PES University, Bangalore

Established under the Karnataka Act No. 16 of 2013

UE20CS312 - Data Analytics

Worksheet 2c - Logistic Regression

Anushka Hebbar - anushkahebbar@pesu.pes.edu

Prerequisites

To download the data required for this worksheet, visit this Github link. Use the following libraries and read the dataset:

```r
library(tidyverse)
library(InformationValue)
char_preds <- read.csv('got_characters.csv')
```

The Logit Model

The linear regression techniques discussed so far are used to model the relationship between a quantitative response variable and one or more explanatory variables. When the response variable is categorical, other techniques are more suited to the task of classification.

The **logit model**, or **logistic model** models the probability, p, that a dichotomous (binary), dependent variable takes on one of two possible outcomes. It achieves this by setting the natural logarithm of the odds of the response variable, called the *log-odds* or *logit*, as a linear function of the explanatory variables.

$$Z_i = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x + ... + \beta_n x_n \text{ for } p \in (0, 1)$$

Here, Z_i is the log-odds of the response variable taking on a value with probability p.

Logistic regression is an algorithm that estimates the parameters, or coefficients, of the linear combination of the logit model. In this worksheet, we will classify a certain binary feature of a dataset using logistic regression.

A Song of Ice and Fire - GoT Character Dataset

A Song of Ice and Fire by George RR Martin is a series of epic fantasy novels that is popularly known for its TV show adaptation, *Game of Thrones*. The show is well known for its vastly complicated political landscape, large number of characters, and its frequent character deaths.

The given dataset contains comprehensive information on characters from the book series till the 5th book, *A Dance with Dragons*. The data was created by the team at A Song of Ice and Data who scraped it from the Wiki of Ice and Fire.

This worksheet will focus on using Logistic Regression to predict whether a character in the SoIaF world remains alive by the end of the 5th book, which is captured by the feature `actual`.

Data Dictionary

```
actual - Whether the character is alive in the consequent books
        (0 if dead, 1 if alive)
name - Name of the character
title - Character's title
male - Gender of the character (1 if male, 0 otherwise)
culture - Culture the character is from
dateofBirth - Character's DoB
mother - Name of the character's mother
father - Name of the character's father
heir - Name of the character's heir
spouse - Name of the character's spouse
book1 - Whether the character appears in Book 1, Game of Thrones
book2 - Whether the character appears in Book 2, A Clash of Kings
book3 - Whether the character appears in Book 3, A Storm of Swords
book4 - Whether the character appears in Book 4, A Feast for Crows
book5 - Whether the character appears in Book 5, A Dance with Dragons
isAliveMother - Whether the character's mother is alive
isAliveFather - Whether the character's father is alive
isAliveHeir - Whether the character's heir is alive
isAliveSpouse - Whether the character's spouse is alive
isMarried - Whether the character is married
isNoble - Whether the character belongs to a noble family
boolDeadRelations - Whether one of the character's relations is dead
numDeadRelations - Count of the character's relations that are dead
isPopular - Whether the character is popular
popularity - Score of the character's popularity
```

Points

The problems for this part of the worksheet are for a total of 10 points, with a non-uniform weightage.

- *Problem 1* : 1 point
- *Problem 2* : 2 points
- *Problem 3* : 2 points
- *Problem 4* : 3 points
- *Problem 5* : 2 points

Problems

Problem 1 (1 point)

How many characters from the SoIaF world does this dataset contain information on? Calculate the percentage of missing data for each column of the dataset and print them out in descending order as a dataframe.

Hint: Make sure to capture data from both missing values in numeric fields and empty strings in descriptive fields. Convert all missing placeholders to type NA.

Solution 1

```r
sprintf("Number of characters listed in the DF: %d", nrow(char_preds))
```

```
## [1] "Number of characters listed in the DF: 1946"
```

```r
# Replace missing strings with NA
char_preds[char_preds == "" | char_preds == " "] <- NA
# Make a DF for number of null vals in each column
df <- data.frame(colSums(is.na(char_preds)) / nrow(char_preds) * 100)
# Rename column to be something recognizable
colnames(df) <- c('% Missing')
# Reset index and make a feature of col names
df$Columns <- rownames(df)
rownames(df) <- NULL
# Order in decreasing order of percentages
dff <- df[order(df$'% Missing', decreasing=TRUE),]
rownames(dff) <- NULL
dff
```

```
##    % Missing          Columns
## 1   98.92086           mother
## 2   98.92086    isAliveMother
## 3   98.81809             heir
## 4   98.81809      isAliveHeir
## 5   98.66393           father
## 6   98.66393    isAliveFather
## 7   85.81706           spouse
## 8   85.81706    isAliveSpouse
## 9   77.74923      dateOfBirth
## 10  77.74923              age
## 11  65.21069          culture
## 12  51.79856            title
## 13  21.94245            house
## 14   0.00000                X
## 15   0.00000             S.No
## 16   0.00000           actual
## 17   0.00000             name
## 18   0.00000             male
## 19   0.00000            book1
## 20   0.00000            book2
## 21   0.00000            book3
## 22   0.00000            book4
## 23   0.00000            book5
## 24   0.00000        isMarried
## 25   0.00000          isNoble
## 26   0.00000  numDeadRelations
## 27   0.00000 boolDeadRelations
## 28   0.00000        isPopular
## 29   0.00000       popularity
```

Problem 2 (2 points)

Step 1 What are the inferences you can draw from the output dataframe of the previous problem? How can we handle columns with extremely high proportions of missing data before moving on?

