Predicting NFL Game Outcomes

Hunter Grigsby

2024-10-07

#Problem 1: Research Topic and Dataset Selection We are trying to predict the NFL game outcomes based on various football statistics.

A: Our binary outcome (y-variable) for predicting is the result of the game (0=loss, 1=win).

B: Our predictor variables are Yard Margin, 3rd down conversion rate, Home/Away, penalties, and 3rd down conversion rate (3=high, 2=moderate, 1=low).

C: A win indicates whether the team won a game, with 1 representing a win and 0 a loss. 'Yards' refers to the total yards gained by the team in each game, while 'Yards Allowed' represents the total yards given up to the opposing teams. 'Yard Margin' is the difference between yards gained and yards allowed. The '3rd Down Conversion Rate' measures the team's performance on third downs, rated on a scale of 1 to 3, with 1 being the worst, 2 average, and 3 the best. 'Points' denote the team's total score in each game, while 'Points Allowed' indicates the points scored by the opposition. 'Point Margin' is the difference between points scored and points allowed. 'First Downs' represents the number of first downs the team achieved per game. 'Penalties' are the total penalties committed by the team. 'Home and Away' specifies the location of the game, with 1 for home and 2 for away.

Problem 2: Data Peparation

We made our own data set, we collected our variables and data from Pro Football Reference. We made the data set in Google Sheets. Data includes NFL statistics from the current NFL season (2024-2025, first 3 weeks). We also have 96 observations meaning we have 96 games of data.

A: For us to have the point margin and yard margin variables we needed to create this by subtracting our variables. For example, to get a point margin we get the points the team scored and then subtract the points allowed.

B: We prepared our dataset using data from this current NFL season, also from Pro Football Reference. We input the data and made new variables like yard margin and point margin. We also used a third-down conversion rate into a high, medium, and low rating system.

```r
# Helper packages
library(dplyr)     # for data wrangling
```

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

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

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

```r
library(ggplot2)   # for plotting
library(rsample)   # for data splitting
library(readr)

# Modeling packages
library(caret)     # for logistic regression modeling
```

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

```r
# Model interpretability packages
# Install 'vip' only if it is not installed
if (!requireNamespace("vip", quietly = TRUE)) {
  install.packages("vip")
}
library(vip)       # variable importance
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```r
# Evaluate model using ROC and AUC
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(ROCR)

file.exists("data_stats_362_NFL2.csv")
```

```
## [1] FALSE
```

```r
NFL2 <- read.csv("data stats 362 - NFL2.csv")

# Initial dimension
dim(NFL2)
```

```
## [1] 96 12
```

```r
# Response variable: Display the first few entries of 'WIN'
head(NFL2$WIN)
```

```
## [1] 1 0 0 1 0 0
```

```r
# Convert target variable 'WIN' to a binary factor
NFL2$WIN <- as.factor(ifelse(NFL2$WIN == TRUE, 1, 0))
NFL2$X3RD.DOWN.CONV.RATE <- as.factor(NFL2$X3RD.DOWN.CONV.RATE)
NFL2$HOME.AWAY <- as.factor(NFL2$HOME.AWAY)
```

Examine the Distribution of "WIN"

```
summary(NFL2$WIN)
```

```
##  0  1
## 48 48
```

Problem 3: Traditional Logistic Regression

```
LR <- glm(formula = WIN ~ YARD.MARGIN + X3RD.DOWN.CONV.RATE + HOME.AWAY + PENALTIES,
          family = "binomial", data = NFL2)
summary(LR)
```

```
##
## Call:
## glm(formula = WIN ~ YARD.MARGIN + X3RD.DOWN.CONV.RATE + HOME.AWAY +
##     PENALTIES, family = "binomial", data = NFL2)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.845211   0.837490  -1.009  0.31287
## YARD.MARGIN           0.007943   0.002423   3.278  0.00105 **
## X3RD.DOWN.CONV.RATE2  0.567481   0.557352   1.018  0.30860
## X3RD.DOWN.CONV.RATE3  0.202381   0.653152   0.310  0.75667
## HOME.AWAY2            0.234699   0.458818   0.512  0.60898
## PENALTIES             0.070085   0.092499   0.758  0.44864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 133.08  on 95  degrees of freedom
## Residual deviance: 112.43  on 90  degrees of freedom
## AIC: 124.43
##
## Number of Fisher Scoring iterations: 4
```

```
# Extract coefficients
coef(LR)
```

```
##          (Intercept)          YARD.MARGIN X3RD.DOWN.CONV.RATE2
##         -0.845211001          0.007943254          0.567480581
## X3RD.DOWN.CONV.RATE3           HOME.AWAY2            PENALTIES
##          0.202380644          0.234699499          0.070084714
```

```
# Calculate Odds Ratios (ORs)
ORs = exp(coef(LR))
CIs = exp(confint(LR))
```

```
## Waiting for profiling to be done...
```

```
cbind(ORs, CIs)
```

```
##                            ORs      2.5 %   97.5 %
## (Intercept)          0.4294667 0.07931864 2.190579
## YARD.MARGIN          1.0079749 1.00343310 1.013088
## X3RD.DOWN.CONV.RATE2 1.7638177 0.59327185 5.363222
## X3RD.DOWN.CONV.RATE3 1.2243139 0.33466296 4.432908
## HOME.AWAY2           1.2645287 0.51512370 3.146271
## PENALTIES            1.0725990 0.89513405 1.290686
```

