```r
# ----------------------------------------------------------------------
# Purpose: Practical for Logistic Regression Models in R
# Author : Liew How Hui (2022)
# Reference & Data:
#  1. https://www.statlearning.com/resources-second-edition
# License: BSD-3
# Software: R 4.x & R 3.6
# Duration: 1 hour
# Remark: Make sure you do the programming instead of running
#         the R script only.  Programming questions do come out in
#         the final exam.
# ----------------------------------------------------------------------

#install.packages("ISLR2")
library(ISLR2)

# Exploring the One-hot encoding?
# model.matrix ???
#install.packages("caret")
# The caret library depends on more than 20 and a lot of packages
# One-hot encode --> retain only the features and not sale price
#full_rank = caret::dummyVars(Sale_Price˜., data=???, fullRank=TRUE)

# ----------------------------------------------------------------------
# Performance Measures for Binary Classification
# ----------------------------------------------------------------------
performance = function(xtab, desc=""){
    cat(desc,"\n")
    ACR = sum(diag(xtab))/sum(xtab)
    TPR = xtab[1,1]/sum(xtab[,1]); TNR = xtab[2,2]/sum(xtab[,2])
    PPV = xtab[1,1]/sum(xtab[1,]); NPV = xtab[2,2]/sum(xtab[2,])
    FPR = 1 - TNR                ; FNR = 1 - TPR
    # https://standardwisdom.com/softwarejournal/2011/12/confusion-matrix-another-single-va
lue-metric-kappa-statistic/
    RandomAccuracy = (sum(xtab[,2])*sum(xtab[2,]) +
      sum(xtab[,1])*sum(xtab[1,]))/(sum(xtab)^2)
    Kappa = (ACR - RandomAccuracy)/(1 - RandomAccuracy)
    print(xtab)
    cat("\n      Accuracy :", ACR, "\n\n         Kappa :", Kappa, "\n")
    cat("\n   Sensitivity :", TPR,   "\n   Specificity :", TNR, "\n")
    cat("Pos Pred Value :", PPV,     "\nNeg Pred Value :", NPV, "\n")
    cat("           FPR :", FPR,     "\n           FNR :", FNR, "\n")
}

# ----------------------------------------------------------------------
#  Logistic Regression Analysis of the ISLR2’s ‘Smarket’ Dataset
# ----------------------------------------------------------------------

### Explore the dataset
#View(Smarket)       # From ISLR
names(Smarket)       # or colnames
summary(Smarket)     # Except for the 1st & last columns, the rests are numerics
#
# pairs = scatter plots of every pair of the columns
# Purpose: try to rule out correlation
# => correlation is BAD for Logistic Regression
#
pairs(Smarket)       # Scatter plots of all columns
# Pair plot is only OK when we have less than 15 or so numeric columns
#cor(Smarket)        # Won’t work, Direction is numeric
cor(Smarket[,-9])    # Remove variable Direction
#
# Correlation can be visualised using HeatMap.
#
# ‘Some’ correlation between Year and Volume
plot(Smarket$Year,Smarket$Volume)

### Split data into train set and validation set
### For time series, we always split the data into ‘past’ and ‘present’
### train set = data from Year 2001-2004
### validation set = data from Year 2005
train = (Smarket$Year < 2005)     # For Boolean selection
Smarket.2005 = Smarket[!train,]   # !TRUE == FALSE
dim(Smarket.2005)     # Check the dimension of the testing data table
Y.2005 = Smarket.2005$Direction

### Fit the train set into logistic regression (parametric predictive model)
# Output / Target / Response = Direction (Interested to see how price go)
# Inputs / Factors / Predictors / Independent Variables = various variations
#
# Why no ’Today’?  Because Up / Down is BASED on Today & they are correlated
#
lr.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
             data=Smarket, subset=train, family=binomial)
### Equivalent expression
#logreg.fits = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
#                   data=Smarket[train,], family=binomial)
summary(lr.fit)

### Apply model into validation set and predict the probability to be Class 1
# Response = P(Y=1 | X=x) = 1/(1 + exp(-(DefaultT)))
# DefaultT = beta0 + beta1*x1 + ... + betap*xp
Y.probs = predict(lr.fit, newdata=Smarket.2005, type="response")
# contrasts is used for the construction of one-hot encoding
contrasts(Smarket$Direction)    # To show the value (1/0) for level (Up/Down)
### Make prediction based on the probability computed (>=0.5 is Up)
yhat = ifelse(Y.probs < 0.5, "Down", "Up")

### Construct confusion matrix and performance measures
cfmat  = table(yhat,Y.2005)
performance(cfmat, "Performance of Logistic Regression Model on Smarket Data")

# ---------------------------------------------------------------------
#  Model Comparison for Badly Fitted Data does not make sense
# ---------------------------------------------------------------------

# The p-values of the coefficients suggest NO variable is
# able to "explain" the output nicely.
#
# Try the model with only one input Lag1 and compare it to the full model
#
lr.fit2 = glm(Direction~Lag1, data=Smarket[train,], family=binomial)
summary(lr.fit2)
#
# The residual deviance improves a bit but the model is still bad
#
anova(lr.fit, lr.fit2)
