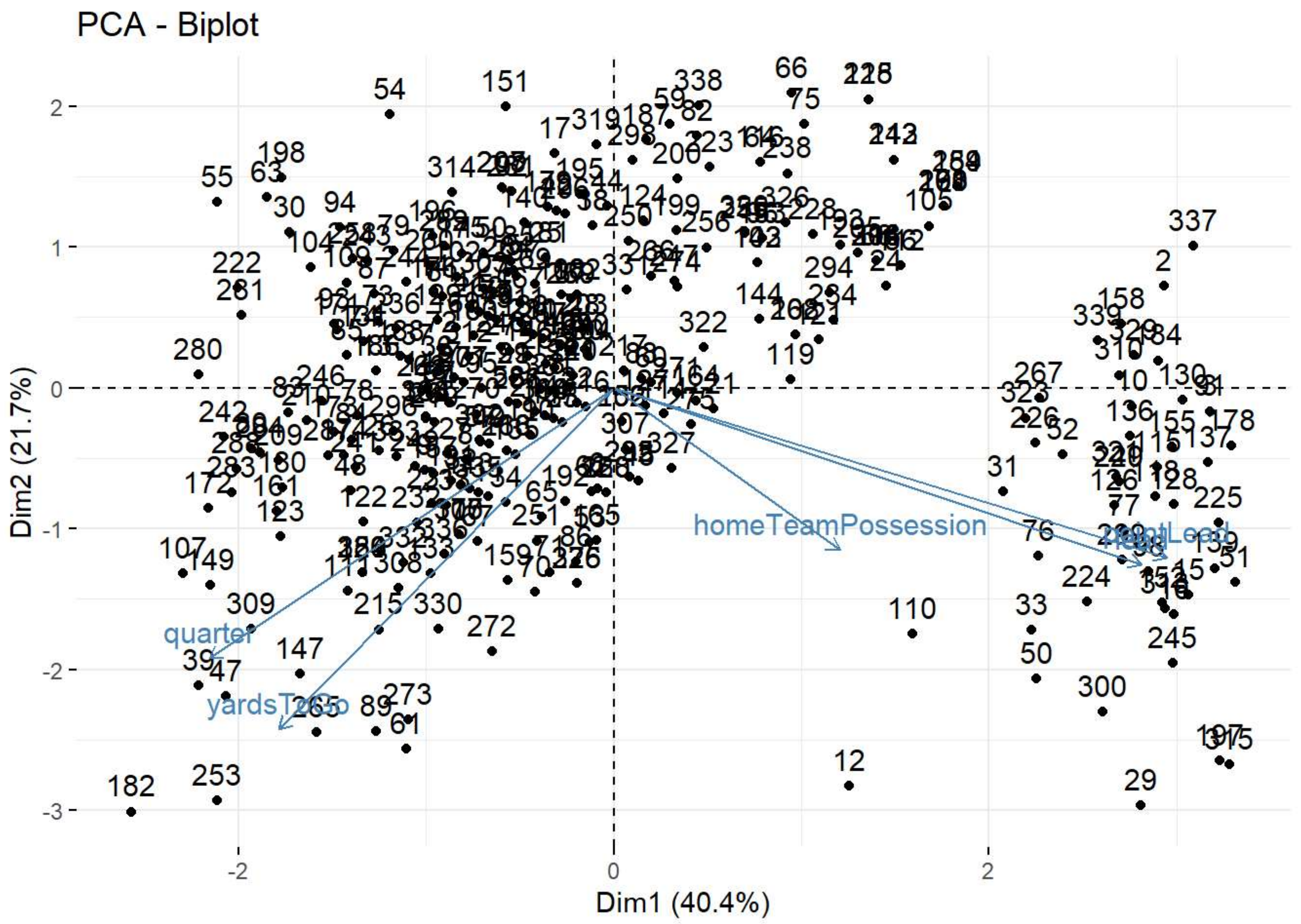


This correlation circle shows that scoring high in PC1 means scoring high in pointLead, lead, and homeTeamPossession and scoring low in quarter and yardsToGo. Scoring low in PC2 means scoring high in yardsToGo, quarter, lead, pointLead, and homeTeamPossession.

f.) Individual and variable contributions

The following graph plots both of the above graphs on top of each other.

```
# visualize both variables and individuals in the same graph
fviz_pca_biplot(pca)
```



This kind of visualization makes it easy to see which variables influence which set of individual points. There is the smaller cluster on the right that is high in Dim1 and therefore high in pointLead and lead, and then there is the bigger cluster on the left that is all over the chart and harder to define. This reinforces how uncorrelated this dataset is.

