DATA MINING APPLICATIONS  
  
 **ALY6040, WINTER 2020  
MID TERM QUIZ**

SUBMITTED BY: Keyur Shah, Jayeta Biswas, Shraddha Gopalakrishnan

NUID: 001089242, 001068448, 001376444

SUBMITTED TO: Prof. Justin Grosz

DATE: 03/11/2020

**Part 1**

1. Who has led the most attacks?

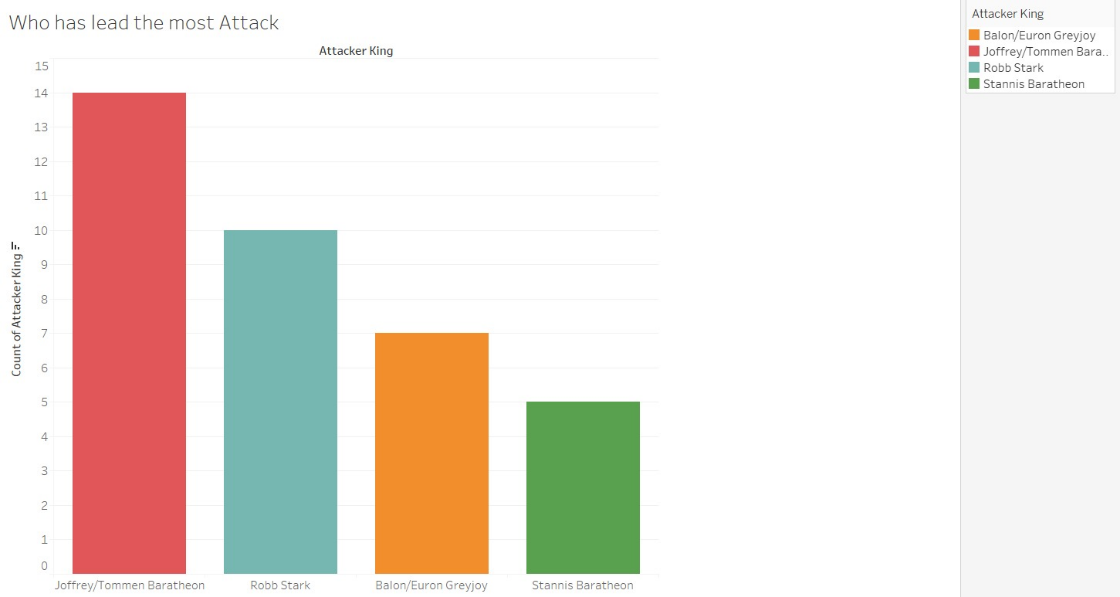
The attacker kings **Joffrey** and **Tommen Baratheon** have led the most attacks as seen from the graph. The count of attacks both the kings led is seen as 14.

Figure 1: Lead Most Attacks

The graph is shown using Attacker Kings and count of Attacker Kings from the csv file.