Interpretation of Odds Ratios

```r
# Predict probabilities on the same dataset
traditional_probs <- predict(LR, type = "response")

# Convert probabilities to binary predictions
traditional_pred <- ifelse(traditional_probs > 0.5, 1, 0)

# Convert predictions and actual values to factors with matching levels
traditional_pred <- factor(traditional_pred, levels = c(0, 1))
NFL2$WIN <- factor(NFL2$WIN, levels = c(0, 1))

# Calculate confusion matrix
confusion_matrix_traditional <- confusionMatrix(traditional_pred, NFL2$WIN)
confusion_matrix_traditional
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 33 16
##          1 15 32
##
##                Accuracy : 0.6771
##                  95% CI : (0.5739, 0.769)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0003374
##
##                   Kappa : 0.3542
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.6875
##             Specificity : 0.6667
##          Pos Pred Value : 0.6735
##          Neg Pred Value : 0.6809
##              Prevalence : 0.5000
##          Detection Rate : 0.3438
##    Detection Prevalence : 0.5104
##       Balanced Accuracy : 0.6771
##
##        'Positive' Class : 0
##
```

Evaulate model using ROC and AUC

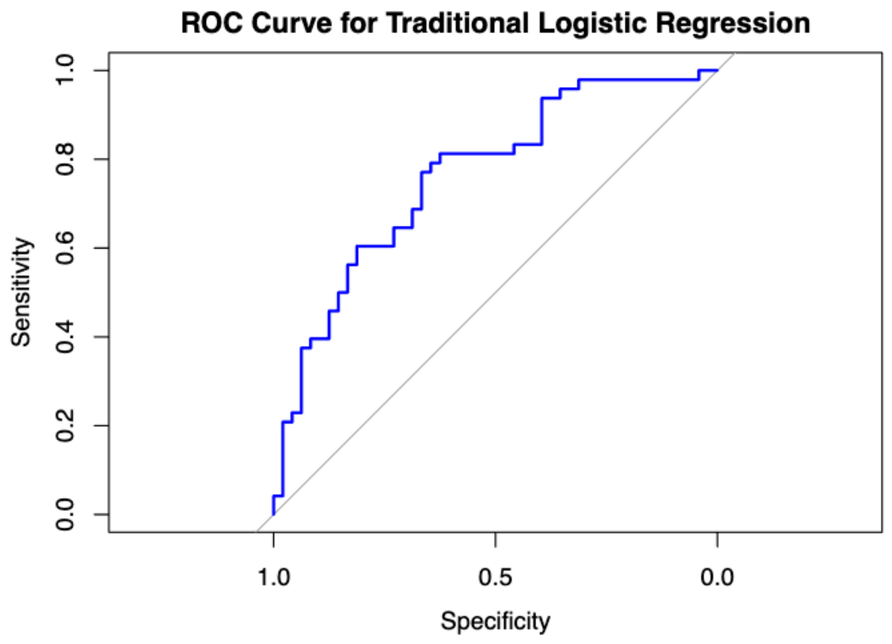
```r
# Evaluate model using ROC and AUC
roc_curve_traditional <- roc(NFL2$WIN, traditional_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_traditional <- auc(roc_curve_traditional)
auc_traditional
```

```
## Area under the curve: 0.7635
```

```r
# Plot ROC curve
plot(roc_curve_traditional, col = "blue",
     main = "ROC Curve for Traditional Logistic Regression")
```



Problem 4: Logistic Regression with Machine Learning

Train vs Test Split

```r
# Split the NFL data into training (70%) and test (30%) sets
set.seed(123)  # for reproducibility
nfl_split = initial_split(NFL2,
                          prop = 0.7,
                          strata = "WIN")
nfl_train = training(nfl_split)
nfl_test = testing(nfl_split)
```

Logistic Regression with Train/Test Split

```r
model1 <- glm(WIN ~ YARD.MARGIN + X3RD.DOWN.CONV.RATE + HOME.AWAY + PENALTIES,
              family = "binomial", data = nfl_train)
summary(model1)
```

```
##
## Call:
## glm(formula = WIN ~ YARD.MARGIN + X3RD.DOWN.CONV.RATE + HOME.AWAY +
##     PENALTIES, family = "binomial", data = nfl_train)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.982763   1.014472  -0.969  0.33267
## YARD.MARGIN           0.008359   0.003098   2.698  0.00698 **
## X3RD.DOWN.CONV.RATE2  0.861399   0.661457   1.302  0.19282
## X3RD.DOWN.CONV.RATE3  0.527612   0.844071   0.625  0.53192
## HOME.AWAY2            0.319925   0.572619   0.559  0.57636
## PENALTIES             0.061894   0.113867   0.544  0.58674
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.495  on 65  degrees of freedom
## Residual deviance: 76.123  on 60  degrees of freedom
## AIC: 88.123
##
## Number of Fisher Scoring iterations: 4
```

```r
# Convert coefficients to odds ratios
cbind(ORs = exp(coef(model1)), exp(confint(model1)))
```

```
## Waiting for profiling to be done...

##                            ORs      2.5 %   97.5 %
## (Intercept)          0.3742757 0.04663516 2.657641
## YARD.MARGIN          1.0083939 1.00259194 1.014992
## X3RD.DOWN.CONV.RATE2 2.3664688 0.65276562 8.979387
## X3RD.DOWN.CONV.RATE3 1.6948803 0.32148661 9.206149
## HOME.AWAY2           1.3770248 0.44894057 4.333454
## PENALTIES            1.0638491 0.85175453 1.339774
```

