Rcode for GV900 HW4

GV900 Political Explanation Homework 4

My student ID: 2107091

17/03/2022

```
# Firstly, it will be advisable to clean the console and the environment from
previously used data and values by using the following functions:

rm(list=ls(all=TRUE))
cat("\014")
```

```
# Then, I will load some packages that I will need during the coding, to
wrangle my dataset, to create regression tables, effects, arrange grapgs, and
create ROC curves.

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(stargazer)

##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary
Statistics Tables.

##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

library(foreign)

## Warning: package 'foreign' was built under R version 4.1.2

library(effects)

## Warning: package 'effects' was built under R version 4.1.2

## Loading required package: carData

## lattice theme set by effectsTheme()
## See ?effectsTheme for details.

library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

library(ROCR)

## Warning: package 'ROCR' was built under R version 4.1.2

# Cite: Hlavac, Marek (2018). stargazer: Well-Formatted Regression and
Summary Statistics Tables.

# Now, I can import the data set in my console by using the "read.csv"
function to read the Comma Separated Values file. Moreover, I will use the
"paste0" function to concatenate all elements without a separator (source:
https://r-lang.com/paste0-function-in-r-with-example/). The path to finding
the data is stored in an object called "myPath":

tita <- read.csv(paste0(myPath, "titanic2.csv"))
```

1.A LOGIT MODEL

```
# To classify based on the "child" variable, I have first to assign the value
of 0 or 1 to it based if it is a child (1) or not (0), and I will store it in
a new column named "young". Hence:

tita$young <- ifelse(tita$child == "Child", 1, 0)

# I will also do a similar process for the "female" variable with female (1)
and male (0), and I will store it in a new column named "gender". Hence:

tita$gender <- ifelse(tita$female == "Female", 1, 0)

# To estimate a logit model, I will use the "glm" (generalised linear models)
function (source:
https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/glm).
Firstly, I will insert the formula with "survived" as DV, and "fare",
"gender" and "young" are the IDVs. After that, I used the "data" argument to
tell the programme from which dataset it has to collect the data. Finally, I
used the "family" argument to ask for the logit regression. Further, I will
store the results in an object called "logit_tita". Thus:

logit_tita <- glm(survived ~ fare + gender + young,
                  data = tita,
                  family = binomial(link = logit))

# Regarding the interpretation of the coefficients, a positive coefficient
related to the "young" variable means that children are more likely to
survive. Similarly, a positive coefficient related to the "gender" variable
means that female passengers are more likely to survive (source:
https://www.unm.edu/~schrader/biostat/bio2/Spr06/lec11.pdf)
```

1.B LOGGED LOGIT MODEL

```
# To check if the "fare" variable is skewed, I will code an histogram.
However, I will insert the code into a comment since it is not required by
this assignment. Thus:

# histo_fare <- ggplot(tita, aes(fare)) + geom_histogram(bins = 100)
# print(histo_fare)

# Since the "fare" variable is right-skewed, I will use the I() function to
log it inside the logit regression. I added 1 to the fare variable to avoid
errors and loss of observations when fare = 0. Hence:

logit_tita_log <- glm(survived ~ I(log(fare + 1)) + gender + young,
                      data = tita,
                      family = binomial(link = logit))
```

1.C STARGAZER TABLE

```
# To display the results of the previous two regressions, I will use the
"stargazer" function with the argument "type" as text. Further, I will add
some other options in the code to improve the readability of the table. Thus:

stargazer(logit_tita, logit_tita_log, type = "text",
          dep.var.labels = "Survival Probability",
          covariate.labels = c("Fare",
                               "Logged Fare",
                               "Gender",
                               "Young"))

##
## ==============================================
##                        Dependent variable:
##                     ----------------------------
##                        Survival Probability
##                          (1)           (2)
## ----------------------------------------------
## Fare                   0.009***
##                        (0.002)
##
## Logged Fare                           0.546***
##                                      (0.084)
##
## Gender                  2.362***       2.318***
##                          (0.156)       (0.156)
##
## Young                   0.675***       0.559**
##                          (0.235)       (0.234)
##
## Constant                -1.722***     -3.061***
##                          (0.119)       (0.282)
##
## ----------------------------------------------
## Observations              1,045         1,045
## Log Likelihood          -528.894       -524.620
## Akaike Inf. Crit.      1,065.788     1,057.241
## ==============================================
## Note:                 *p<0.1; **p<0.05; ***p<0.01
```