1. Who has been attacked the most?

**King Robb Stark** has been attacked the most whose count is also 14.

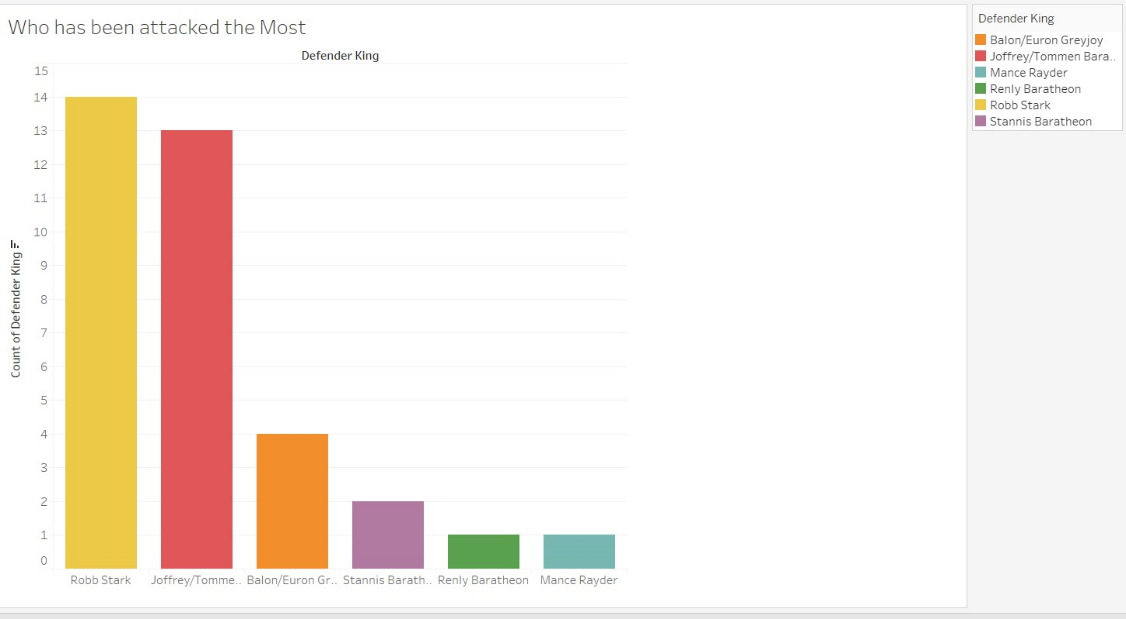
The data is plotted using Defender Kings and Count of Defender King columns of the dataset.

Figure 2: Attacked the most

1. What is the region with second most battle?

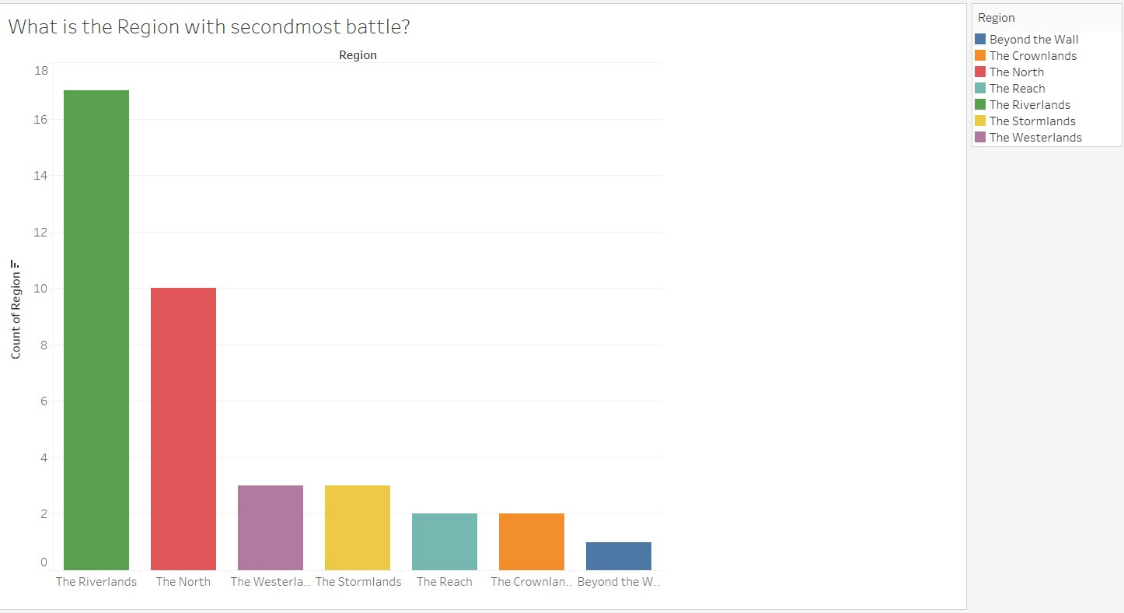
**The North** is the region with second most battle.

