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

Name : Adarsh Subhas Nayak SRN : PES1UG20CS620 Roll No : 54 Section : K

Prerequisites

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

```
library(tidyverse)
#install.packages("InformationValue")
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
```

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.

```
sprintf("The no. of characters listed in the data = %d", nrow(char_preds))
```

```
## [1] "The no. of characters listed in the data = 1946"
```

```
char_preds[char_preds == "" | char_preds == " "] <- NA
df <- data.frame(colSums(is.na(char_preds)) / nrow(char_preds) * 100)
colnames(df) <- c('% Missing')
df$Columns <- rownames(df)
rownames(df) <- NULL


new_df <- df[order(df$'% Missing', decreasing = TRUE),]
rownames(new_df) <- NULL
new_df
```

```
##     % 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.

```
remove <- new_df[new_df$'% Missing' > 80,'Columns']
char_preds <- char_preds[!names(char_preds) %in% remove]
ggplot(char_preds, aes(x=age)) + geom_bar()
```

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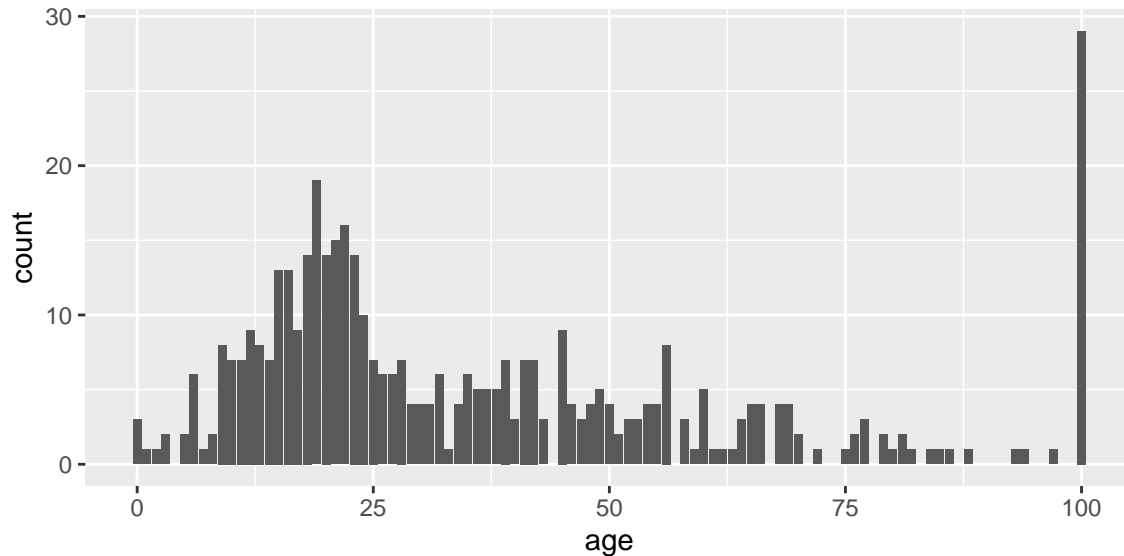


```
get_mode <- function(x) {
  mode0 <- names(which.max(table(x)))
  if(is.numeric(x)) return(as.numeric(mode0))
  mode0
}
char_categorical <- c('culture')
char_preds[, char_categorical] <- lapply(char_preds[, char_categorical], as.factor)
```

```
## Warning in '[<-.data.frame'('*tmp*', , char_categorical, value =
## list(structure(NA_integer_, levels = character(0), class = "factor"), : provided
## 1946 variables to replace 1 variables
```

```
char_preds[, char_categorical] <- sapply(char_preds[, char_categorical], unclass)
char_preds[is.na(char_preds)] = -1
char_preds[is.na(char_preds$age)] <- mean(char_preds$age, na.rm = TRUE)
char_preds[is.na(char_preds$dateOfBirth)] <- get_mode(char_preds$dateOfBirth)
```

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?

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.

