---
title: "CUNY=SPS-DATA621: Business Analytics and Data Mining\nHomework #3: LOGISTIC REGRESSION"
output: github_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r load-packages, results='hide',warning=FALSE, message=FALSE, echo=FALSE}

library(tidyverse) #loading all library needed for this assignment
#library(openintro)
library(caret)
library(knitr)
library(markdown)
library(rmarkdown)
library(naniar)
library(reshape)
library(ggplot2)
library(qqplotr)
library(stats)
library(statsr)
library(GGally)
library(pdftools)
library(correlation)
library(Metrics)
library(e1071)
library(rocc)
library(pROC)
library(plm)
library(car)
library(VIF)
library(MASS)
data(lpsa)
library(AICcmodavg)

```

Github Master files link: https://github.com/asmozo24/Data621_Logistic_Regression
Web link: https://rpubs.com/amekueko/756445
code link

Overview

In this homework assignment, we will explore, analyze and model a data set information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate

is above the median crime rate (1) or not (0). This assignment is about learning how to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels. We will provide classifications and probabilities for the evaluation data set using the built binary logistic regression model. Let's recall the definition of binary logistic regression with R. According to R-bloggers (https://www.r-bloggers.com/2020/05/binary-logistic-regression-with-r/), "Binary Logistic Regression is used to explain the relationship between the categorical dependent variable and one or more independent variables."

1. DATA EXPLORATION

There are 02 datasets: crime-training-data_modified, crime-evaluation-data_modified provided by Instructor:Nasrin Khansari. These are csv files and we used R-programming language to acquire the 02 datasets pre-stored in Github repository. These 13 variables of interest are all predictors except the variable called "target", which is the response variable, and are already defined within the dataset package(see below). The case study: the crime level for various neighborhoods of a major city.

```{r, echo=FALSE}
variable_names <-
read.csv("https://raw.githubusercontent.com/asmozo24/Data621_Logistic_Regression/main/Variable_description.csv", stringsAsFactors=FALSE)
variable_names %>% kable()

```
Data Structure

These datasets include 446 observations and 13 variables. All values are numerical of type integer or double excepted the "chas" and response variable. The "chas" and response variables are categorical variables because of the binary input values(1 or 0). We called the dataframe “training_df” which contains 12 predictors and 01 response variables.

Data overlook
```{r, echo=FALSE}

training_df <-
read.csv("https://raw.githubusercontent.com/asmozo24/Data621_Logistic_Regression/main/crime-training-data_modified.csv", stringsAsFactors=FALSE)
evaluation_df <-
read.csv("https://raw.githubusercontent.com/asmozo24/Data621_Logistic_Regression/main/crime-evaluation-data_modified.csv", stringsAsFactors=FALSE)
str(training_df)
#glimpse(training_df)
#dim(training_df)
training_df %>%
    head(05)%>%
    kable()
#view(training_df)
summary(training_df)

```

2.Data Preparation

It is important to check the missing values before applying regression analysis because missing values can increase the error and add bias to the regression model. As we can see below, the dataset shows no missing values. This means the dataset is good for analysis. In addition, the variable called 'target' = whether the crime rate is above the median crime rate (1) or not (0) is a two level response, so we want to set it as factor as well as chas(even it is not the response variable).

```{r, echo=FALSE}
# changind data type from int to factor

training_df1 <-training_df
training_df1$target <- as.factor(training_df$target)
training_df1$chas <- as.factor(training_df$chas)
evaluation_df1 <- evaluation_df
evaluation_df1$chas <-as.factor(evaluation_df$chas)
evaluation_df1$target = NA
evaluation_df1$target <-as.factor(evaluation_df1$target)



