Statistical Methods - HW7

2023-03-06

Problem 1

Find the dataset Placekick.csv in the Datasets subfolder. Use this dataset to build a logistic regression model to estimate the probability of success for a placekick.

```
require(car)
# Load data
data <- read.csv("Placekick.csv")
```

a. Fit a logistic regression model with good as response and distance as predictor. Interpret the fitted model coefficients and visualize the model fit.

```
# Fit logistic regression model with distance as predictor
model_distance <- glm(good ~ distance, data = data, family = binomial(link = logit))

# Summary of model coefficients
summary(model_distance)
```

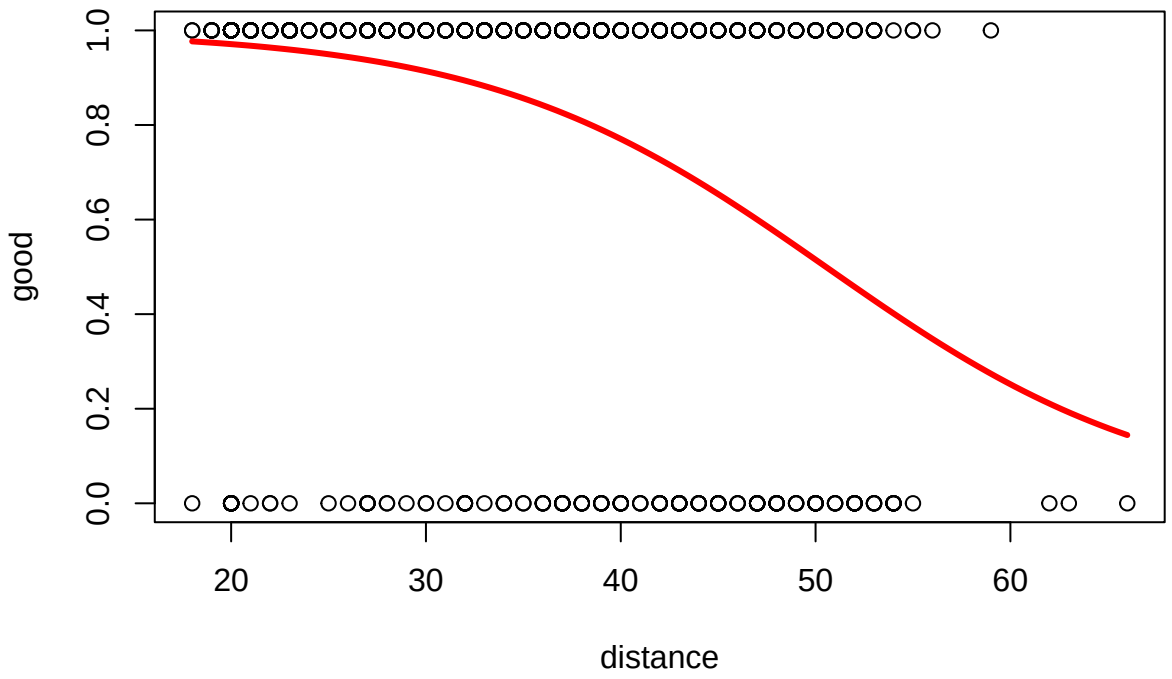
```
##
## Call:
## glm(formula = good ~ distance, family = binomial(link = logit),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7441   0.2425   0.2425   0.3801   1.6092
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.812080   0.326277   17.81   <2e-16 ***
## distance    -0.115027   0.008339  -13.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1013.43  on 1424  degrees of freedom
## Residual deviance:  775.75  on 1423  degrees of freedom
## AIC: 779.75
##
## Number of Fisher Scoring iterations: 6
```

```
summary(model_distance)$coefficients
```