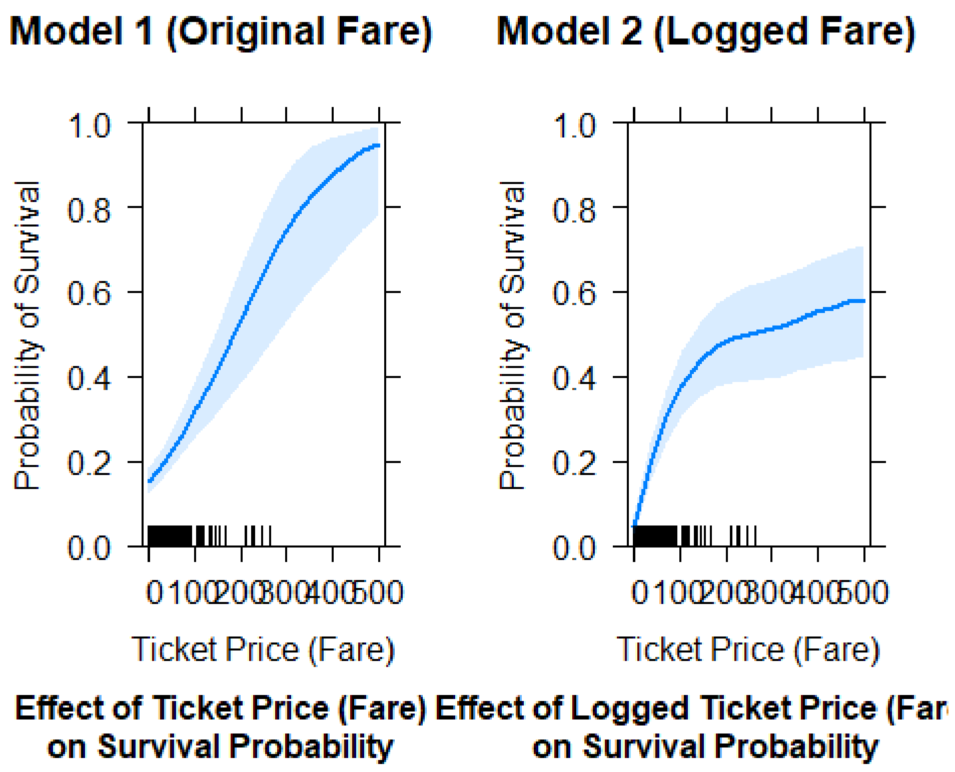
1.D MODEL PERFORMANCE

```
# The Log-Likelihood measures the goodness of fit and is better at higher
values. It associated each parameter with the probability of observing the
given sample. Further, the Akaike Information Criteria (AIC) also is a
function to assess goodness of fit, and it is better at lower values.

# The log-likelihood is higher (smaller in absolute terms) in the logged
model, and the value of the AIC (comparable since the number of observations
is the same in the models) is lower in the logged model. Therefore, Model 2
(logit_tita_log), with the logged fare variable, performs better.

# Source: https://www.statology.org/interpret-log-likelihood/ ;
https://www.statlect.com/glossary/log-likelihood ;
https://www.scribbr.com/statistics/akaike-information-criterion/
```

1.E EFFECT GRAPHS

```
# To calculate the marginal effect of fare on passenger survival, I have
firstly to store the median value of the other IDVs and use it in the coding.
Further, I will also eliminate the missing values using the "na.rm" argument.
Thus, for the gender variable:

med_gender = median(tita$gender, na.rm = TRUE)

# For the child variable:

med_young = median(tita$young, na.rm = TRUE)

# Then, I will use the effect function to store the effects of the IDVs on
```

```
the DV with age as its median. After that, I will create two graphs for each
model, one for children and one for adults. Thus, for Model 1:

eff_lr1 <- effect(term = "fare", mod = logit_tita,
                  given.values = c(gender = med_gender, young = med_young))

# For Model 2:

eff_lr2 <- effect(term = "I(log(fare + 1))", mod = logit_tita_log,
                  given.values = c(gender = med_gender, young = med_young))

# Now, I can create the graph with the marginal effect using the "plot"
function. Further, I will insert "ylim" between 0 and 1 since the survived
variable (y) can contain values of either 0 or 1. Moreover, I will insert
other arguments to increase the readability of the graphs. Hence, for Model
1:

g_lr1 <- plot(eff_lr1, type = "response",
              ylim = c(0,1),
              main = "Model 1 (Original Fare)",
              sub = "Effect of Ticket Price (Fare)\non Survival Probability",
              xlab = "Ticket Price (Fare)",
              ylab = "Probability of Survival")

# For Model 2:

g_lr2 <- plot(eff_lr2, type = "response",
              ylim = c(0,1),
              main = "Model 2 (Logged Fare)",
              sub = "Effect of Logged Ticket Price (Fare)\non Survival
Probability",
              xlab = "Ticket Price (Fare)",
              ylab = "Probability of Survival")

# Finally, I will use the "grid.arrange" function to display one graph near
the other:

grid.arrange(g_lr1, g_lr2, ncol = 2)
```