#str(training_df)

misValues <- sum(is.na(training_df))# Returning the column names with missing values
column_na <- colnames(training_df)[ apply(training_df, 2, anyNA) ] # 2 is dimension(dim())

cat("The total of missing values is : ", misValues)
cat("\nThe total of missing 'NA' is: ")
column_na

missing.values <- function(df){
    df %>%
    gather(key = "variables", value = "val") %>%
    mutate(is.missing = is.na(val)) %>%
    group_by(variables, is.missing) %>%
    summarise(number.missing = n()) %>%
    filter(is.missing==T) %>%
    select(-is.missing) %>%
    arrange(desc(number.missing))
}

#missing.values(training_df)%>% kable()
#missing.values(evaluation_df) %>% kable()
# plot missing values
# missing.values %>%
#   ggplot() +
#     geom_bar(aes(x=variables, y=number.missing), stat = 'identity') +
#     labs(x='variables', y="number of missing values", title='Number of missing values') +
#   theme(axis.text.x = element_text(angle = 45, hjust = 1))

#Another way of visualizing missing value
#vis_miss(training_df)
#gg_miss_var(training_df, show_pct = TRUE) + labs(y = "Missing Values in %")+ theme()
#colSums(is.na(training_df))##%>% kable()
```

```
cat("\n Even more, the table below shows the total number of missing values per variable")
apply(is.na(training_df), 2, sum)
```

Data Distribution

We want to take a look at how the data are distributed across all variables. We see that the response variable(target) has a binomial normal distribution.This makes sense because the response variable only has 02 outputs(0 and 1). Beside the average number of rooms per dwelling(rm) which as a normal distribution, the rest of variables show right and left skewed. Based on these density plots, we want to see what the variance between predictors and response variable look like.

```{r, echo=FALSE}
training_df %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density() +                       # as density
  theme_dark()
cat("\nAnother way of looking at the data distribution for all 13 variables.")
df1 <- c(1:1, 2:13)
df2 <- training_df[ , -14]
par(mfrow = c(3,5))
for (i in df1) {
  #hist(X[ ,i], xlab = names(X[i]), main = names(X[i]))
  d <- density(df2[,i])
  plot(d, main = names(df2[i]))
  polygon(d, col="blue")
}
```

Among the boxplots (below), we can see that the (nox) =median for nitrogen oxides concentration (parts per 10 million) places the neighborhood at risk for high crime levels. We also see that chas (a dummy var, for whether the suburb borders the Charles River (1) or not (0)) and zn(proportion of residential land zoned for large lots (over 25000 square feet)) are irrelevant for statistical analysis since the median is insignificant.

```{r, echo=FALSE}

#training_df$target <- factor(training_df$target)
#training_df$chas <- factor(training_df$chas)

# df1a <- training_df[training_df$control==0,]
# df1b <- c("zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "lstat", "medv")
# par(mfrow = c(4,3))
#
#
```

```
# for(i in df1b){
# boxplot_df <- df1a[,c(i,"target")]
# colnames(boxplot_df) <- c("i","target")
# boxplot(i ~ target,xlab="target",ylab=i, col = "blue", data=boxplot_df)
# }

# par(mfrow=c(4,3))
# for (i in 1:12) {
#       #boxplot(training_df[,i], main=names(training_df[i]), type="h",horizontal = FALSE)
#       #boxplot(i ~ target,xlab="target",ylab=i, col = "blue", data=training_df)
#       #boxplot(training_df$target,training_df[,i], xlab="target",ylab=i, col = "blue", data=training_df , horizontal
= FALSE,col=(c("blue","yellow")))
#
# }

