Binary and Penalised Logistic Regression Analysis

Johnny Lo

June 2021

Table of Contents

Binary Logistic Regression Modelling

Logistic regression is a modelling approach that uses a logit function to model a **categorical** outcome or response variable, however, it is mostly applied to binary outcomes, i.e. variables with **two** categories. In binary logistic regression modelling, the two possible outcomes, such as Yes/No, Success/Failure, Dead/Alive, are labelled as 1 and 0, respectively, although the 0-1 label may be switch around depending on the category of interest.

In binary logistic regression, supposing that *Success=1* and *Failure=0*, the probability of success is given by p, i.e., $P(Y = 1) = p$, and hence, the probability of failure is $1 - p$, i.e., $P(Y = 0) = 1 - p$. The odds of success over failure is given by:

$$Odds = \frac{p}{1-p}$$

Logistic regression models log-odds as a linear function of k features $X_1, X_2, \ldots, X_k$. That is,

$$ln(Odds) = logit(p) = ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + + \beta_k X_k$$

Once the above logistic (or logit) regression model is fully defined, it is then used to estimate log-odds, which can then be used to compute the probability p of *Success* via the following equation.

$$p = \frac{e^{ln(Odds)}}{1 + e^{ln(Odds)}} = \frac{Odds}{1 + Odds}$$

For any given untested sample, if the estimated probability $\hat{p} > 0.5$, the sample is classified a *Success*. The sample is classified as a *Failure* if the estimated probability $\hat{p} < 0.5$.

To start, you need to install and load the relevant packages for this workshop.

```
#De-comment to install the packages below
#install.packages(c("tidyverse","timeDate","caret","glmnet","e1071"))
library(tidyverse)  #For ggplot2 and dplyr
library(caret)  #Classification and Regression Training package
library(glmnet)  #For visualisng correlation matrix
```

We are going to use the *COVID19.csv* dataset to illustrate the logistic regression process in R. The dataset contains COVID-19 cases across a number of regions from 13-Jan to 28-Feb 2020. You can download the data from the original source by clicking on this link.

```
#Note that you may need to change directory path
COVID19 <- read.csv("COVID19.csv", header=TRUE,
                    stringsAsFactors=FALSE,
                    na.strings="NA");  #Read in the data
View(COVID19)  #View the dataset
```

Note the inconsistencies in data entry and the amount missing values. Data cleaning and data wrangling will be required in order to extract meaningful information from the data. We will also need to make certain assumptions about particular variables.

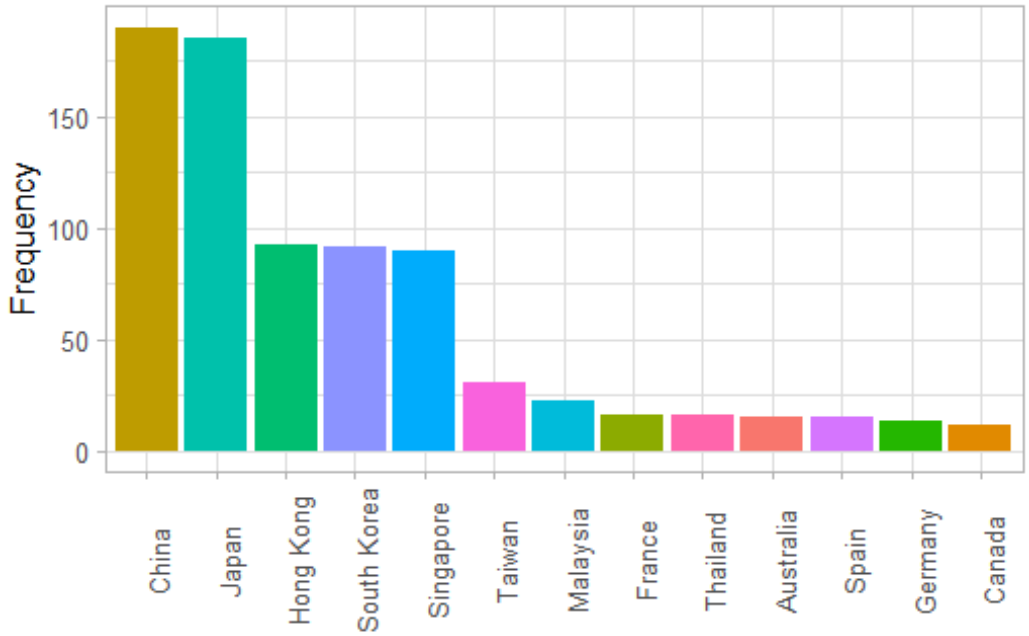
Next, we will extract the variables 1) **region**, 2) **gender**, 3) **age**, 4) **hosp_visit_dat**, and 5) **death**, from the dataset.