Classification and Cross-Validation

a.) Logistic regression model

In this section, I will use a classifier (the logistic regression model) to predict the converted variable using the quarter, yardsToGo, homeTeamPossession, pointLead, and lead variables, all still filtered by 4th down. As a refresher, the converted variable is a binary variable that indicates if the team converted to a new set of downs or scored a touchdown.

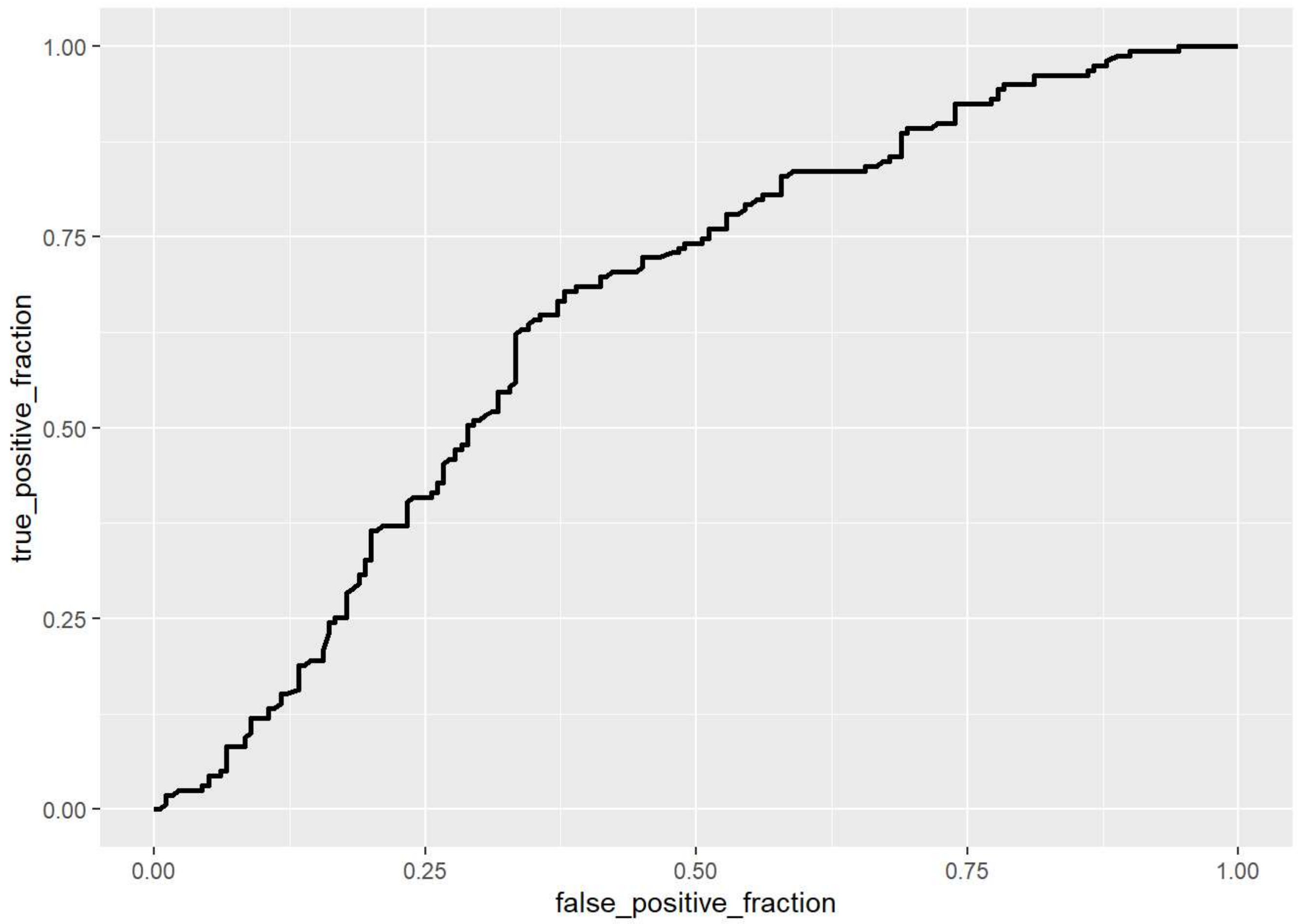
```
# create the logistic regression fit
fit <- glm(converted ~ quarter + yardsToGo + homeTeamPossession + pointLead, lead, data = games_
plays, family = "binomial")

# calculate a predicted probability and add that to a new dataframe
log_games_plays <- games_plays %>%
  mutate(probability = predict(fit, type = "response"),
         predicted = ifelse(probability > 0.5, 1, 0)) %>%
  select(quarter, yardsToGo, homeTeamPossession, pointLead, lead, converted, probability, predic
ted)

# display first 6 rows of new dataframe with predictions added from the logistic regression mode
l
head(log_games_plays)
```

```
##   quarter yardsToGo homeTeamPossession pointLead lead converted probability
## 1       4         5                  1       -14    0         0   0.4322119
## 2       1         3                  0         3    1         0   0.6617609
## 3       2         1                  1         4    1         0   0.5927969
## 4       4         7                  0       -10    0         0   0.4750699
## 5       3         5                  0       -15    0         0   0.5513442
## 6       4         6                  1        -5    0         0   0.4127174
##   predicted
## 1         0
## 2         1
## 3         1
## 4         0
## 5         1
## 6         0
```

