Does the attacking house win the battle considering who and how it attacks?

Aakash Mayur Kumar Shah - S3636808 & Simarpreet Luthra - S3706588

14 October 2018

Table of contents

Introduction

The Game of Thrones is a fatasy drama series that is based on the popular book series written by R.R.Martin. The dataset in this study was sourced from Kaggle: (https://www.kaggle.com/mylesoneill/game-of-thrones/home) and contains the collection of all the battles waged throughout the series.

The aim of this project is to identify if the battle outcome could be predicted using factors such as attacking house, defending house, season, region, battle type and so on. Thus, we applied logistic regression model on the attacker_outcome variable using the predictors that were deemed most significant by genetic algorithm.

Import useful libraries and input

```r
knitr::opts_chunk$set(echo = TRUE)

library(glmulti) # To run genetic algorithm for model selection
library(car) # For Anova
library(caret) # For Cross Validation
library(tidyverse) # To clean the data

battle <- read.csv("battle.csv") # read the data cleaned in project phase 1
```

```r
# preparing the response v ariable for analysis
battle$attacker_outcome <- factor(battle$attacker_outcome,levels =
c("loss","win"),labels = c(FALSE,TRUE))
battle$attacker_outcome <- as.logical(battle$attacker_outcome)
battle$attacker_outcome <- as.numeric(battle$attacker_outcome)

head(battle) # A look at the dataset
```

```
##   attacker_1 defender_1 attacker_outcome    battle_type major_capture
## 1  Lannister      Tully                1 pitched battle             0
## 2  Lannister  Baratheon                1         ambush             0
## 3  Lannister      Tully                1 pitched battle             1
## 4      Stark  Lannister                0 pitched battle             1
## 5      Stark  Lannister                1         ambush             1
## 6      Stark  Lannister                1         ambush             0
##   major_death summer attacker_collab defender_collab
## 1           1      1               0               0
## 2           1      1               0               0
## 3           0      1               0               0
## 4           1      1               0               0
## 5           1      1               1               0
## 6           0      1               1               0
```

Model selection

We ran the genetic algorithm six times by changing the criteria and considering just the main effects and pairwise interactions. This was done to predict the best model possible, since the dataset is limited to just thirty six observations, that is the number of major battles fought in the five books of the Game of Thrones. Thus for each run of the algorithm, we got different models which were compiled in the finalmods dataframe for comparison. Even though the model "attacker_outcome ~ 1 + major_death" seems like a better model, it seems underfitting as we just assume if there's a major character dying, the attackers win. Since, there is no practical way of knowing, whether any major character will die in battle or not. Thus, this model doesn't have any practical significance. Thus, we go with the second best model that has the lowest aic: "attacker_outcome ~ 1 + defender_1 + battle_type + defender_collab". Thus, this model states that the outcome of the battle is a function of who the defending house is, what kind of battle is being waged and how many collaborators does a defender have, all of which can be known before the battle, which makes this model more significant, statistically and practically.