#
# p2 <- melt(training_df, id.vars = 'target') %>%
#   ggplot(., aes(x=variety, y=note)) +
#   geom_boxplot(aes(fill=target)) +
#   facet_wrap(~variety, dir ='h', scale="free")+
#   labs(title="Predictors Vs. Response Variable")

par(mfrow = c(2,3))
boxplot(zn~target, ylab="zn", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(indus~target, ylab="indus", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(chas~target, ylab="chas", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(nox~target, ylab="nox", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(rm~target, ylab="rm", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(age~target, ylab="age", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(dis~target, ylab="dis", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(rad~target, ylab="rad", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(tax~target, ylab="tax", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(ptratio~target, ylab="ptratio", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(lstat~target, ylab="lstat", xlab= "target", col=(c("light blue", "blue")),data = training_df)
boxplot(medv~target, ylab="medv", xlab= "target", col=(c("light blue", "blue")),data = training_df)




```

3.Build Models

Model 1- Backward Elimination

Model 1 accounts for all variables. To build model 1, we could split the training_df into train (80%) and test(20%) using the createDataPartition() function. However, we have the evaluation_df that we can use for the prediction. So, instead of partitioning or sampling, we will keep 0.912 of training_df1 for train and evaluation_df for test.

```{r, echo=FALSE}
```

```
train1 <-training_df1

#evaluation_df1$target = as.factor(evaluation_df1$target)


# Training the model
logit_l <- glm(target ~ ., family = binomial(), train1)
#checking the model
summary(logit_l)
```

Interpreting Model 1 Output

Something strange happened: the chas variable changed to chas1.
Looking at the coefficient, we can say that for every increase in nox, the log odds of being high risk(crime level) increases by 49.122297, and similarly for the other variables.

Most of the variables show to be significant. There are few variables(zn, chas, indus, rm and lstat) with higher pvalue (Pvalue>0.05). These variables should be removed as there are non-significant to the model.

We can see from the logit_1 output that adding 12 (465-453 =12) independent variables decreased the deviance from 645.88 to 192.05, which is a significant reduction in deviance. The Residual Deviance has reduced by 645.88-192.05=453.83 with a loss of 12 degrees of freedom.

The Fisher Scoring iterations indicates the number of iterations perform in model 1. This is an indication that the model converged after 09 iterations which tells us about some level of model goodfit.

Model 2 Backward Elimination (reduced variables)

For the second model, we will exclude variables(zn, chas, indus, rm and lstat) for being insignificant. This is a backward selections. As we started all variables, we look at those variables that do not contribute to the model. These variables tend to have high pvalues (pvalue >0.05 on the 95% confidence interval)

```{r, echo=FALSE}

#removing variable by names
#train1 <- subset(training_df, select=-c(zn, chas))
train2 <- train1 %>%
  dplyr::select(-zn, -chas, -indus, -rm, -lstat)

evaluation_df2 <- evaluation_df1 %>%
  dplyr::select(-zn, -chas, -indus, -rm, -lstat)


# Training the model
logit_2 <- glm(target ~ ., family = binomial(), train2)
summary(logit_2)
```

```

#### Interpreting Model 2 Output

Model 2 has less independent variables compared to the initial training dataset. All the predictors are highly significant (pvalue<0.05). We also observed that the intercept has increased while the coefficient has decreased. So, for nox variable , which has the lowest pvalue and can better explain the response variable, y2 = 42.338378x2-36.824228,y1 = 49.122297x1-40.822934 , one unit nitrogen from model 2 gives 5.51415 while model 1 gives 8.299363 contribution toward determining neighborhood crime level.

#### Model 3 Stepwise Regression

For the model 3, we want to try the stepwise method since model 2 output shows that all variables (07) are significant to the built model.

```{r, echo=FALSE}

model3 <- glm(target ~ ., family = binomial(), train1)%>% stepAIC(trace=FALSE)

#stepModel3 <- step(model3)
summary(model3)

#removing variable by names
#train1 <- subset(training_df, select=-c(zn, chas))
require(mosaic)
require(Stat2Data)
require(leaps)

# output the 07 best models for each number of predictors
#model3 <- regsubsets(target ~ ., data=training_df1, nbest=2, nvmax = 6, method = "seqrep")
#with(summary(model3), data.frame(rsq, adjr2, cp, rss, outmat))

#response_df <- training_df1['target']  # Y variable
#independent_df <- training_df1[, !names(training_df1) %in% "target" ]  # X variables

#model3 <- leaps(x=independent_df, y=training_df1$target, nbest = 2, method = "adjr2") # we could use "Cp", "adjr2", "r2".

# we want to choose a model with 07 variables.
#select_vars <- models$which[7, ]  # pick selected variables
#new_df <- cbind(response_df, independent_df[, select_vars()])  # new data for building selected model
#logit_3 <- lm(target ~ ., data=new_df)  # build model
#summary(logit_3)