The outcome variable is defined by **death**, where 1 = death and 0 = no death. However, you will notice that some of the responses are given as dates. We will assume that these are dates of death and therefore, we will need to recode these as well. We will also recode (then remove) **hosp_visit_date** to either "Yes" or "No", and assign them to a new variable **hosp_visit**. Furthermore, we will only analyse the complete cases.

```
#Extract the features of interest
dat <- COVID19[,c("region","gender","age","hosp_visit_date","death")]

#Recode hosp_vist_dat and assign to new variable.
dat$hosp_visit <- ifelse(is.na(dat$hosp_visit_date),"No","Yes");
dat <- na.omit(dat[,-4]) #Remove hosp_visit_date and missing values

#Recode death to Yes and No
dat$death <- ifelse(dat$death=="0","No","Yes");

View(dat) #View the data to check recoding is okay.
```

Let us explore how many COVID-19 cases there are during this period in each region.

```
#Tabulate the number of cases in each region
df <- table(dat$region) %>% data.frame;
colnames(df) <- c("Region","Frequency");  #renaming the columns

#Only plot those with more than 10 cases
ggplot(df[df$Frequency>10,],
       #Show the frequency bars in descending order of counts
       aes(x=reorder(Region,-Frequency),y=Frequency))+
  geom_bar(stat="identity",aes(fill=Region)) +
  theme_light(base_size = 12)+
  xlab("") +
  theme(legend.position="",
        axis.text.x=element_text(angle = 90))
```



Note that the number of COVID-19 cases varies substantially from region to region. Given that rate of death is low (statistically speaking), it is not worthwhile to include regions with low instances of COVID-19. In this case, we will only select the top 4 regions with the most COVID-19 cases. Singapore (5th ranked) is not included as there is no reported cases of death during this period.

```
dat.top4 <- subset(dat,region=="China"|
                       region=="Japan"|
                       region=="South Korea"|
                       region=="Hong Kong")
#Check the structure
str(dat.top4)
```

```
## 'data.frame':    560 obs. of  5 variables:
##  $ region    : chr  "China" "China" "China" "China" ...
##  $ gender    : chr  "male" "female" "male" "female" ...
##  $ age       : num  66 56 46 60 58 44 34 37 39 56 ...
##  $ death     : chr  "No" "No" "No" "No" ...
##  $ hosp_visit: chr  "Yes" "Yes" "Yes" "Yes" ...
```

Next, we will convert all the *categorical* variables in the dataset to factors. Note that **age** is numeric.

```
#Convert the categorical features into factors, except for age (3rd column)
temp <- lapply(dat.top4[,-3],as.factor)
#Combine age to the newly converted factors
dat.top4 <- data.frame(age=dat.top4$age,temp); str(dat.top4)
```

```
## 'data.frame':    560 obs. of  5 variables:
##  $ age       : num  66 56 46 60 58 44 34 37 39 56 ...
##  $ region    : Factor w/ 4 levels "China","Hong Kong",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ gender    : Factor w/ 2 levels "female","male": 2 1 2 1 2 1 2 2 2 2 ...
##  $ death     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ hosp_visit: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 2 2 2 ...
```

Note that for each of the categorical features, i.e factors, the first level is the reference level. For instance, with the feature **region**, *China* is the **reference** region and the logistic regression model will compare all other regions to China. For **gender**, *female* is the reference level and the males will be compared to the females. The *No* response for both **hosp_visit** and **death**, is the reference level.

If required, we can use the *relevel(.)* function and the **ref** argument to change the reference level. For example, the command *relevel (dat.top4$gender, ref="male")* will change the reference level in **gender** to *male*. Changing the reference level of a categorical feature will impact the interpretation of the relationship between the feature and outcome, but it will not affect the overall model fit nor its predictive performance.