Selected model

The selected model has a considerably low residual deviance and a notably low AIC as compared to the null deviance, hence we can conclude that the model has good predictability. Further, by performing the likelihood ratio test, we come to a conclusion that all the selected predictors are quite signifact statistically.

```r
finalmods # Info on the models shortlised
```

```
##
model
## 1
attacker_outcome ~ 1 + defender_1 + battle_type + defender_collab
## 2
attacker_outcome ~ 1 + battle_type + major_death + defender_collab
## 3
attacker_outcome ~ 1 + major_death
## 4 attacker_outcome ~ 1 + major_capture + major_death + attacker_collab +
defender_collab + major_death:major_capture + defender_collab:major_death
## 5
attacker_outcome ~ 1 + summer + defender_collab
## 6
attacker_outcome ~ 1 + major_death + defender_collab +
defender_collab:major_death
##        aic     aicc      bic residual.deviance degrees.of.freedom criteria
## 1 23.81909 32.61909 39.65427          3.819085                 26      aic
## 2 24.75844 27.65499 34.25955         12.758439                 30     aicc
## 3 28.27515 28.63879 31.44219         24.275152                 34      bic
## 4 24.75464 27.65119 34.25575         12.754635                 30      aic
## 5 28.32480 29.07480 33.07535         22.324797                 33     aicc
## 6 27.72291 28.47291 32.47347         21.722910                 33      bic
##                   predictors
## 1               main effects
## 2               main effects
## 3               main effects
## 4 main effects + interactions
## 5 main effects + interactions
## 6 main effects + interactions
```

```r
paste("Selected model string is ", mod1) # Final winner model
```

```
## [1] "Selected model string is  attacker_outcome ~ 1 + defender_1 +
battle_type + defender_collab"
```

```r
# Summary of the model
summary(mod.fit1)
```

```
##
## Call:
## glm(formula = mod1, family = binomial(link = "logit"), data = battle)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.48230   0.00000   0.00001   0.00002   0.90052
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 24.841  47996.935   0.001    1.000
## defender_1Greyjoy           42.185  61947.143   0.001    0.999
## defender_1Lannister         -2.249  43762.699   0.000    1.000
```

```
## defender_1Others            42.185  54878.611   0.001    0.999
## defender_1Stark             40.219  51465.216   0.001    0.999
## defender_1Tully             19.960  46753.353   0.000    1.000
## battle_typepitched battle  -44.107  27173.799  -0.002    0.999
## battle_typerazing          -43.459  68711.774  -0.001    0.999
## battle_typesiege            -1.557  30674.369   0.000    1.000
## defender_collab            -89.034  88898.504  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29.0118  on 35  degrees of freedom
## Residual deviance:  3.8191  on 26  degrees of freedom
## AIC: 23.819
##
## Number of Fisher Scoring iterations: 22
```

```
Anova(mod.fit1)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: attacker_outcome
##                 LR Chisq Df Pr(>Chisq)
## defender_1        12.229  5   0.031779 *
## battle_type       12.137  3   0.006929 **
## defender_collab   10.535  1   0.001171 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
paste0("Thus the final model is logit(attacker_outcome) = ",
mod.fit1$coefficients[1]," + (", mod.fit1$coefficients[2],") *
defender_1Greyjoy + (", mod.fit1$coefficients[3], ") * defender_1Lannister +
(", mod.fit1$coefficients[4], ") * defender_1Others +
(",mod.fit1$coefficients[5], ") * defender_1Stark +
(",mod.fit1$coefficients[6], ") * defender_1Tully +
(",mod.fit1$coefficients[7], ") * battle_typepitched battle +
(",mod.fit1$coefficients[8], ") * battle_typerazing +
(",mod.fit1$coefficients[9], ") * battle_typesiege +
(",mod.fit1$coefficients[10], ") * defender_collab")
```

```
## [1] "Thus the final model is logit(attacker_outcome) = 24.8405890189167 +
(42.1846352251284) * defender_1Greyjoy + (-2.24864877246593) *
defender_1Lannister + (42.184635225451) * defender_1Others +
(40.2192840587923) * defender_1Stark + (19.959641621419) * defender_1Tully +
(-44.1070834597757) * battle_typepitched battle + (-43.4591553557399) *
battle_typerazing + (-1.55728023589943) * battle_typesiege + (-
89.0340125319182) * defender_collab"
```

Cross Validation

The model was re-trained on a 75 percent subset of data and tested on the remaining 25 percent unseen data. The predictions of the model were found out to be really close to the outcomes in the battle which supports the predictive power of the model.

```
#Train model on 75% data and test on the rest 25%
Train = createDataPartition(battle$attacker_outcome,p=0.75,list = FALSE)
training <- battle[Train,]
testing <- battle[-Train,]
mod.fit.train <- glm(mod1,data = training,family = binomial(link="logit"))
p1 <- round(predict(mod.fit.train,newdata = testing,type = "response"),3)
p <- as.data.frame(p1)
test <- as.data.frame((testing$attacker_outcome))
y <- cbind(test,p)
colnames(y) <- c("Outcome","Prediction")
y # Comparison of actual and predicted values
```

