Statistical_Modeling_Report

J.Hastings

2025-06-07

Statistical Modelling

> **Objective:** The objective of the statistical modeling section is to evaluate how well entrance assessment scores, summarized as a success score, predict the likelihood of program completion using logistic regression. The model also explores whether other variables can contribute additional predictive value to the model.

```r
# Load and format the data
my_data = read.csv("/Users/oakmoreroadinc./Desktop/Data Science /Portfolio Data Science /R_script.R/data

head(my_data)
```

```
##   X Youth.ID Gender  PWS   DL   SC   RC  HMM  WSL  CEP   LF Days_in_Program
## 1 0        1      M 3.80 4.47 4.41 4.33 3.96 4.10 5.00 5.00             264
## 2 1        2      M 4.00 4.41 4.53 4.78 3.70 4.40 4.10 4.00             467
## 3 2        3      M 4.30 4.65 4.29 4.78 3.78 4.60 4.11 4.50              49
## 4 3        4      M 3.55 2.24 2.71 3.83 2.09 2.70 2.11 3.25             908
## 5 4        5      F 4.90 4.71 4.76 4.83 3.35 4.65 4.78 5.00             241
## 6 5        6      M 4.50 4.59 4.65 4.83 4.09 4.35 3.89 4.62             495
##   Category Result.Score
## 1     Good            1
## 2     Good            1
## 3      Bad            0
## 4     Good            1
## 5      Bad            0
## 6      Bad            0
```

Created a new variable called **success.score** by using coefficients from the model because the variables by itself did not show significance.

```r
# Fit logistic regression model
coe_model <- glm(Result.Score ~ PWS + DL + SC + RC + HMM + WSL + CEP + LF,
                 data = my_data, family = binomial)

# View model summary
summary(coe_model)
```

```
##
## Call:
## glm(formula = Result.Score ~ PWS + DL + SC + RC + HMM + WSL +
```

```
##     CEP + LF, family = binomial, data = my_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  33.4536    18.5200   1.806   0.0709 .
## PWS          -2.2796     2.2256  -1.024   0.3057
## DL            0.7933     3.8338   0.207   0.8361
## SC           -1.0448     3.5996  -0.290   0.7716
## RC           -8.6021     6.2500  -1.376   0.1687
## HMM           1.4400     2.3165   0.622   0.5342
## WSL           6.6385     4.9175   1.350   0.1770
## CEP          -0.1141     2.2017  -0.052   0.9587
## LF           -3.5557     3.2343  -1.099   0.2716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29.767  on 21  degrees of freedom
## Residual deviance: 17.329  on 13  degrees of freedom
## AIC: 35.329
##
## Number of Fisher Scoring iterations: 6
```

```r
# Extract coefficients
intercept <- coef(coe_model)[1]
b1 <- coef(coe_model)[2]
b2 <- coef(coe_model)[3]
b3 <- coef(coe_model)[4]
b4 <- coef(coe_model)[5]
b5 <- coef(coe_model)[6]
b6 <- coef(coe_model)[7]
b7 <- coef(coe_model)[8]
b8 <- coef(coe_model)[9]

# Calculate success score manually
my_data$success.score <- intercept +
                      b1 * my_data$PWS +
                      b2 * my_data$DL +
                      b3 * my_data$SC +
                      b4 * my_data$RC +
                      b5 * my_data$HMM +
                      b6 * my_data$WSL +
                      b7 * my_data$CEP +
                      b8 * my_data$LF


#creating Bernoulli full model with engineered "success.score"
my.model <- glm(Result.Score ~ success.score,
                data = my_data, family = binomial)


#splitting with training and test set
set.seed(123)
```

```r
n <- nrow(my_data)
train_idx <- sample(1:n, size = 0.7 * n)

train_set <- my_data[train_idx, ]
test_set <- my_data[-train_idx, ]
cat("Training set rows:", nrow(train_set), "\n")
```

```
## Training set rows: 15
```

```r
cat("Test set rows:", nrow(test_set), "\n")
```

```
## Test set rows: 7
```

```r
summary(my.model)
```

```
##
## Call:
## glm(formula = Result.Score ~ success.score, family = binomial,
##     data = my_data)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -8.206e-09  6.073e-01   0.000   1.0000
## success.score  1.000e+00  4.707e-01   2.125   0.0336 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29.767  on 21  degrees of freedom
## Residual deviance: 17.329  on 20  degrees of freedom
## AIC: 21.329
##
## Number of Fisher Scoring iterations: 6
```

```r
cat("Training set rows:", nrow(train_set), "\n")
```

```
## Training set rows: 15
```

```r
cat("Test set rows:", nrow(test_set), "\n")
```

```
## Test set rows: 7
```

success.score, is significant from the summary. p-value = 0.03 which is less then 0.05.