We will now split the dataset into *training* and *test* sets. The training set is used to build the logistic regression model, and the test set is used to evaluate its predictive performance. Here, we will use a 75/25 split, i.e. randomly select 75% of the original data to be in the training set, and the remaining 25% form the test set.

```
#Create the training and test datasets

set.seed(1)  #Set the random seed.

# Step 1: Get row numbers for the training data
trainRowNum <- createDataPartition(dat.top4$death, #The outcome variable
                                   #proportion of data to form the training
set
                                   p=0.75,
                                   #Don't store the result in a list
                                   list=FALSE);

# Step 2: Create the training  dataset
trainData <- dat.top4[trainRowNum,]

# Step 3: Create the test dataset
testData <- dat.top4[-trainRowNum,]
```

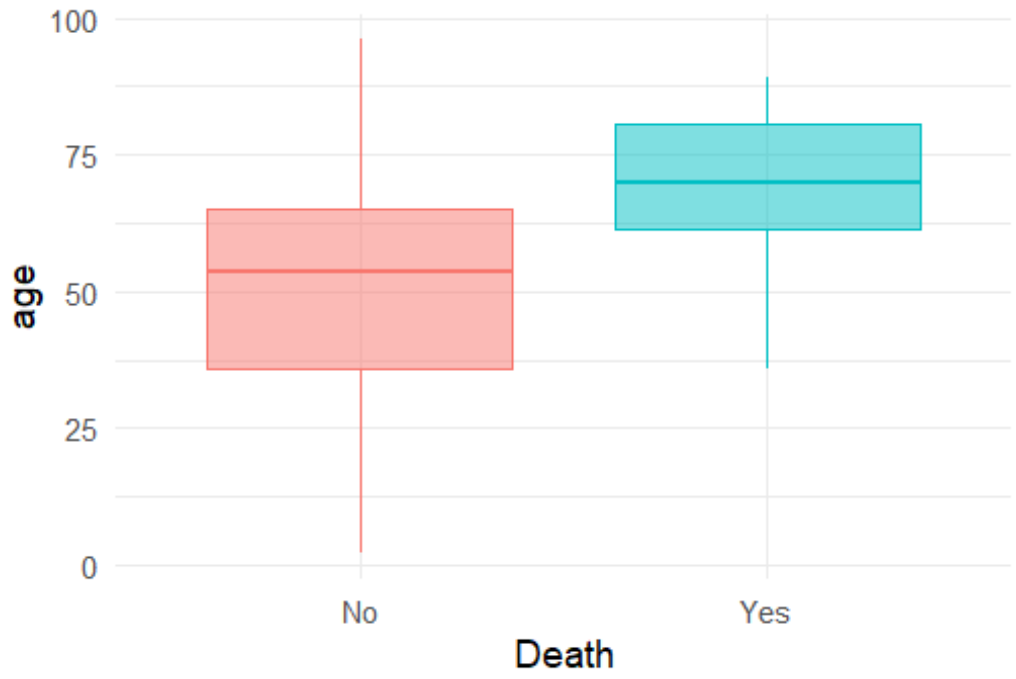
Binary logistic regression is performed in R by using the *glm(.)* function, and with the argument **family="binomial"**.

```
#Logistic regression modelling
mod.covid.lg <- glm(death~., family="binomial", data=trainData);
summary(mod.covid.lg)  #Summarise the model
```

```
##
## Call:
## glm(formula = death ~ ., family = "binomial", data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8092  -0.3058  -0.1438  -0.0509   3.5150
##
## Coefficients:
##                   Estimate Std. Error z value        Pr(>|z|)
## (Intercept)       -9.90071    1.41131  -7.015 0.0000000000023 ***
## age                0.11617    0.01873   6.203 0.0000000005542 ***
## regionHong Kong   -3.06508    0.82298  -3.724        0.000196 ***
## regionJapan       -3.37803    0.65596  -5.150 0.0000002608459 ***
## regionSouth Korea  0.66035    0.66078   0.999        0.317625
## gendermale         1.48519    0.49471   3.002        0.002681 **
## hosp_visitYes      0.77433    0.56429   1.372        0.169999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.85  on 420  degrees of freedom
## Residual deviance: 157.30  on 414  degrees of freedom
## AIC: 171.3
##
## Number of Fisher Scoring iterations: 7
```

Exercise: What can you conclude from the above summary of the logistic regression model? Is age a factor? How do Japan, South Korea and Hong Kong compare to China in relation to COVID-19 deaths? Are the rate of death higher for males as compared to females? Do hospital visits have any impact on COVID-19 deaths?

```
ggplot(dat.top4,aes(x=death,y=age,colour=death,fill=death)) +
  geom_boxplot(alpha=0.5,show.legend=FALSE) +
  theme_minimal(base_size=14) +
  labs(x="Death","Age in Years")
```



Exercise: What can you observe from the above plot ?