#
# Chisq, Rao, tests are OK for binary output:
# The result suggests bad model.
#
anova(lr.fit, lr.fit2, test="Cp")      # Cp == AIC
anova(lr.fit, lr.fit2, test="Chisq")  # Same thing as LRT for logistic regression
anova(lr.fit, lr.fit2, test="LRT")    # LRT = Likelihood Ratio Test
anova(lr.fit, lr.fit2, test="Rao")


# ---------------------------------------------------------------------
#  Analysis of the original ‘Fraud’ Dataset using Logistic Regression glm
# ---------------------------------------------------------------------

#https://liaohaohui.github.io/MEME19903/fraud.csv
fraud = read.csv("fraud.csv")
### change data type from integer to categorical
col_fac = c("gender", "status", "employment", "account_link", "supplement", "tag")
fraud[col_fac] = lapply(fraud[col_fac], factor)
```

```
### Manual stratified sampling
### Ref: https://stackoverflow.com/questions/23479512/stratified-random-sampling-from-data-
frame
set.seed(123)
fraud_tag0 = fraud[fraud$tag=="0", ]
fraud_tag1 = fraud[fraud$tag=="1", ]
tag0_idx = sample(nrow(fraud_tag0), size=0.7*nrow(fraud_tag0))
tag1_idx = sample(nrow(fraud_tag1), size=0.7*nrow(fraud_tag1))
fraud.train = rbind(fraud_tag0[ tag0_idx,],fraud_tag1[ tag1_idx,])
fraud.test  = rbind(fraud_tag0[-tag0_idx,],fraud_tag1[-tag1_idx,])
summary(fraud.test)

### logistic regression (use the data without normalization)
logreg_model = glm(tag~.-id_person, data=fraud.train, family=binomial)
summary(logreg_model)

## Remove those with p-value > 0.05
#logreg_model = glm(tag~.-id_person-base_value, data=fraud.train, family=binomial)
#summary(logreg_model)

fraud.test.prob = predict(logreg_model,
  newdata=subset(fraud.test,select=1:8), type='response')
#fraud.test.prob = predict(logreg_model, newdata=fraud.test[ ,1:8], type='response')
yhat = ifelse(fraud.test.prob < 0.5, "pred_0", "pred_1")
cfmat = table(yhat, fraud.test$tag)
performance(cfmat, "Performance of the Logistic Regression Model")

cat("
# ---------------------------------------------------------------------
#  Analysis of the standardised 'Fraud' Dataset using Logistic Regression
# ---------------------------------------------------------------------
")
summary(fraud)

#
# Standardisation
#
#normalise.vec <- function(column,ref.col) {
#    return ((column - mean(ref.col)) / sd(ref.col))
#}
fraud.tran.std     = fraud.train
fraud.test.std     = fraud.test
#fraud.tran.std$age = normalise.vec(fraud.tran.std$age, fraud.train$age)
#fraud.test.std$age = normalise.vec(fraud.test.std$age, fraud.train$age)
mu_age = mean(fraud.train$age)
si_age = sd(  fraud.train$age)
fraud.tran.std$age = scale(fraud.tran.std$age,mu_age,si_age)[,1]
fraud.test.std$age = scale(fraud.test.std$age,mu_age,si_age)[,1]

mu_bsv = mean(fraud.train$base_value)
si_bsv = sd(  fraud.train$base_value)
fraud.tran.std$base_value = scale(fraud.tran.std$base_value, mu_bsv, si_bsv)[,1]
fraud.test.std$base_value = scale(fraud.test.std$base_value, mu_bsv, si_bsv)[,1]

model.for.scaleddata = glm(tag~., data=fraud.tran.std[,2:9], family=binomial)
summary(model.for.scaleddata)

#
# Logit/link can be used instead of probability in prediction
#
fraud.test.logit = predict(model.for.scaleddata, fraud.test.std[,2:8])
yhat = ifelse(fraud.test.logit > 0, "pred_1", "pred_0")
cfmat = table(yhat, fraud.test$tag)
performance(cfmat, "Performance of the Logistic Regression Model")

# ---------------------------------------------------------------------
#  Model Comparison for Nicely Fitted Data makes sense
# ---------------------------------------------------------------------

# Is removing the base_value and age better?
reduced.model = glm(tag~ gender + status + employment + account_link + supplement, data=fra
ud.tran.std[,2:9], family=binomial)
summary(reduced.model)
# What about removing employment as well?
rm2 = glm(tag~ gender + status + account_link + supplement, data=fraud.tran.std[,2:9], fami
ly=binomial)
summary(rm2)

anova(model.for.scaleddata, reduced.model, rm2, test="Cp")
anova(model.for.scaleddata, reduced.model, rm2, test="Chisq")
anova(model.for.scaleddata, reduced.model, rm2, test="Rao")

# ---------------------------------------------------------------------
#  Model Comparison of the full model to Multinomial LR
# ---------------------------------------------------------------------

library(nnet)
mlr.model = multinom(tag ~ ., data=fraud.tran.std[,2:9])
print(mlr.model)
print(summary(mlr.model))

# Basically multinom() witk K=2 is the same as glm() except Z-statistic
# and p-values are not provided.
```