Residual deviance to test goodness of fit can't be used because I have only 1 trial per observation(n<5). To compensate, I will compare my model to the null deviance to see if my predictors are meaningful. I will combine Mean Squared Prediction Error which is not technically a goodness of fit test but due to small sample size I can get a good idea how well my models predicted probabilities match the actual outcomes. Then I will create a confusion matrix to see how accurate my model is on unseen data.

Hypothesis Test for Model Fit
- **H** : The model with just **parameters fits the data well**.
- **H** : The model with just **parameters does not** fit the data well.

```r
anova(my.model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Result.Score
##
## Terms added sequentially (first to last)
##
##
##               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                             21     29.767
## success.score  1   12.438        20     17.329 0.0004206 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#creating binomial train model with engineered "success.score"
train.model <- glm(Result.Score ~ success.score,
                data = train_set, family = binomial)

#Mean square prediction error
p_binomial <- predict(train.model, newdata = test_set, type = "response")
MSPE.model <- mean((test_set$Result.Score - p_binomial)^2)

#confusion matrix
# Step 1: Predict probabilities on the test set using the test_set data:
pred_probs <- predict(train.model, newdata = test_set, type = "response")

# Step 2: Convert probabilities to class predictions (threshold = 0.5)
pred_class <- ifelse(pred_probs >= 0.5, 1, 0)

# Step 3: Create confusion matrix
conf_matrix <- table(Predicted = pred_class, Actual = test_set$Result.Score)
```

Results(Null Deviance test):

- My model shows a significant difference against the null deviance(rejecting the null).My predictor shows that it has some predictive power.

```r
#Print Mean Squared Error Results
print(MSPE.model)
```

```
## [1] 0.05284237
```

MSPE Results:

- The Mean Squared Prediction Error for the test set is **0.05**.

- A value close to 0 means the models predicted probabilities are closely matching the true outcomes.
- This score suggests strong, well-calibrated model performance on this test set.

```
#Create confusion matrix
conf_matrix <- table(Predicted = pred_class, Actual = test_set$Result.Score)
#Display confusion matrix
print(conf_matrix)
```

```
##          Actual
## Predicted 0 1
##         0 1 0
##         1 0 6
```

```
#Evaluating statistics
# Assign values
TP <- 6
TN <- 1
FP <- 0
FN <- 0

# Accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)

# Precision (Positive Predictive Value)
precision <- TP / (TP + FP)

# Recall (Sensitivity or True Positive Rate)
recall <- TP / (TP + FN)

# F1 Score (harmonic mean of precision and recall)
f1_score <- 2 * (precision * recall) / (precision + recall)

# Display results
cat("Accuracy:", round(accuracy, 3), "\n")
```

```
## Accuracy: 1
```

```
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 1
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 1
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 1
```

Interpretation:
All scores are 1.00, which means the model made perfect predictions on the test set: - **Accuracy** of 1.00 means every prediction was correct. - **Precision** of 1.00 means every positive prediction was actually positive (no false alarms). - **Recall** of 1.00 means the model found all actual positives (no misses). - **F1 Score** of 1.00 is the harmonic mean of precision and recall—also perfect.

Note: With a small test set, perfect scores may not reflect real-world performance. Always validate on more data when possible.

```
###
odds_ratio <- exp(coef(my.model)["success.score"])
cat(sprintf("A one-unit increase in success.score is associated with an odds ratio of %.3f.\n", odds_ra
```

```
## A one-unit increase in success.score is associated with an odds ratio of 2.718.
```

```
cat("This means the odds of completing the program are multiplied by that factor for each one-unit incre
```

```
## This means the odds of completing the program are multiplied by that factor for each one-unit increas
```

```
cat("This effect is statistically significant (p = 0.0336).\n")
```

```
## This effect is statistically significant (p = 0.0336).
```

```
### Quantifying uncertainty in parameters:
# Get 95% CI in log-odds
ci_log <- confint(my.model)["success.score", ]
```

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

```
# Convert to odds ratio
ci_odds <- exp(ci_log)

coef_value <- coef(my.model)["success.score"]
round(coef_value, 3)
```

```
## success.score
##             1
```

```
#in odds
exp(round(coef_value, 3))
```

```
## success.score
##      2.718282
```

```
# Display results
cat("95% CI (log-odds): [", round(ci_log[1], 3), ",", round(ci_log[2], 3), "]\n")
```

```
## 95% CI (log-odds): [ 0.306 , 2.247 ]
```

```r
cat("95% CI (odds ratio): [", round(ci_odds[1], 3), ",", round(ci_odds[2], 3), "]\n")
```

```
## 95% CI (odds ratio): [ 1.358 , 9.455 ]
```