We can also cross-tabulate the results, say between **region** and **death**, to assist us with the interpretation.

```
freq <- table(trainData$region,trainData$death); freq
```

```
##
##                No Yes
##   China       113  27
##   Hong Kong    70   2
##   Japan       139   4
##   South Korea  58   8
```

```
prop <- freq %>% prop.table(1); prop %>% round(digit=3)
```

```
##
##                   No   Yes
##   China       0.807 0.193
##   Hong Kong   0.972 0.028
##   Japan       0.972 0.028
##   South Korea 0.879 0.121
```

We can see that during this period, the rate of death for China (19%) and South Korea (12%) are higher than both Hong Kong and Japan, where the death rate was just under 3%.

Exercise: Cross-tabulate **gender** and **hosp_visit** against **death** and interpret.

We will now evaluate the predictive power of the logistic regression model on the **test set**. Here, we will use the *confusionMatrix(.)* function from the **caret** package to summarise the confusion matrix. However, note that when using this function, we must have *Yes* as the first level (opposite of what *glm(.)* requires) and *No* as the second level. Given this, we will need to reverse the order of the *Yes* and *No* levels for the predicted and actual cases of deaths.

```
#predicted probability of death on the test data
pred.prob <- predict(mod.covid.lg,new=testData,type="response")
pred.class <- ifelse(pred.prob>0.5,"Yes","No") #Assign to the respective
class

#Confusion matrix with re-ordering of "Yes" and "No" responses
cf.lg <- table(pred.class %>% as.factor %>% relevel(ref="Yes"),
               testData$death %>% relevel(ref="Yes"));

prop <- prop.table(cf.lg,2); prop %>% round(digit=3) #Proportions by columns
```

```
##
##         Yes    No
##   Yes 0.462 0.024
##   No  0.538 0.976
```

```
#Summary of confusion matrix
confusionMatrix(cf.lg);
```

```
## Confusion Matrix and Statistics
##
##
##       Yes  No
##   Yes   6   3
##   No    7 123
##
##                Accuracy : 0.9281
##                  95% CI : (0.8717, 0.965)
##     No Information Rate : 0.9065
##     P-Value [Acc > NIR] : 0.2390
##
##                   Kappa : 0.5078
##
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.46154
##             Specificity : 0.97619
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.94615
##              Prevalence : 0.09353
##          Detection Rate : 0.04317
##    Detection Prevalence : 0.06475
##       Balanced Accuracy : 0.71886
##
##        'Positive' Class : Yes
##
```

Here we note that the accuracy of the logistic regression model is 0.928; however, this is largely due to its high specificity (0.976). The sensitivity is low at 0.462, which means the model performs poorly in predicting cases of death.

Penalised Logistic Regression Modelling

To perform penalised regression modelling, we will use the *glmnet(.)* function, concurrently with the *train(.)* function from the **caret** package. We will need to define two parameters for the *train(.)* function.

1) **alpha**: elastic-net mixing parameter
   i) 0 = Ridge regression
   ii) 1 = LASSO regression
   iii) A value between 0 and 1 for elastic-net regression
2) **lambda**: a sequence of lambda values to test with

The *train(.)* function can automatically choose the optimal α (for elastic net) and λ values using cross-validation (CV) for us, or we can define a search range for each of these two parameters.

Logistic Ridge Regression

As we are running ridge regression in the first instance, alpha will be set to 0 here. Furthermore, we will create a list of 100 λ values between 0.001 and 1000 to search through. The search range is selected to ensure that it is large enough to capture the optimal value. However, you can always narrow down the search range after the initial search to refine the optimal value.

```
lambdas <- 10^seq(-3,3,length=100) #A sequence 100 lambda values

set.seed(1)
mod.covid.ridge <- train(death ~., #Formula
                   data = trainData, #Training data
                   method = "glmnet",  #Penalised regression modelling
                   #Set to c("center", "scale") to standardise data
                   preProcess = NULL,
                   #Perform 10-fold CV, 5 times over.
                   trControl = trainControl("repeatedcv",
                                            number = 10,
                                            repeats = 5),
                   tuneGrid = expand.grid(alpha = 0, #Ridge regression
                                          lambda = lambdas)
                  )

#Optimal lambda value
mod.covid.ridge$bestTune

##    alpha     lambda
## 18     0 0.01072267
```

