R&D Intern Application Questionnaire

Karl Mbouombouo

2023-11-18

MODELING QUESTION ## Packages

```
library(Lahman)
library(dplyr)
library(caret)
library(glmnet)
teams = Teams
```

```
teams$R_diff <- teams$R - teams$RA
```

Linear model with the entire data

```
fit <- lm(W ~ R_diff, data = teams)

summary(fit)
```

```
##
## Call:
## lm(formula = W ~ R_diff, data = teams)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -69.504  -1.658   3.392   7.095  20.272
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 74.674530   0.235864  316.60   <2e-16 ***
## R_diff       0.096989   0.001835   52.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.95 on 3013 degrees of freedom
## Multiple R-squared:  0.481,  Adjusted R-squared:  0.4808
## F-statistic:  2793 on 1 and 3013 DF,  p-value: < 2.2e-16
```

Linear model with seasons after 1970

```
# Filter data for seasons after 1970
# Data after 1970 is called "teams2"
teams2 <- subset(teams, yearID > 1970)

# Clean the data, to use is for the model
colSums(is.na(teams2))
```

```r
fit2 <- lm(W ~ R_diff, data = teams2)

summary(fit2)
```

```
##
## Call:
## lm(formula = W ~ R_diff, data = teams2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.876  -1.322   1.787   4.683  16.472
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 78.684828   0.250514  314.09   <2e-16 ***
## R_diff       0.101123   0.002398   42.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.539 on 1448 degrees of freedom
## Multiple R-squared:  0.5511, Adjusted R-squared:  0.5508
## F-statistic:  1778 on 1 and 1448 DF,  p-value: < 2.2e-16
```

1.

```r
# R_diff Coefficient
coefficient_R_diff <- coef(fit2)["R_diff"]
cat("The R_diff coefficient is:", coefficient_R_diff, "\n")
```

```
## The R_diff coefficient is: 0.1011233
```

```r
# Calculate the number of runs needed for 1 extra win
runs <- 1 / coefficient_R_diff
runs
```

```
##   R_diff
## 9.888914
```

It means that for every additional run in run differential, you can expect approximately 0.1 additional wins.

The number of runs needed for 1 extra win is approximately 9.888914 (10 if rounded)

2.

```r
# Using Intercept
intercept <- coef(fit2)["(Intercept)"]
cat("The intercept is:", intercept,"\n")
```

```
## The intercept is: 78.68483
```

```r
# Using the function predict()
run0 <- predict(fit2, newdata = data.frame(R_diff = 0))
round(run0)
```

```
##  1
## 79
```

Based on the model's summary, the number of wins that a team is expected to have with a 0 run differential (x = 0) is 78.68483.

The model expects a team with a 0 run differential to win approximately: 79 games (rounded) in a full season.

3.

```
# Predict expected win total
expected_wins <- predict(fit2)

# Calculate residuals
residuals <- resid(fit2)

# Calculate standard deviation of residuals
sd_residuals <- sd(residuals)

# Calculate the z-score, we use the 7 for the difference
z_score <- 7 / sd_residuals

# Standard normal distribution to estimate the probability (round to 2 decimals)
p_7_wins <- pnorm(z_score, lower.tail = FALSE)
round(p_7_wins * 100, 2)
```

```
## [1] 23.15
```

The estimated probability that a team wins 7 games more than their expected win total is approximately 23.15%

4.

```
# Create a lagged variable for next season's wins
teams2$next_season_wins <- c(teams2$W[-1], NA)
# And remove rows with missing values in 'next_season_wins'
teams3 <- teams2[complete.cases(teams2$next_season_wins), ]

# Split the data into training and testing sets
set.seed(123)  # Set seed for reproducibility
index <- createDataPartition(teams3$next_season_wins, p = 0.8, list = FALSE)
train_data <- teams3[index, ]
test_data <- teams3[-index, ]

# Create a control object for cross-validation
ctrl <- trainControl(method = "cv", number = 5)  # 5-fold cross-validation

# Fit linear regression models with cross-validation
model_current_w <- train(next_season_wins ~ W, data = train_data,
                         method = "lm", trControl = ctrl)
model_r_diff <- train(next_season_wins ~ R_diff, data = train_data,
                      method = "lm", trControl = ctrl)
model_r <- train(next_season_wins ~ R, data = train_data,
                 method = "lm", trControl = ctrl)
model_ra <- train(next_season_wins ~ RA, data = train_data,
                  method = "lm", trControl = ctrl)
```

```r
# Print the cross-validated model summaries
print(model_current_w)
```

```
## Linear Regression
##
## 1161 samples
##    1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 929, 928, 929, 928, 930
## Resampling results:
##
##   RMSE      Rsquared    MAE
##   13.54104  0.08604448  10.86232
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
print(model_r_diff)
```

```
## Linear Regression
##
## 1161 samples
##    1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 929, 928, 929, 929, 929
## Resampling results:
##
##   RMSE      Rsquared     MAE
##   14.05641  0.009792585  10.85246
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
print(model_r)
```

```
## Linear Regression
##
## 1161 samples
##    1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 928, 929, 930, 929, 928
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   13.27702  0.1236989  10.77117
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
print(model_ra)
```

```
## Linear Regression
##
```

```
## 1161 samples
##    1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 930, 929, 929, 927, 929
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   12.80965  0.1847015  10.42654
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

RMSE for the model predicting next season's wins based on current wins: 13.55977

RMSE for the model predicting next season's wins based on run differential model: 14.05394

RMSE for the model predicting next season's wins based on runs scored model: 13.26109

RMSE for the model predicting next season's wins based on runs allowed model: 12.79468

Based on RMSE alone, model_ra appears to be the best-performing model, because it has the lowest RMSE (12.79468).