Figure 3: Second most battle Region

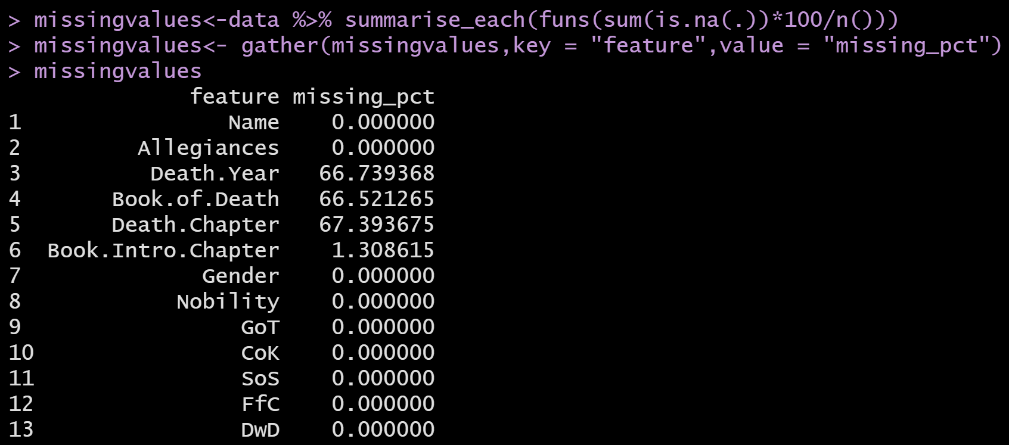
The number of battles is 10 in this specific region.

1. How will you clean the NULL variables from the dataset?

The percentage of Null values for attributes **attacker size** & **defender size** is **36.84%** and **50%**. Since the range of NA values is above 30%, we will have to find a solution instead of directly ignoring the figures. The elucidation for the same is considering the battle types and the reviewing the attacker and defender army sizes to come up with a statistical quick fix. Then we will check for which attacker Kings, defender Kings and battle types the poll of attacker or defender is more. Later we decide what values to fill in the cells containing NULL. It is easier to find the proportions for battle types. If either of attacker/defender sizes are missing, the cell will be filled with the mean of that particular battle’s attacker/defender size (whichever is provided i.e., NULL). For a given row, if both columns are NULL, then we can try to find this data online from one of the websites before removing the tuple.

**Part 2**

1. Clean Up the Dataset & Walk through the Method that you choose for each variable.

To perform data cleansing, we first compare our data to summarized version using **is.na** to get how many values are NA. Using **gather** function, we obtain a table of all the attributes and their corresponding percentage of missing values as follows.

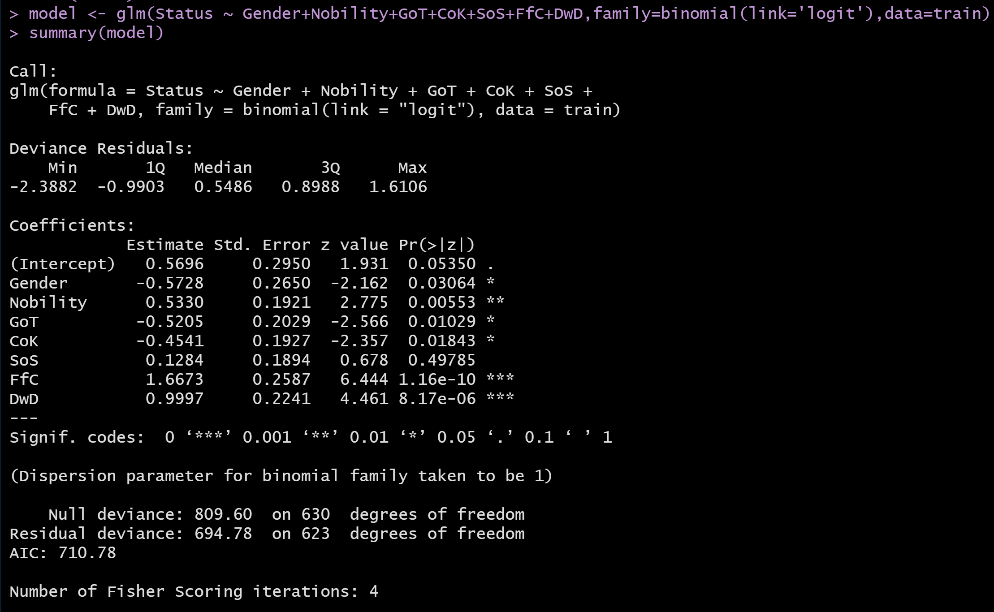
From this we are sure to consider only *Gender, Nobility, GoT, CoK, SoS, FfC,* and *DwD* as the main parameters for further analysis.