The optimal λ is 0.01. The regression coefficients of the optimal logistic ridge regression model are as follows:

```
# Model coefficients
coef(mod.covid.ridge$finalModel, mod.covid.ridge$bestTune$lambda)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)       -7.10628564
## age                0.07574143
## regionHong Kong   -1.94672906
## regionJapan       -2.16725072
## regionSouth Korea  0.48083126
## gendermale         1.06614492
## hosp_visitYes      0.57360121
```

Now that we have our optimal logistic ridge regression model, we can now evaluate its predictive performance on the **test** set. Note that the **classes** are predicted by default in this instance. If you wish to predict the **probabilities** instead, then add the argument **type="prob"** to the *predict(.)* function.

```
#predicted classes of death on the test data
pred.class.ridge <- predict(mod.covid.ridge,new=testData)

#Confusion matrix with re-ordering of "Yes" and "No" responses
cf.ridge <- table(pred.class.ridge %>% relevel(ref="Yes"),
                  testData$death %>% relevel(ref="Yes"));

prop <- prop.table(cf.ridge,2); prop %>% round(digit=3) #Proportions by
columns
```

```
##
##         Yes    No
##   Yes 0.308 0.024
##   No  0.692 0.976
```

```
#Summary of confusion matrix
confusionMatrix(cf.ridge)
```

```
## Confusion Matrix and Statistics
##
##
##       Yes  No
##   Yes   4   3
##   No    9 123
##
##                Accuracy : 0.9137
##                  95% CI : (0.8541, 0.9546)
##     No Information Rate : 0.9065
##     P-Value [Acc > NIR] : 0.4577
##
##                   Kappa : 0.358
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.30769
##             Specificity : 0.97619
##          Pos Pred Value : 0.57143
##          Neg Pred Value : 0.93182
##              Prevalence : 0.09353
##          Detection Rate : 0.02878
##    Detection Prevalence : 0.05036
##       Balanced Accuracy : 0.64194
##
##        'Positive' Class : Yes
##
```

The accuracy of the logistic ridge regression model is 0.914, and with a specificity of 0.976, and a sensitivity of 0.308. It seems the ridge regression model has performed worse than the standard logistic regression model. Their specificities are the same, but the logistic ridge regression model has a lower sensitivity, corresponding to the mis-classification of two more cases of death.

Logistic LASSO Regression

LASSO regression differs to ridge regression in that certain regression coefficient(s) will be set to **exactly zero** as result of the penalty term applied. This also means that LASSO regression can be used as a variable selection technique.

The execution of LASSO regression modelling is exactly the same as ridge regression except the argument alpha is **set to 1**.

```
set.seed(1)
mod.covid.LASSO <- train(death ~., #Formula
                    data = trainData, #Training data
                    method = "glmnet",  #Penalised regression modelling
                    #Set preProcess to c("center", "scale") to standardise
data
                    preProcess = NULL,
                    #Perform 10-fold CV, 5 times over.
                    trControl = trainControl("repeatedcv",
                                             number = 10,
                                             repeats = 5),
                    tuneGrid = expand.grid(alpha = 1, #LASSO regression
                                           lambda = lambdas)
                   )

#Optimal lambda value
mod.covid.LASSO$bestTune

##    alpha     lambda
## 18     1 0.01072267
```

The optimal λ is 0.011. The regression coefficients of the optimal LASSO model are as follows:

```
# Model coefficients
coef(mod.covid.LASSO$finalModel, mod.covid.LASSO$bestTune$lambda)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)       -6.92364206
## age                0.08095509
## regionHong Kong   -1.85046732
## regionJapan       -2.25404598
## regionSouth Korea  .
```

```
## gendermale          0.92335028
## hosp_visitYes       0.09817193
```

Here we note that the regression coefficient for the comparison between South Korea and China is set to zero, and subsequently, the corresponding parameter is removed from the model. Therefore, we have a simplified model as compared to the standard and ridge regression models. Let us find out what implication this has on the prediction accuracy as compared to the other two previous models.

```
#predicted classes of death on the test data
pred.class.LASSO <- predict(mod.covid.LASSO,new=testData)

#Confusion matrix with re-ordering of "Yes" and "No" responses
cf.LASSO <- table(pred.class.LASSO %>% relevel(ref="Yes"),
                  testData$death %>% relevel(ref="Yes"));

prop <- prop.table(cf.LASSO,2); prop %>% round(digit=3) #Proportions by
columns

##
##         Yes    No
##   Yes 0.308 0.024
##   No  0.692 0.976

#Summary of confusion matrix
confusionMatrix(cf.LASSO)

## Confusion Matrix and Statistics
##
##
##       Yes  No
##   Yes   4   3
##   No    9 123
##
##                Accuracy : 0.9137
##                  95% CI : (0.8541, 0.9546)
##     No Information Rate : 0.9065
##     P-Value [Acc > NIR] : 0.4577
##
##                   Kappa : 0.358
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.30769
##             Specificity : 0.97619
##          Pos Pred Value : 0.57143
##          Neg Pred Value : 0.93182
##              Prevalence : 0.09353
##          Detection Rate : 0.02878
##    Detection Prevalence : 0.05036
```

```
##       Balanced Accuracy : 0.64194
##
##        'Positive' Class : Yes
##
```