Hint: Does missing data in a column tell you something about the target variable (`actual`)? If not, set a missing percentage threshold of 80%, deeming the column as having insufficient data, and drop the column.

Step 2 Impute missing data in the remaining numeric features with a reasonable statistic. Make sure you observe the distribution of the data in the columns to pick out a reasonable statistic. For categorical variables, convert the columns to numeric features. *Hint: Use the `unclass` function and impute missing categorical feature values with a -1.*

Bonus

After plotting the `age` column, do you notice any discrepancies in the graph? What do you think might have given rise to a such a distribution?

Solution 2

None of the columns with more than 80% missing data tell us anything substantial about the target feature because of their absence.

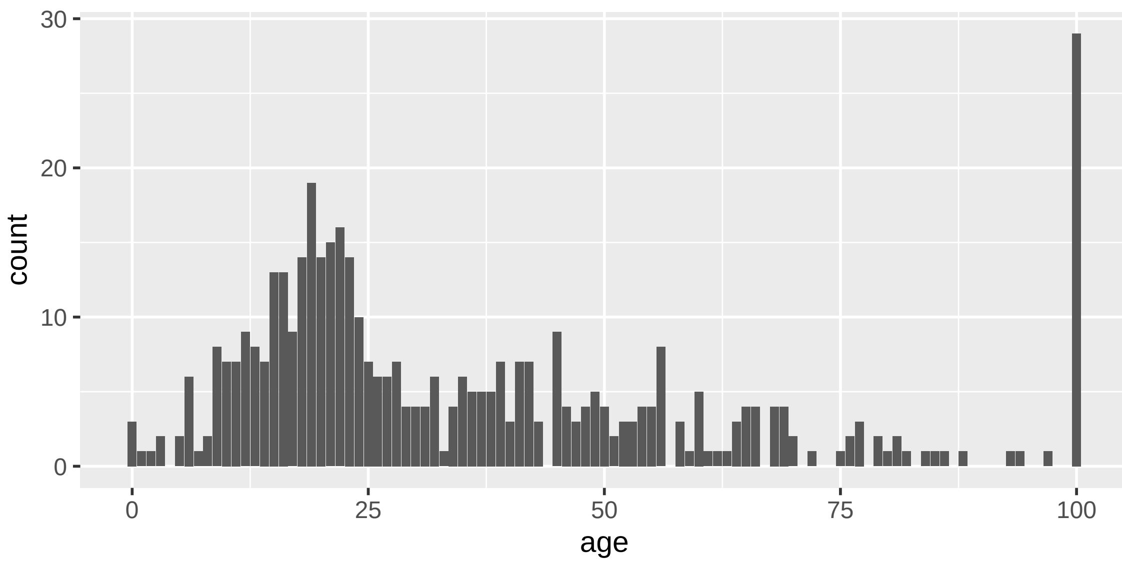
```
remove <- dff[dff$'% Missing' > 80,'Columns']
char_preds <- char_preds[!names(char_preds) %in% remove]
```

The columns left to be dealt with are `age`, `culture`, `title` and `house`.

For `age`,

```
ggplot(char_preds, aes(x=age)) + geom_bar()
```

```
## Warning: Removed 1513 rows containing non-finite values (stat_count).
```



This distribution has a lot of samples with 100 as the age, which might indicate that in the case of missing data about `age`, 100 was allotted by default. To fix this, calculate the mode of the other samples of `age` and replace both the samples with `100` and `NA` as values for `age` by the mode.

```
# Function to calculate mode
get_mode <- function(x) {
    mode0 <- names(which.max(table(x)))
    if(is.numeric(x)) return(as.numeric(mode0))
    mode0
}
# Mode from the set of samples that are not wrong
age_without_100 <- char_preds[char_preds$age < 100, 'age']
age_to_replace <- get_mode(age_without_100)

# Replace with mode
```

```r
char_preds$age[is.na(char_preds$age)] <- age_to_replace
#char_preds$age[char_preds$age == 100] <- age_to_replace
```

```r
# Categorical to numeric features
chr_categorical <- c('culture', 'house', 'title')
char_preds[, chr_categorical] <- lapply(char_preds[, chr_categorical], as.factor)
char_preds[, chr_categorical] <- sapply(char_preds[, chr_categorical], unclass)

# Replace missing with -1
char_preds[is.na(char_preds)] = -1
```

Problem 3 (2 points)

Step 1: Check for Class Bias Ideally, the proportion of both classes of the target variable should be the same. Is this so in the case of the target variable in this dataset?

Step 2: Create Training and Test Samples Perform undersampling in case of a class bias in the dataset. Create train and test splits.