Figure 4: Missing Values Analysis

Since Death.Year, Book.of.Death, Death.Chapter show maximum missing values, we go on to convert those to 0. With the usage of for loop, a new column named “Status” is created to indicate whether a Game of Thrones character is Alive (1) or Dead (0) based on the current Death.Year abstracts. Next, the rows of Status column are checked for complete cases. This will append “Status” as the 14th attribute of character-deaths dataset.

1. Run a logistic regression model to predict which characters would live or die.

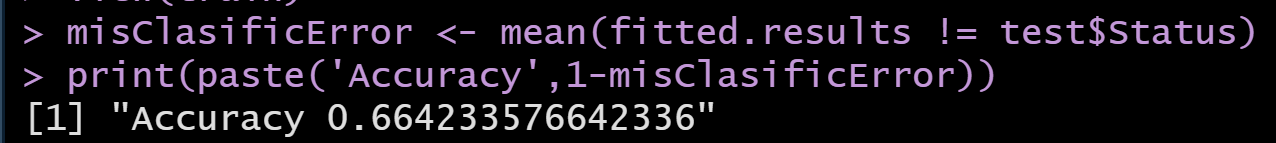
Before the regression, we finish data encapsulation for selecting random rows in the ratio 70:30 and treating these split samples as training and testing datasets. A logistic regression is run wherein Status is our dependent variable in the glm function. It is a form of binary regression since Status has binary dependent variables and our prediction revolves around Character being Alive or Dead.

1. Interpret: How accurate & what do coefficients mean?

Logistic regression analysis shows Gender, Nobility, GoT, CoK, FfC, and DwD are statistically significant and SoS is not. The Akaike Information Criterion AIC is calculated as **710.78**.

Figure 5: Logistic Regression

Further we run the prediction model on the test dataset. The accuracy of the predicted model is seen to be **66.42%**. The logic behind this is subtracting the probability of errors from 1.



The coefficients i.e., z-values are the correlation figures between dependent variable “Status” and independent variables- “Gender, Nobility, GoT, CoK, FfC, and DwD”.

4. Run another model using an optimization technique from last week (Forward, Backward, PCA), explain why you choose that method & compare this model & the one in step 2.

We chose Stepwise Algorithm for executing a different model. From the previous regression model, we know that SoS no longer holds significance in affecting the final prediction. So we do not include it in the glm function for a new logistic modeling. Now, we obtain Gender, Nobility, GoT, CoK, FfC, and DwD as the final statistically significant seers. In the summary, we see that AIC value is 709.24. Libraries “leaps” and “MASS” are called in order to execute the stepAIC model in both directions; similar to the previous assignment. Average helps us in giving a threshold for predicting the outcomes. Moving on we convert the content in the predicted model to factors using library “caret”. For the purpose of easy computations, we make use of a package named **e1071** which is for training support vector machines to help the stepwise algorithm make a better prediction and entail decisions for the binary classification.