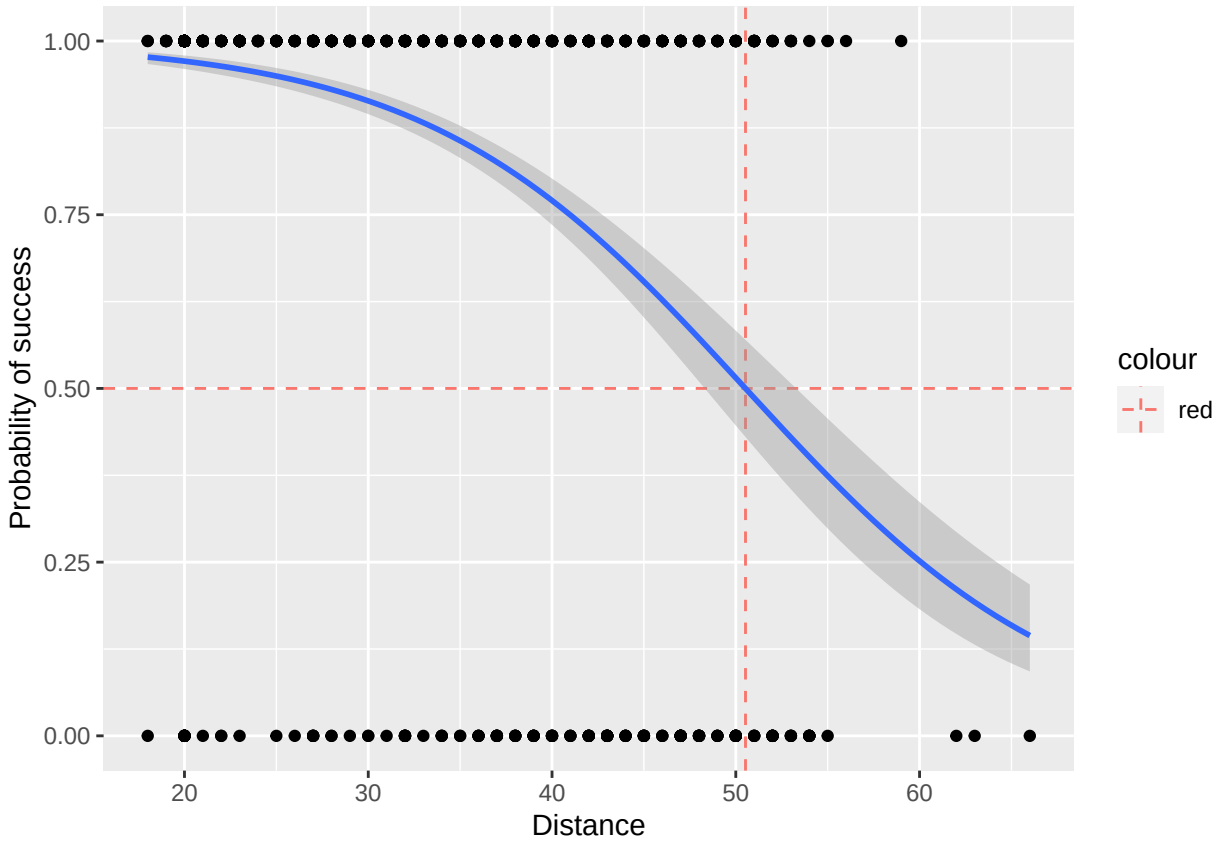
```
##               Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)  5.8120798 0.326277110  17.81332 5.570180e-71
## distance    -0.1150267 0.008339062 -13.79372 2.780574e-43
```

```r
# Plot the fitted model using curve() function
plot(data$distance, data$good, xlab = 'distance', ylab = 'good')
curve(expr = predict(object = model_distance,
                     newdata = data.frame(distance = x), type = "response") ,
      col = "red", add = TRUE ,lwd = 3)

# Plot predicted probability of success against distance using ggplot
dec <- 5.8120798/0.1150267
library(ggplot2)
```




```r
ggplot(data, aes(x = distance, y = good)) +
  geom_point() +
  geom_hline(aes(yintercept = 0.5, colour = "red"), linetype = 2) +
  geom_vline(aes(xintercept = dec, colour = "red"), linetype = 2) +
  geom_smooth(method = "glm", method.args = list(family = binomial)) +
  labs(x = "Distance", y = "Probability of success")
```



In logistic regression, the coefficients represent the change in the log odds of the response variable (i.e., the dependent variable) for a one-unit increase in the corresponding predictor variable (i.e., the independent variable), holding all other predictor variables constant.