Hint: To create the training sample set, sample 70% of the class with lesser rows and sample the same number from the other class. Use the remaining rows from both classes to create the test split.

Solution 3

```r
# Original distribution
table(char_preds$actual)
```

```
##
##    0    1
##  495 1451
```

```r
# Create training data
input_ones <- char_preds[which(char_preds$actual == 1), ]
input_zeros <- char_preds[which(char_preds$actual == 0), ]  # all 0's
set.seed(100)

# Sample from all alive characters
input_ones_training_rows <- sample(1:nrow(input_ones), 0.7*nrow(input_zeros))
# Sample from all dead characters
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.7*nrow(input_zeros))

training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]

# Row bind both class dataframes
trainingData <- rbind(training_ones, training_zeros)
# Shuffle rows
trainingData <- trainingData[sample(1:nrow(trainingData)), ]

# Create testing data
test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]

# Row bind both class dataframes
testData <- rbind(test_ones, test_zeros)
```

```r
# Shuffle rows
testData <- testData[sample(1:nrow(testData)), ]

# Distribution of classes in the splits
table(trainingData$actual)
```

```
##
##   0   1
## 346 346
```

```r
table(testData$actual)
```

```
##
##    0    1
##  149 1105
```

Problem 4 (3 points)

Step 1: Build the Logisitic Regression Model Train a logistic regression model to predict whether a character is dead by Book 5 (feature: `actual`) using the training split. Use the `summary` function to print the beta coefficients, p values and other statistics. Predict characters' deaths on the test split available.

Step 2: Decide on the Optimal Prediction Probability Cutoff

The default cutoff prediction probability score is 0.5 or the ratio of 1's and 0's in the training data. But sometimes, tuning the probability cutoff can improve the accuracy in both the training and test samples. Compute the optimal cutoff score (using the test split) that minimizes the misclassification error for the trained model.

Hint: Use a function from the InformationValue library to perform this task.

Solution 4

```r
# Build the Logistic Regression model
logitmod <- glm(actual ~ age + culture + title + house +
                                male + book1 + book2 + book3 + book4 + book5 +
                                isMarried + isNoble + numDeadRelations +
                                boolDeadRelations + isPopular + popularity,
                                family = binomial(link="logit"), data=trainingData)

summary(logitmod)
```

```
##
## Call:
## glm(formula = actual ~ age + culture + title + house + male +
##     book1 + book2 + book3 + book4 + book5 + isMarried + isNoble +
##     numDeadRelations + boolDeadRelations + isPopular + popularity,
##     family = binomial(link = "logit"), data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1925  -0.9087   0.1398   0.8684   2.2765
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.8109320  0.2600005   3.119  0.00181 **
```

```
## age                -0.0204693  0.0064996  -3.149  0.00164 **
## culture            -0.0167707  0.0078912  -2.125  0.03357 *
## title               0.0015170  0.0017361   0.874  0.38224
## house              -0.0014205  0.0007648  -1.857  0.06325 .
## male               -0.6131787  0.1983632  -3.091  0.00199 **
## book1              -0.4565891  0.2411541  -1.893  0.05831 .
## book2              -0.3593306  0.2186999  -1.643  0.10038
## book3              -0.3078403  0.2210841  -1.392  0.16380
## book4               1.8701684  0.2137800   8.748  < 2e-16 ***
## book5               0.1751177  0.2005916   0.873  0.38266
## isMarried           0.2275204  0.2734752   0.832  0.40543
## isNoble            -0.4582075  0.3481133  -1.316  0.18809
## numDeadRelations   -0.2100341  0.1323135  -1.587  0.11242
## boolDeadRelations   0.2505786  0.5897263   0.425  0.67090
## isPopular           0.5916993  0.6759320   0.875  0.38137
## popularity         -1.7844636  1.3167000  -1.355  0.17534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 959.32  on 691  degrees of freedom
## Residual deviance: 767.98  on 675  degrees of freedom
## AIC: 801.98
##
## Number of Fisher Scoring iterations: 5
```

```
predicted <- plogis(predict(logitmod, testData))  # predicted scores
```

```
library(InformationValue)
optCutOff <- optimalCutoff(testData$actual, predicted)[1]
optCutOff
```

```
## [1] 0.1409422
```

Problem 5 (2 points)

Using the optimal cutoff probability, compute and print the following using the predictions on the test set:

1. Misclassification error
2. Confusion Matrix
3. Sensitivity
4. Specificity

Plot the ROC Curve (Receiver Operating Characteristics Curve). What is the area under the curve?

Hint: Use predefined functions for this problem.

Solution 5

```
misClassError(testData$actual, predicted, threshold = optCutOff)
```

```
## [1] 0.1124
```

```
sensitivity(testData$actual, predicted, threshold = optCutOff)
```

```
## [1] 0.9846154
```

```
specificity(testData$actual, predicted, threshold = optCutOff)
```

```
## [1] 0.1677852
```

```
# The columns are actuals, while rows are predicteds.
confusionMatrix(testData$actual, predicted, threshold = optCutOff)
```

```
##      0    1
## 0   25   17
## 1  124 1088
```

```
plotROC(testData$actual, predicted)
```