1.F GRAPH COMPARISON

```
# The graph with the original fare variable suggests that the effect of fare
on survival is linear. A one-unit increase in the fare variable has almost
the same, constant effect on the odds that the observation is a survivor when
all other variables are held constant. On the other hand, the graph with the
logged fare variable suggests that the effect of fare is non-linear. A one-
unit increase in the fare variable has decreasing marginal effects on the
odds that the observation is a survivor (1) when all other variables are held
constant. Based on the model fit statistics, we should believe the second
story to be more plausible (at elevated ticket costs, the increase in the
probability of surviving with a one-unit increase in the fare is less than
the increase in the likelihood of surviving, always with a one-unit increase
in the fare for low ticket costs).

# Source: https://towardsdatascience.com/interpreting-coefficients-in-linear-
and-logistic-regression-6ddf1295f6f1 ; https://www.statology.org/logistic-
regression-vs-linear-regression/
```

1.G LOGIT MODEL FOR ROC CURVES

```
# Now, I will compare the substantive importance of the logged fare variable
and its predictive abilities with the ROC curves. For the ROC curve, I will
first create a subset of "tita" without missing values in the fare variable,
and I will store it in an object called "tita_pure". Thus:
```

```
tita_pure <- subset(tita, tita$fare != "NA")

# Then, I will create the training set (with 80% of the original data) and
the test set (with 20% of the original). I will use the "set.seed" function
to maintain the randomly selected rows constant during the code. Then, I will
implement the "sample" function to have the training set. Finally, I will
store the training set in an object called "train_ds" while the test set
inside "test_ds". Hence:

set.seed(12345)
index <- sample(1:nrow(tita_pure), size = nrow(tita_pure)*0.80)
train_ds <- tita_pure[index, ]
test_ds <- tita_pure[-index, ]

# After that, I will recreate another logit regression, using only the
training set of variables and without the logged fare variable. Thus:

train_lr1 <- glm(survived ~ gender + young,
                 data = train_ds,
                 family = binomial(link = logit))
```

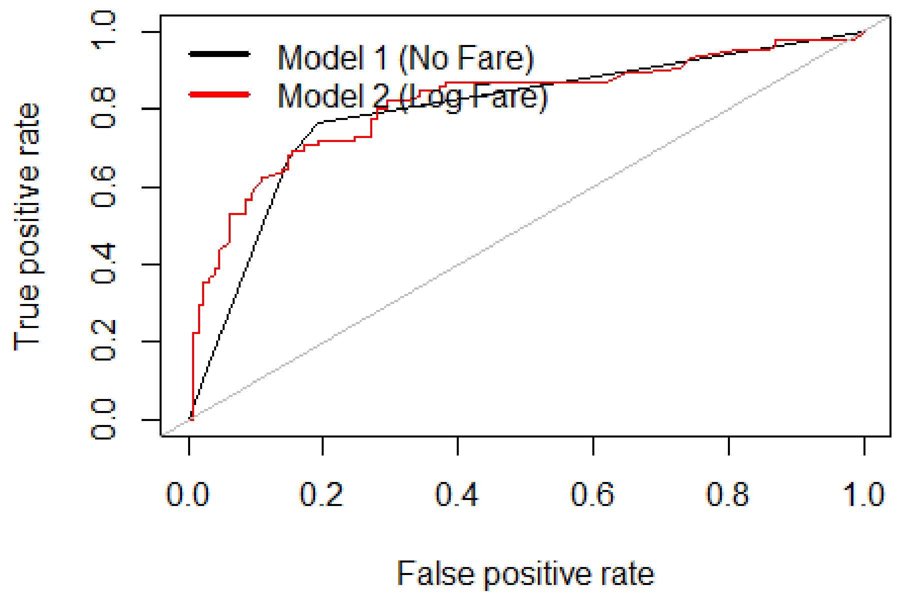
1.H STARGAZER TABLE 2

```
# First of all, I also need to create the predictions for the logged model to
have a comparison. Thus:

train_lr2 <- glm(survived ~ I(log(fare + 1)) + gender + young,
                 data = train_ds,
                 family = binomial(link = logit))

# Now, I can compare the two logit models again. Further, I am sure that they
are using the same training set since in both of them I selected as "data"
argument the "train_ds" set from which to take the values, and I inserted the
set.seed function to keep the randomly selected rows constant during the
exercise. Thus:

stargazer(train_lr1, train_lr2, type = "text",
          dep.var.labels= "Survival Probability",
          covariate.labels = c("Logged Fare",
                               "Gender",
                               "Young"))

##
## ==============================================
##                       Dependent variable:
##                   ----------------------------
##                       Survival Probability
##                         (1)           (2)
## ------------------------------------------------
## Logged Fare                           0.584***
##                                       (0.095)
##
## Gender               2.450***        2.339***
##                       (0.171)        (0.175)
##
## Young                 0.475*          0.409
##                       (0.268)        (0.264)
##
## Constant             -1.395***       -3.172***
##                       (0.113)        (0.322)
##
## ------------------------------------------------
## Observations            831             831
## Log Likelihood       -436.859        -416.666
## Akaike Inf. Crit.     879.719         841.331
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

1.I ROC GRAPHS