In this case, for one unit increase in distance, the odds decreases by a factor of `exp(-0.1150267) = 0.89134234474`. This is about 11% decrease in odds of the response `good`.

From the graph, if we assume the threshold as 0.5, then we can infer that the decision boundary for this model is obtained when the distance is approximately 51. This implies that the distance < 51 will be predicted as 1 and > 51 will be predicted as 0. The model doesn't seem to fit the data well. This is expected as we only have one predictor in this model.

b. Now consider all predictors. Apply the forward selection algorithm to compute the forward selection path from 'intercept only' to 'full model' and chooses the model on that path that minimizes the AIC.

```
library(MASS)
# Fit intercept only model
int_model <- glm(good ~ 1, data = data, family = binomial(link = logit))
# Fit full model
final_model <- glm(good ~ week + distance + change + elap30 + PAT + type + field + wind,
                   data = data, family = binomial(link = logit))

# model_selection <- step(int_model, scope = formula(final_model), direction="forward")
```

```r
# Model selection using forward selection algorithm
model_selection <- stepAIC(int_model, scope = formula(final_model), direction="forward")
```

```
## Start:  AIC=1015.43
## good ~ 1
##
##            Df Deviance     AIC
## + distance  1   775.75  779.75
## + PAT       1   834.41  838.41
## + change    1   989.15  993.15
## + elap30    1  1007.71 1011.71
## + wind      1  1010.59 1014.59
## + week      1  1011.24 1015.24
## + type      1  1011.39 1015.39
## <none>         1013.43 1015.43
## + field     1  1012.98 1016.98
##
## Step:  AIC=779.75
## good ~ distance
##
##          Df Deviance    AIC
## + PAT     1   762.41 768.41
## + change  1   770.50 776.50
## + wind    1   772.53 778.53
## <none>        775.75 779.75
## + week    1   773.86 779.86
## + type    1   775.67 781.67
## + elap30  1   775.68 781.68
## + field   1   775.74 781.74
##
## Step:  AIC=768.41
## good ~ distance + PAT
##
##          Df Deviance    AIC
## + change  1   759.33 767.33
## + wind    1   759.66 767.66
## <none>        762.41 768.41
## + week    1   760.57 768.57
## + type    1   762.25 770.25
## + elap30  1   762.31 770.31
## + field   1   762.41 770.41
##
## Step:  AIC=767.33
## good ~ distance + PAT + change
##
##          Df Deviance    AIC
## + wind    1   756.69 766.69
## + week    1   757.26 767.26
## <none>        759.33 767.33
## + elap30  1   759.11 769.11
## + type    1   759.13 769.13
## + field   1   759.33 769.33
##
```

```
## Step:  AIC=766.69
## good ~ distance + PAT + change + wind
##
##          Df Deviance    AIC
## <none>        756.69 766.69
## + week    1   755.07 767.07
## + type    1   756.06 768.06
## + elap30  1   756.43 768.43
## + field   1   756.66 768.66
```

```
writeLines(c("", "", "The selected model is:"))
```

```
##
##
## The selected model is:
```

```
model_selection
```

```
##
## Call:  glm(formula = good ~ distance + PAT + change + wind, family = binomial(link = logit),
##     data = data)
##
## Coefficients:
## (Intercept)     distance          PAT       change         wind
##     4.75157     -0.08724      1.22992     -0.33505     -0.52344
##
## Degrees of Freedom: 1424 Total (i.e. Null);  1420 Residual
## Null Deviance:       1013
## Residual Deviance: 756.7     AIC: 766.7
```