Predicited Probabilities

```r
# Predict probabilities on the test set
ml_probs <- predict(model1, newdata = nfl_test, type = "response")

# Convert probabilities to binary predictions
ml_pred <- ifelse(ml_probs > 0.5, 1, 0)

# Convert predictions and actual values to factors with matching levels
ml_pred <- factor(ml_pred, levels = c(0, 1))
nfl_test$WIN <- factor(nfl_test$WIN, levels = c(0, 1))
```

```r
# Calculate confusion matrix
confusion_matrix_ml <- confusionMatrix(ml_pred, nfl_test$WIN)
confusion_matrix_ml
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 11  4
##          1  4 11
##
##                Accuracy : 0.7333
##                  95% CI : (0.5411, 0.8772)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.008062
##
##                   Kappa : 0.4667
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7333
##             Specificity : 0.7333
##          Pos Pred Value : 0.7333
##          Neg Pred Value : 0.7333
##              Prevalence : 0.5000
##          Detection Rate : 0.3667
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7333
##
##        'Positive' Class : 0
##
```

```r
# Evaluate ML logistic model using ROC and AUC
roc_curve_ml <- roc(nfl_test$WIN, ml_probs)
```

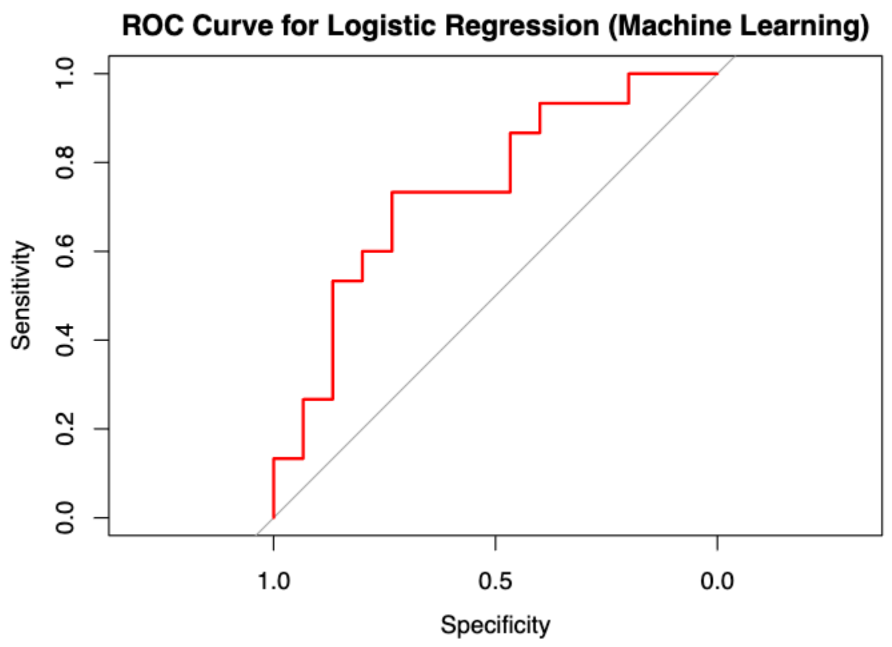
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc_ml <- auc(roc_curve_ml)
auc_ml
```

```
## Area under the curve: 0.7422
```

```r
# Plot ROC curve
plot(roc_curve_ml, col = "red",
     main = "ROC Curve for Logistic Regression (Machine Learning)")
```



Problem 5: Model Comparison

```r
# AIC for the traditional logistic regression model
aic_traditional <- AIC(LR)
print(aic_traditional)
```

```
## [1] 124.4271
```

```r
# AIC for the logistic regression model with train-test split
aic_ml <- AIC(model1)
print(aic_ml)
```

```
## [1] 88.12273
```

Problem 6: Conclusion

A: The machine learning approach using the train/test split performed better than the traditional logistic regression model, as indicated by a potentially higher AUC and a lower AIC. This improvement is due to the machine learning model's better handling of overfitting by evaluating performance on unseen test data. Although regularization was not explicitly applied in this analysis, the split strategy itself enhanced the model's predictive accuracy.

B: A key limitation of this analysis is the limited set of features used, potentially missing other influential factors like team-specific statistics. Additionally, no regularization was employed to address multicollinearity or overfitting. Future improvements could involve adding more features, applying regularization techniques, and using more advanced models like Random Forests or cross-validation to refine predictive accuracy.