```
# Then, I will create the predicted probabilities for the logit models
implementing the "predict" function with argument type as a response to
obtain the P-hat value. Hence:

pred_lr1 <- predict(train_lr1, newdata = test_ds, type = "response")

pred_lr2 <- predict(train_lr2, newdata = test_ds, type = "response")

# Then, I will eliminate further NA values in the columns. Thus:

filtered1_ds <- data.frame(cbind(pred_lr1, test_ds$survived))
filtered1_ds <- na.omit(filtered1_ds)
colnames(filtered1_ds) <- c("pred_lr1", "Survived")

filtered2_ds <- data.frame(cbind(pred_lr2, test_ds$survived))
filtered2_ds <- na.omit(filtered2_ds)
colnames(filtered2_ds) <- c("pred_lr2", "Survived")

# After that, I will combine the predicted values with the actual outcomes by
using the "prediction" function. Thus:

prediction_lr1 <- prediction(filtered1_ds$pred_lr1, filtered1_ds$Survived)

prediction_lr2 <- prediction(filtered2_ds$pred_lr2, filtered2_ds$Survived)

# Finally, I will use the "performance" function to individuate the true
positive rate and the false positive rate in the two models. Thus:
```

```
performance_lr1 <- performance(prediction_lr1, "tpr", "fpr")

performance_lr2 <- performance(prediction_lr2, "tpr", "fpr")

# Now, I can plot the ROC curve using the "plot" function. I will also add
the 45 degrees line using the "abline" function. I added some options to
distinguish the colours of the curves to differentiate between them. Further,
I also added a legend to explain the graph more comprehensively. Thus:

plot(performance_lr1)
plot(performance_lr2, add = TRUE, col = "red")
abline(a = 0, b = 1, col="gray")
legend("topleft",
       c("Model 1 (No Fare)","Model 2 (Log Fare)"),
       col = c("black","red"),
       cex=1, lwd=3, bty="n")
```



```
# Source: https://www.datatechnotes.com/2019/03/how-to-create-roc-curve-in-
r.html ; https://www.journaldev.com/47626/plot-roc-curve-r-programming ;
https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/
```

1.J AUC SCORES

```
# To calculate and report the Area Under the ROC Curve (AUC), I will use the
"performance" function, implementing the argument "auc". Further, I
```

```
implemented the "@y.values" argument to contain the y values of the curve of
this particular cross-validation run (source:
https://www.rdocumentation.org/packages/ROCR/versions/1.0-
11/topics/performance-class). Hence:

auc_lr1 <- performance(prediction_lr1, "auc")@y.values

auc_lr2 <- performance(prediction_lr2, "auc")@y.values

# Then, I will print them:

print(paste0("The AUC for Model 1 without fare is: ", auc_lr1))

## [1] "The AUC for Model 1 without fare is: 0.794117647058823"

print(paste0("The AUC for Model 2 with logged fare is: ", auc_lr2))

## [1] "The AUC for Model 2 with logged fare is: 0.814911080711354"
```

1.K MODEL PERFORMANCE 2

```
# The best model is the second model (with the logged fare variable). The ROC
curve for Model 2 is "higher" and "closer to the top-left corner" than the
ROC curve for Model 1, demonstrating that Model 2 is more adequate in this
case. Indeed, the AUC is greater for Model 2 than for Model 1, ulteriorly
confirming that Model 2 (with the logged fare) performs better than the model
without the logged fare.

# Source: https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/
```