c. Consider the model selected by the forward selection algorithm. Compute the decision boundary when the decision threshold for the probability of success is 0.5.

```
# Compute log odds for the probability of success = 0.5
logistic_inv <- function(x) { log(x / (1 - x)) }
log_odds <- logistic_inv(0.5)

# Coefficients of selected model
beta <- coef(model_selection)

# Compute the decision boundary
pred <- names(model_selection$coefficients)
db <- ""
for(i in 1:length(beta)){
  if(i == 1) {
    db <- paste(db, round(beta[i], 3))
  } else {
    db <- paste(db, "+ (", round(beta[i], 3), "*", pred[i], ")")
  }
}

# Print the decision boundary
db <- paste(db, "= ", log_odds)
writeLines(c("The decision boundary is computed by solving the following equation:", db))
```

```
## The decision boundary is computed by solving the following equation:
##  4.752 + ( -0.087 * distance ) + ( 1.23 * PAT ) + ( -0.335 * change ) + ( -0.523 * wind ) =  0
```

Problem 2

a. Write a function bootGLM(x, y, B=1000) that resamples observations and returns standard errors for each of the predictor variables (when the others are present in the model) in a logistic model.

```
# Definition of bootGLM function
bootGLM <- function(x, y, B=1000) {
  y_len = length(y)
  x_len <- length(x) + 1
  # Vector to store standard errors for each predictor
  se <- numeric(length(x))
  # Matrix to store coefficients from each of the B samplings
  beta_boot <- matrix(NA, nrow = B, ncol = x_len)
  for (i in 1:B) {
    # Resampling
    indices = sample(1:y_len, y_len, replace=TRUE)
    x_boot = x[indices,]
    y_boot = y[indices]
    # Dataframe with both x_boot anf y_boot values
    dt = cbind(x_boot, y_boot)
    # Fit the logistic regression model
    mdl <- glm(y_boot ~ ., data = dt, family = binomial(link = logit))
    # Store the coefficients of fitted model
    beta_boot[i,] = coef(mdl)
  }
  # Compute standard error using the standard deviation of the coefficients
  for (j in 1: x_len) {
    se[j] <- sd(beta_boot[,j])
  }
  # Return the standard errors
  return(se)
}
```

b. Consider the model selected by the forward selection algorithm from Problem 1(b). Apply your bootGLM, and compare with the standard errors returned by the summary function.

```
# Get the predictors of the selected model
predictors <- names(model_selection$coefficients)
n <- predictors[2:length(predictors)]

# x and y of the model selected by the forward selection algorithm
x <- data[c(n)]
y <- data$good

# Standard errors computed using the bootGLM function
boot_se <- bootGLM(x, y)
bdt = rbind(predictors, round(boot_se, 4))
writeLines(c("The standard errors computed using bootGLM function:"))
```

```
## The standard errors computed using bootGLM function:
```

```
print(bdt)
```

```
##            [,1]          [,2]       [,3]     [,4]     [,5]
## predictors "(Intercept)" "distance" "PAT"    "change" "wind"
##            "0.4803"      "0.0113"   "0.3941" "0.191"  "0.2829"
```

```r
# Standard errors returned by the summary function of the selected model
mdl_se <- summary(model_selection)$coefficients[, "Std. Error"]
writeLines(c("", "The standard errors returned by the summary function:"))
```

```
##
## The standard errors returned by the summary function:
```

```r
print(round(mdl_se, 4))
```

```
## (Intercept)    distance         PAT      change        wind
##      0.4720      0.0111      0.3847      0.1933      0.3132
```

From the above results, we can see that the standard errors computed using the bootGLM function using 1000 samples and the standard errors returned by the summary function of the selected model using forward selection algorithm are slightly different.

This is because bootstraping gives us a approximated value of standard error by estimating the standard deviation of the coefficients of models over 1000 samples taken with replacement which indicates the behavior of the population.

Contribution Statement:

1. Swetha Arunraj (PID: A59019948):

- Coded Problem 1
- Added comments to the code of Problem 1
- Added description/comments to Problem 1
- Contributed to embed the codes in R Markdown and create the final PDF

2. Harin Raja Radha Krishnan (PID: A59019874):

- Coded Problem 2
- Added comments to the code of Problem 2
- Added description/comments to Problem 2
- Contributed to embed the codes in R Markdown and create the final PDF