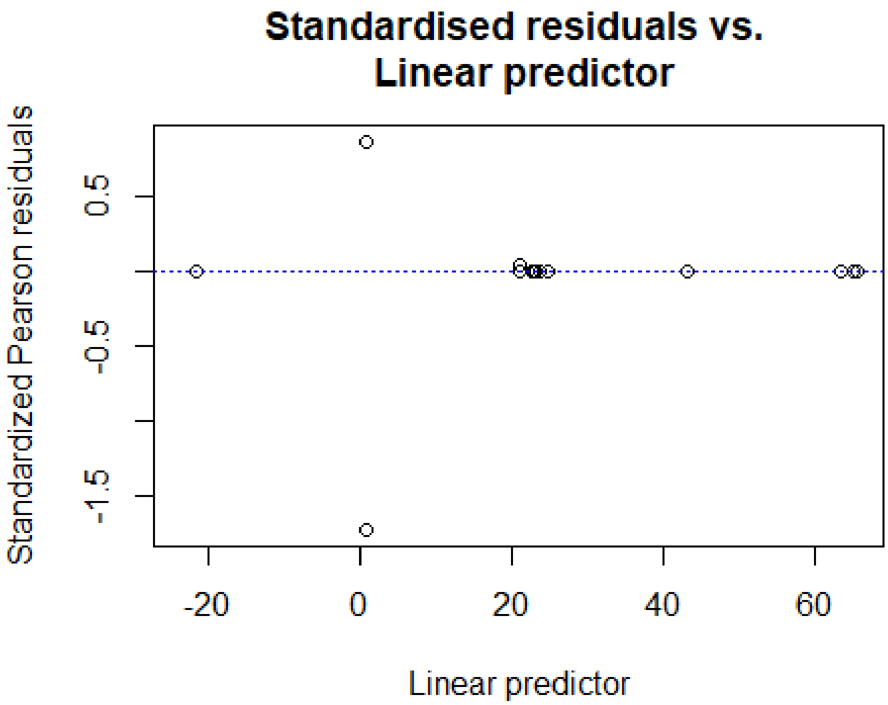
```
##    Outcome Prediction
## 1        1        0.5
## 2        1        1.0
## 4        0        1.0
## 10       1        1.0
## 19       1        1.0
## 28       1        1.0
## 32       1        1.0
## 35       1        1.0
## 36       1        1.0
```