Based on the profile likelihood method (used due to small sample size), the 95% confidence interval for the **success.score** coefficient is [**0.306, 2.247**] in log-odds. The model estimated a coefficient of **1**, which lies well within this interval. Since the range does not include zero, the effect is considered statistically significant. In odds ratio terms, this corresponds to a range of [**1.358, 9.455**], indicating a strong positive association between **success.score** and program completion. Specifically, the model estimates that for each one-unit increase in **success.score**, the odds of successfully completing the program increase by a factor of approximately **2.72** — meaning the odds are nearly three times higher for someone with a one-point higher success score.

```r
# Extract intercept and slope from the model
intercept <- coef(my.model)[1]
slope <- coef(my.model)[2]

cat("Intercept ( ):", round(intercept, 4), "\n")
```

```
## Intercept ( ): 0
```

```r
cat("Coefficient for success.score ( ):", round(slope, 4), "\n")
```

```
## Coefficient for success.score ( ): 1
```

```r
# Define the sigmoid function
sigmoid <- function(x) {
  return(1 / (1 + exp(-x)))
}

# Choose a success.score to test
score <- -1

# Calculate log-odds
log_odds <- intercept + slope * score

# Convert log-odds to probability
prob <- sigmoid(log_odds)

cat("Log-odds:", round(log_odds, 4), "\n")
```

```
## Log-odds: -1
```

```r
cat("Predicted probability:", round(prob, 4), "\n")
```

```
## Predicted probability: 0.2689
```

```r
cat("Model: logit(p) =   +   × success.score\n")
```

```
## Model: logit(p) =   +   × success.score
```

```r
cat(sprintf("With success.score = %d:\n", score))
```

```
## With success.score = -1:
```

```r
cat(sprintf("logit(p) = %.4f + %.4f × %d = %.4f\n", intercept, slope, score, log_odds))
```

```
## logit(p) = -0.0000 + 1.0000 × -1 = -1.0000
```

```r
cat(sprintf("p = 1 / (1 + exp(-%.4f)) = %.4f\n", log_odds, prob))
```

```
## p = 1 / (1 + exp(--1.0000)) = 0.2689
```

```r
if (prob >= 0.5) {
  cat("Classified as: SUCCESS (1)\n")
} else {
  cat("Classified as: FAILURE (0)\n")
}
```

```
## Classified as: FAILURE (0)
```

Model Interpretation at success.score = -1

To demonstrate how the model classifies individuals, we used the logistic regression equation:

$$\text{logit}(p) = \beta_0 + \beta_1 \cdot \text{success.score}$$

Using the estimated model coefficients:

- Intercept ( ) = 0
- Coefficient for `success.score` ( ) = 1

We evaluated the model for an individual with a `success.score` of -1:

- Log-odds $= 0 + 1 \cdot (-1) = -1$
- Probability $= \frac{1}{1+e^1} \approx 0.2689$

Based on a classification threshold of 0.5, the model predicts that this individual **will not complete the program** (classified as 0).

```r
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sequoia 15.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK ve
```

```
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] DescTools_0.99.60 pROC_1.18.5       ggplot2_3.5.2     dplyr_1.1.4
## [5] readxl_1.4.5
##
## loaded via a namespace (and not attached):
##  [1] generics_0.1.4     class_7.3-23       lattice_0.22-7     hms_1.1.3
##  [5] digest_0.6.37      magrittr_2.0.3     evaluate_1.0.4     grid_4.5.1
##  [9] RColorBrewer_1.1-3 mvtnorm_1.3-3      fastmap_1.2.0      cellranger_1.1.0
## [13] plyr_1.8.9         Matrix_1.7-3       e1071_1.7-16       httr_1.4.7
## [17] scales_1.4.0       cli_3.6.5          rlang_1.1.6        expm_1.0-0
## [21] withr_3.0.2        yaml_2.3.10        rootSolve_1.8.2.4  tools_4.5.1
## [25] tzdb_0.5.0         lmom_3.2           gld_2.6.7          Exact_3.3
## [29] forcats_1.0.0      boot_1.3-31        vctrs_0.6.5        R6_2.6.1
## [33] proxy_0.4-27       lifecycle_1.0.4    fs_1.6.6           MASS_7.3-65
## [37] pkgconfig_2.0.3    pillar_1.10.2      gtable_0.3.6       data.table_1.17.6
## [41] glue_1.8.0         Rcpp_1.0.14        haven_2.5.5        xfun_0.52
## [45] tibble_3.3.0       tidyselect_1.2.1   rstudioapi_0.17.1  knitr_1.50
## [49] farver_2.1.2       htmltools_0.5.8.1  rmarkdown_2.29     readr_2.1.5
## [53] compiler_4.5.1
```