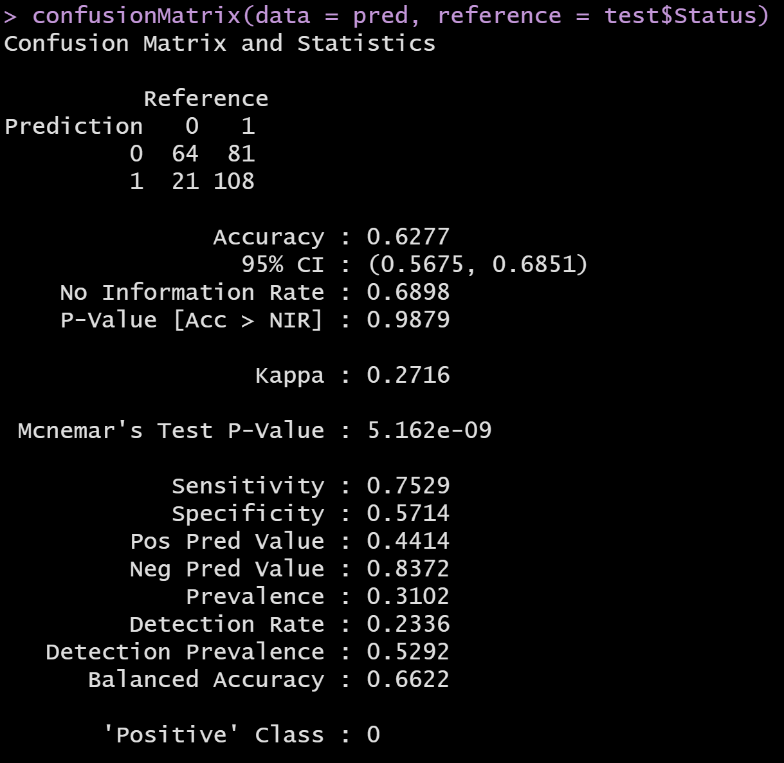
The code besides gives an interesting confusion matrix with a prediction accuracy of **62.77%**. The Stepwise AIC model has predicted the dominant class i.e., ‘Positive class’ for our dataset that means “With 95% Confidence interval, the Game of Thrones character will have the Status as Alive from ur dataset”.

Figure 6: ConfusionMatrix Stepwise

We aimed to use Stepwise because it’s a combination of both Forward and Backward selection criteria and moreover chooses the model that best fits our dataset with all the required/optimal predictor variables. From our in-depth analysis, it is clear that Logistic Regression works way better than Stepwise for this dataset because of two simple reasons:

a. Accuracy of Logistic model 66% is greater than that of Stepwise model 62%.

b. AIC value for Logistic is 709.24 and for Stepwise it’s 710.78; Lower AIC value means better fit.

**Appendix** (Code)

library(readr)

library(dplyr)

library(caret)

library(tidyverse)

# Data import and summary

data<-read.csv(file.choose())

View(data)

summary(data)

attach(data)

# Handling missing values

missingvalues<-data %>% summarise\_each(funs(sum(is.na(.))\*100/n()))

missingvalues<- gather(missingvalues,key = "feature",value = "missing\_pct")

missingvalues

data$Death.Year[is.na(data$Death.Year)] <- 0

data$Book.of.Death[is.na(data$Book.of.Death)] <- 0

data$Death.Chapter[is.na(data$Death.Chapter)] <- 0

for (i in 1:nrow(data)) {

if (data[i,3] == '0') {

data[i,14] <- 1

}

else {

data[i,14] <- 0

}

}

colnames(data)[14]<-"Status"

resultDF = data[complete.cases(data), ]

View(resultDF)

attach(resultDF)

# model <- lm(Status ~., data = data)

# summary(model)

###### Logistic Model #####

set.seed(1234)

ind <- sample(2, nrow(resultDF), replace = T, prob = c(0.7, 0.3))

train <- resultDF[ind==1,]

test <- resultDF[ind==2,]

ind

model <- glm(Status ~ Gender+Nobility+GoT+CoK+SoS+FfC+DwD,family=binomial(link='logit'),data=train)

summary(model)

model

fitted.results <- predict.glm(model,newdata=subset(test,select=c(7,8,9,10,11,12,13)),type='response')

fitted.results <- ifelse(fitted.results > 0.5,1,0)

fitted.results

view(train)

misClasificError <- mean(fitted.results != test$Status)

print(paste('Accuracy',1-misClasificError))

##### New Model using Stepwise Algorithm #####

model.null <- glm(Status ~ Gender+Nobility+GoT+CoK+FfC+DwD, family = "binomial" ,data = train)

summary(model.null)

model.null

library(leaps)

library(MASS)

stepmodel<- stepAIC(model.null, direction = "both", trace=FALSE)

summary(stepmodel)

pred <- predict.glm(model.null,newdata = test, type = "response")

mean(pred)

pred <- ifelse(pred > 0.7,1,0)

view(pred)

pred <- as.factor(pred)

levels(pred)

library(caret)

test$Status <- as.factor(test$Status)

install.packages("e1071")

library(e1071)

??e1071

confusionMatrix(data = pred, reference = test$Status)