Residual plots

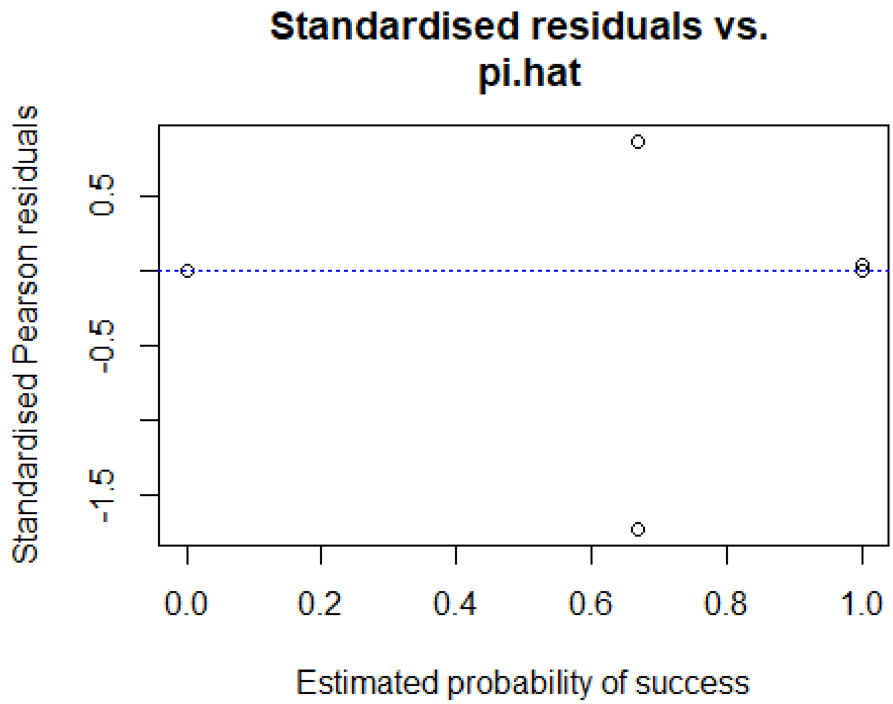
The plots of the standardised residual against the linear predictor and the predicted values seem to be evenly distributed, thus in being accordance to the assumptions of regression.

```
 # Predicted values
pi.hat <- round(predict(mod.fit1, newdata = battle, type = "response"),3)

 # Regular Pearson residuals
p.res <- round(residuals(mod.fit1, type = "pearson"),3)

 # Standardised Pearson residuals
s.res <- round(rstandard(mod.fit1, type = "pearson"),3)

# Link value prediction
lin.pred <- round(predict(mod.fit1, type = "link"),3)

battle <- data.frame(battle, pi.hat, p.res, s.res, lin.pred)

# Standardised residuals vs. Linear predictor
plot(x = battle$lin.pred, y = battle$s.res, xlab = "Linear predictor", ylab =
```

```
"Standardized Pearson residuals",
     main = "Standardised residuals vs. \n Linear predictor")
abline(h = c(3, 2, 0, -2, -3), lty = "dotted", col = "blue")
```




```
smooth.stand <- loess(formula = s.res ~ lin.pred, data = battle)

# Standardised residuals vs. Predicted values
plot(x = battle$pi.hat, y = battle$s.res, xlab = "Estimated probability of
success", ylab = "Standardised Pearson residuals",
     main = "Standardised residuals vs. \n pi.hat")
abline(h = c(3, 2, 0, -2, -3), lty = "dotted", col = "blue")
```




```
smooth.stand <- loess(formula = s.res ~ pi.hat, data = battle)
```

Goodness of Fit

To check the goodness of fit we take the ratio of residual deviance and the degrees of freedom for the model, which is way less than the problem value, thus further supporting the predictive power of the model.

```
rdev <- mod.fit1$deviance
rdev

## [1] 3.819085

dfr <- mod.fit1$df.residual
dfr

## [1] 26

ddf <- rdev/dfr
ddf

## [1] 0.1468879

thresh2 <- 1 + 2*sqrt(2/dfr)
thresh3 <- 1 + 3*sqrt(2/dfr)
round(c(rdev, dfr, ddf, thresh2, thresh3),3)

## [1]  3.819 26.000  0.147  1.555  1.832
```

Conclusions

From the logistic regression analysis, we come to a conclusion that the outcome of the Game of Thrones battles can be predicted by knowing the defending house, whether they have a collaborator or not and the battle type for the battles so far given by the equation:

"logit(attacker_outcome) = 24.8405890189167 + (42.1846352251284) * defender_1Greyjoy + (-2.24864877246593) * defender_1Lannister + (42.184635225451) * defender_1Others + (40.2192840587923) * defender_1Stark + (19.959641621419) * defender_1Tully + (-44.1070834597757) * battle_typepitched battle + (-43.4591553557399) * battle_typerazing + (-1.55728023589943) * battle_typesiege + (-89.0340125319182) * defender_collab".

With the new book coming out early next year, we eagerly await to test whether our model still holds for the battles to come in Game of Thrones!