```

Interpreting Model 3 Output
```

Model 3 has about 08 variables of the initial training dataset. All variable are significant and the AIC value did drop. We could also find the variables that best explain the response variable. This could have been possible with a more robust stepwise method called Leaps. We also wonder if this could have been capture early by checking the correlation among these variable.

4. Model Selection

We will use cross check The Akaike information criterion (AIC) for all models. We can recall that the AIC is an estimator of prediction error and thereby relative quality of statistical models for a given set of data. AIC estimates the quality of each model, relative to each of the other models. Model 3 is the best model because has the lowest AIC. We also tried the anova , but encountered some error in the output.

```{r, echo=FALSE}
logit_l$aic
logit_2$aic
model3$aic

train3 <- train1 %>%
  dplyr::select(-chas, -indus, -rm, -lstat)

evaluation_df3 <- evaluation_df1 %>%
  dplyr::select(-chas, -indus, -rm, -lstat)

modela <- aov(target ~ ., data = train1)
modelb <- aov(target ~ ., data = train2)
modelc <- aov(target ~ ., data = train3)

# model_list <- list(modela, modelb, modelc)
# model_name <- c("modela", "modelb", "modelc")
 anova(modela, modelb, modelc)
# aictab(model_list, modnames = model_name)

```

We can visualize the distribution of predicted probability of high level of crime by plotting histogram.

```{r , echo=FALSE}

summary(model3$fitted.values)

hist(model3$fitted.values,main = " Distribution of predicted probability of high level of crime ",xlab =
"Probability of high Crime", col = 'blue')

```

4. Select Models

Cros-validation for model performance. We put Model3 through the confusionMatrix to find out abou the model accuracy. We encountered many code errors which results in using the evaluation_df. Perphaps, the evaluation needs some data cleaning.

```{r, echo=FALSE}
set.seed(11121)
# Splitting the data into train and test
index <- createDataPartition(train3$target, p = .912, list = FALSE)
train3a <- train3[index, ]
train3b <- train3[-index, ]
#str(train1b)
#str(train1)


# library(caret)
# trControl <- trainControl(method = "repeatedcv",
#                           repeats = 3,
#                           classProbs = TRUE,
#                           number = 10,
#                           savePredictions = TRUE,
#                           summaryFunction = twoClassSummary)
#
#   model3a <- train(target~.,
#                        data=train1,
#                        method="glmStepAIC",
#                        family = "binomial",
#                        direction ="backward",
#                        trControl=trControl)



#
# train_model3 = train(target ~., data = train1,
#              method = "glmStepAIC", family = "binomial",trace=FALSE,
#              preProcess = c("center", "scale"),trControl = trainControl(
#                 method = "cv", number = 10,
#                 savePredictions = TRUE,
#
#                 ),tuneLength = 5)
# summary(train_model3)

# Predicting in the test dataset
#model3a_Pred <- predictionFunction(method = train_model3$modelInfo, modelFit =
train_model3$finalModel, newdata = #evaluation_df1, preProc = train_model3$preProcess)

evaluation_df3_Predi <- predict(model3, type = "response", newdata = evaluation_df3)

evaluation_df3_Predi = ifelse(evaluation_df3_Predi>.5,1,0)
#Issue with the evaluation_df3_Predi not being the same size as train1b
length(evaluation_df3_Predi)
length(train3b$target)
```

```
confusionMatrix(factor(evaluation_df3_Predi),factor(train3b$target),dnn = c("Predicted", "Trained"))