At first glance, the model does not seem too accurate. Only three of the first six rows were predicted correctly.

To evaluate my model, I will build an ROC curve and check the value of the AUC.

```
# ROC curve
ROC <- ggplot(log_games_plays) +
  geom_roc(aes(d = converted, m = probability), n.cuts = 0)
ROC
```



```
# calculate the area under the curve
calc_auc(ROC)
```

```
##   PANEL group       AUC
## 1     1    -1 0.6534242
```

At 0.65, the AUC of this model is poor. Again, this makes sense because the dataset is so uncorrelated.

The next two code chunks build a confusion matrix so that I can calculate the accuracy of the model. The actual values are the rows and the predictions are the columns.

```
# confusion matrix: compare true to predicted condition
table(log_games_plays$converted, log_games_plays$predicted) %>% addmargins
```

```
##
##         0   1 Sum
##   0   107  73 180
##   1    50 109 159
##   Sum 157 182 339
```

```
# calculate accuracy (correct predictions/total data points)
(107+109)/339
```

```
## [1] 0.6371681
```

At about 64% accuracy, this model is not very good. However, it is better than a coin-flip and better than the kmeans clustering.

b.) k-fold cross-validation

In this section I will build the same logistic regression fit, but I will use the k-fold cross-validation method with 10 folds to check for overfitting. The output of this will be the average AUC for all 10 folds.

```
# make this example reproducible by setting a seed
set.seed(322)

# choose number of folds
k = 10

# randomly order rows in the dataset
data <- games_plays[sample(nrow(games_plays)), ]

# create k folds from the dataset
folds <- cut(seq(1:nrow(data)), breaks = k, labels = FALSE)

# use a for loop to get diagnostics for each test set
diags_k <- NULL

for(i in 1:k){
  # create training and test sets
  train <- data[folds != i, ] # all observations except in fold i
  test <- data[folds == i, ]  # observations in fold i

  # train model on training set (all but fold i)
  fit <- glm(converted ~ quarter + yardsToGo + homeTeamPossession + pointLead, data = train, fam
ily = "binomial")

  # test model on test set (fold i)
  df <- data.frame(
    probability = predict(fit, newdata = test, type = "response"),
    converted = test$converted)

  # consider the ROC curve for the test dataset
  ROC <- ggplot(df) +
    geom_roc(aes(d = converted, m = probability, n.cuts = 0))

  # get diagnostics for fold i (AUC)
  diags_k[i] <- calc_auc(ROC)$AUC
}

# average performance
mean(diags_k)
```

```
## [1] 0.6454116
```

The average AUC is 0.65, which is the same as it was in the first model. This shows me that there is not really overfitting going on. It still performs just as poorly. Personally, I would not like to rely on this model to make predictions for me. The reason I chose this dataset is because there has been a lot of discussion recently about the place for data analytics in the world of football and I wanted to see for myself how reliable it can be. This shows me that either I am looking at the data wrongly (whether that's a result of too many variables or too few variables), or there really isn't a reliable way to predict 4th down conversion rates based on historical data. Football is a sport that really allows any team to make a spectacular play at any time, and you never know when that might be, which is what makes the sport so fun.