```
table(char_preds$actual)
```

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

```
input_ones <- char_preds[which(char_preds$actual == 1), ]
input_zeros <- char_preds[which(char_preds$actual == 0), ]  # all 0's
set.seed(100)
input_ones_training_rows <- sample(1:nrow(input_ones), 0.7*nrow(input_zeros))
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, ]
trainingData <- rbind(training_ones, training_zeros)
trainingData <- trainingData[sample(1:nrow(trainingData)), ]


test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]
testData <- rbind(test_ones, test_zeros)
testData <- testData[sample(1:nrow(testData)), ]


table(trainingData$actual)
```

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

```
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.

```
logitmod <- glm(actual ~ age + culture + male + book1 +
                    isMarried + boolDeadRelations + isPopular + popularity,
                family = binomial(link="logit"), data=trainingData)



summary(logitmod)
```

```
##
## Call:
## glm(formula = actual ~ age + culture + male + book1 + isMarried +
##     boolDeadRelations + isPopular + popularity, family = binomial(link = "logit"),
##     data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6215  -1.2123   0.2037   1.1073   2.4596
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.763448   0.159773   4.778 1.77e-06 ***
## age               -0.016552   0.004099  -4.038 5.39e-05 ***
## culture                  NA         NA      NA       NA
## male              -0.605964   0.174437  -3.474 0.000513 ***
## book1             -0.535203   0.197888  -2.705 0.006839 **
## isMarried          0.284827   0.241133   1.181 0.237522
## boolDeadRelations -0.601724   0.360337  -1.670 0.094941 .
## isPopular         -0.071665   0.635256  -0.113 0.910179
## popularity        -0.672859   1.125599  -0.598 0.549988
## ---
## 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: 893.99  on 684  degrees of freedom
## AIC: 909.99
##
## Number of Fisher Scoring iterations: 4
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## [1] 0.101461
```

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

```
## [1] 0.118
```

```
sensitivity(testData$actual, predicted, threshold = optCutOff)
```
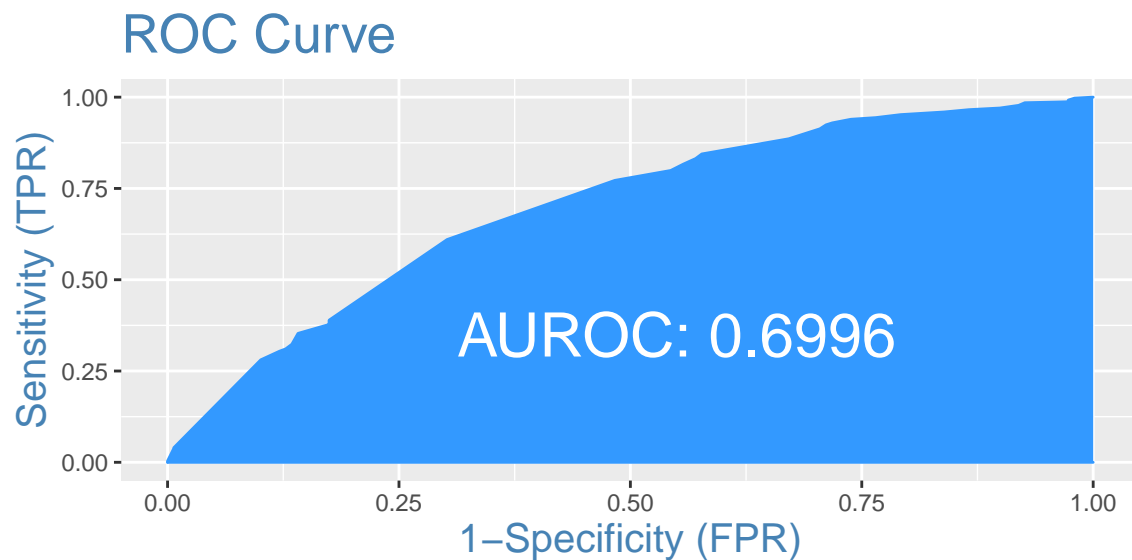
```
## [1] 0.9972851
```

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

```
## [1] 0.02684564
```

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

```
##      0    1
## 0    4    3
## 1  145 1102
```

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


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.

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

```
## [1] 0.2472
```

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

```
## [1] 0.7900452
```

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

```
## [1] 0.4765101
```

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

```
##     0   1
## 0 71 232
## 1 78 873
```

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