# Something wrong with this code
# train_model3 <- train(training_df1)
# test_evaluation_df3 <-test(evaluation_df1)
# train_model3a <- glm(target ~ ., family = binomial(), train_model3)%>% stepAIC(trace=FALSE)
#
# evaluation_df3_Pred<- predict(train_model3a, type = "response", newdata = test_evaluation_df3)
# evaluation_df3$predicted =  as.factor(ifelse(test_evaluation_df3$ >= 0.5, 1, 0))


#model3a <- glm(target ~ ., family = binomial(), train3)%>% stepAIC(trace=FALSE)

#evaluation_df3$pred<- predict(model3, evaluation_df3, type = "response")
#evaluation_df3$predicted =  as.factor(ifelse(evaluation_df3$pred >= 0.5, 1, 0))

#confusionMatrix(evaluation_df3$predicted, train3$target, positive = '1')

plot(roc(train3b$target, evaluation_df3_Predi),print.auc = TRUE)


```

```{r , echo=FALSE}

set.seed(1121)
# Splitting the data into train and test
index <- createDataPartition(train2$target, p = .912, list = FALSE)
train2a <- train2[index, ]
train2b <- train2[-index, ]
#str(train1b)
#str(train1)


# Predicting in the test dataset
#model3a_Pred <- predictionFunction(method = train_model3$modelInfo, modelFit =
train_model3$finalModel, newdata = #evaluation_df1, preProc = train_model3$preProcess)

evaluation_df2_Predi <- predict(logit_2, type = "response", newdata = evaluation_df2)

evaluation_df2_Predi = ifelse(evaluation_df2_Predi>.5,1,0)
#Issue with the evaluation_df3_Predi not being the same size as train1b
#length(evaluation_df2_Predi)
#length(train2b$target)

confusionMatrix(factor(evaluation_df2_Predi),factor(train2b$target),dnn = c("Predicted", "Trained"))
plot(roc(train2b$target, evaluation_df2_Predi), print.auc = TRUE)
```

```
```{r, echo=FALSE}
set.seed(1721)
# Splitting the data into train and test
index <- createDataPartition(train1$target, p = .912, list = FALSE)
train1a <- train1[index, ]
train1b <- train1[-index, ]
#str(train1b)
#str(train1)


# Predicting in the test dataset
#model3a_Pred <- predictionFunction(method = train_model3$modelInfo, modelFit =
train_model3$finalModel, newdata = #evaluation_df1, preProc = train_model3$preProcess)

evaluation_df1_Predi <- predict(logit_l, type = "response", newdata = evaluation_df1)

evaluation_df1_Predi = ifelse(evaluation_df1_Predi>.5,1,0)
#Issue with the evaluation_df3_Predi not being the same size as train1b
#length(evaluation_df3_Predi)
#length(train1b$target)

confusionMatrix(factor(evaluation_df1_Predi),factor(train1b$target),dnn = c("Predicted", "Trained"))
plot(roc(train1b$target, evaluation_df1_Predi), print.auc = TRUE)

```


Conclusion

After running the ConfusionMatrix, we found some results that did not correlate to the previous finding. Using AIC method, we found that model 3 was the best among the 03 models when we used stepwise regression. We got a much more better AIC compared to the other models. The confusionMatrix shows that model 2 is the best since the model accuracy is greater than model 1 and 3. This is rather strange than conclusive. Anova method or Leaps could have split this model selection if much more simplicity was applied on the data.

https://www.r-bloggers.com/2020/05/binary-logistic-regression-with-r/

https://towardsdatascience.com/implementing-binary-logistic-regression-in-r-7d802a9d98fe

https://www.scribbr.com/statistics/akaike-information-criterion/

https://cran.r-project.org/web/packages/caret/vignettes/caret.html

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.567.459&rep=rep1&type=pdf#:~:text=The%20categorical%20dependent%20variable%20here,OLS%20is%20biased%20and%20inefficient.
```