Although they have the same performance outcomes, the logistic LASSO regression model would be the preferred model here as compared to the ridge regression model given that it is the simpler of the two. However, the standard logistic regression approach is still the optimal model at this stage with an accuracy of 0.928.

Logistic Elastic-Net Regression

Elastic-Net is a compromise between ridge and LASSO regression. In elastic-net regression, we can define a search range for alpha (α), just as we did with lambda. Note that $0 \leq \alpha \leq 1$.

```
alphas <- seq(0.1,0.9,by=0.1); alphas  #A sequence of alpha values to test

## [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

set.seed(1)
mod.covid.elnet <- train(death ~., #Formula
                   data = trainData, #Training data
                   method = "glmnet",  #Penalised regression modelling
                   #Set preProcess to c("center", "scale") to standardise
data
                   preProcess = NULL,
                   #Perform 10-fold CV, 5 times over.
                   trControl = trainControl("repeatedcv",
                                            number = 10,
                                            repeats = 5),
                   tuneGrid = expand.grid(alpha = alphas,
                                          lambda = lambdas)
                  )

#Optimal lambda value
mod.covid.elnet$bestTune

##    alpha     lambda
## 18   0.1 0.01072267
```

The optimal α and λ values are 0.1 and 0.01, respectively. The regression coefficients of the optimal logistic elastic-net model are as follows:

```
# Model coefficients
coef(mod.covid.elnet$finalModel, mod.covid.elnet$bestTune$lambda)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)       -7.07667566
## age                0.07611556
## regionHong Kong   -1.93862965
```

```
## regionJapan       -2.17505748
## regionSouth Korea  0.42459622
## gendermale         1.05196872
## hosp_visitYes      0.52741229
```

The logistic elastic-net model has retained all input features.

```
#predicted classes of death on the test data
pred.class.elnet <- predict(mod.covid.elnet,new=testData)

#Confusion matrix with re-ordering of "Yes" and "No" responses
cf.elnet <- table(pred.class.elnet %>% relevel(ref="Yes"),
                  testData$death %>% relevel(ref="Yes"));

prop <- prop.table(cf.elnet,2); prop %>% round(digit=3) #Proportions by
columns

##
##         Yes    No
##   Yes 0.308 0.024
##   No  0.692 0.976

#Summary of confusion matrix
confusionMatrix(cf.elnet)

## Confusion Matrix and Statistics
##
##
##       Yes  No
##   Yes   4   3
##   No    9 123
##
##                Accuracy : 0.9137
##                  95% CI : (0.8541, 0.9546)
##     No Information Rate : 0.9065
##     P-Value [Acc > NIR] : 0.4577
##
##                   Kappa : 0.358
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.30769
##             Specificity : 0.97619
##          Pos Pred Value : 0.57143
##          Neg Pred Value : 0.93182
##              Prevalence : 0.09353
##          Detection Rate : 0.02878
##    Detection Prevalence : 0.05036
##       Balanced Accuracy : 0.64194
##
```

```
##        'Positive' Class : Yes
##
```

The results for the elastic-net model here, are identical to the other two penalised regression models but are still inferior to the first model. In summary, the standard logistic regression model is the optimal model based on this test set.

Final Note

There are a number of issues associated with this particular dataset, many of which were discussed in the data preparation stage at the beginning of this workshop. Another key issue here relates to the imbalance of death and non-death cases. The minority class (i.e. death) is harder to predict because there are few instances of this class, by definition. Therefore, it is more challenging for a model to learn the characteristics of samples from this class, and to differentiate those in the minority class from those in the majority class (i.e. non-death). We can see this with the low sensitivity across all the (penalised) logistic regression models.

To overcome/alleviate this issue, one needs to rely on more specialised techniques or by using alternative, and more unbiased, performance evaluation measures. This is beyond the scope of this unit, but you should keep this in mind in the future when training a model. However, we will examine an alternative performance measure in the next module in ROC analysis.