Shiva Sankar Modala | A20517528 | CSP 571 Spring 2023

Data Preparation and Analysis Assignment 2

Recitation Exercises:
Chapter 4
Exercise 4 Answers:

a. Given,
If X ∈ [0.05,0.95] then the observations are in the interval [X−0.05, X+0.05].
Length = 0.1 (a fraction of 10%) based on the information provided, which tells us that X ∈ [0.05,0.95].
If X<0.05, we will use observations that occur within the range [0, X+0.05] that demonstrate the fraction of (100X + 5)%.
By a similar argument it is concluded that if X>0.95, then the percentage of observations we will employ is (105 - 100X)%.

$$\int_{0.05}^{0.95} 10\,dx + \int_{0}^{0.05} (100x + 5)dx + \int_{0.95}^{1} (105 - 100x)dx$$

= 9.5 - 0.5 + 0.125 +0.25 + 55 - 54.625
= 9.75
≈ 10%

The fraction of available observations used to make the prediction is 9.75% ≈ 10%.

b. If we assume that X_1 and X_2 to be independent for the given data and in accordance with the logic from (a), then the fraction of the available observations to make a prediction can be expressed as follows:
= 9.75% x 9.75%
= 0.950625%
≈ 1%

c. The fraction of the available observations to make a prediction can be expressed as follows using the same explanation as (a) and (b):
$(9.75\%)^{100} \approx 0\%$

d. Similar to the answers from (a) to (c), the number of features, p, is the proportion of available observations we will use to make the predictions, which is $(9.75\%)^p$. Therefore, if p is large, let's say, we have

$$\lim_{p\to\infty} (9.75\%)^p = 0$$

Hence, we can say that the fraction of available observations to make predictions decreases as p increases.

e. We have l = 0.1 for p = 1,
$l = (0.1)^{1/2} = 0.316$ for p = 2, and
$l = (0.1)^{1/100} = 0.977$ for p = 100.

Each side of the cube must extend further into range of each of the p dimensions to fall 10% of the training observations within the cube. These 10% of training observation data is not close.

Exercise 6 Answers:

a. Given that X1 = 40 and X2 = 3.5.

We know for Logistic Regression with multiple variables,

$$p(X) = \frac{e^{\beta_0 + K_1\beta_1 + K_2\beta_2}}{1 + e^{\beta_0 + K_1\beta_1 + K_2\beta_2}}$$

Given: $\beta_0 = -6$, $\beta_1 = 0.05$, $\beta_2 = 1$, $e = 2.71828$

$\hat{p}(X) = e^{-6+0.05X_1+X_2} / (1+e^{-6+0.05X_1+X_2})$

$= \frac{e^{0.5}}{1+e^{-0.5}}$

$= 0.3775$

$=37.75\%$

b. In this problem we have given P(X)= 0.5 and we need to find X1 :

$0.5 = \frac{e^{-6+(0.05)(X1)+(1)(3.5)}}{1+e^{-6+(0.05)(40)+(1)(3.5)}}$

$0.5 + 0.5 * e^{(0.05)x1-2.5} = e^{(0.05)x1-2.5}$

$= e^{(0.05)x1-2.5}=1$

Applying the log on both sides,

$\log(1) = \log(e^{(0.05)x1-2.5})$

$X1 = \frac{2.5}{0.05}$

X1 = 50

Hence students need to study for 50 hours to have a 50 % chance of getting an A in the class.

Exercise 7 Answers:

Bayes' theorem states that

$\prod(\Theta|x) = \frac{\prod(\Theta).f(x|\Theta)}{f(x)}$

$\prod(4) = \frac{0.8e^{-\left(\frac{1}{72}\right)(4-10)^2}}{0.8e^{-\left(\frac{1}{72}\right)(4-10)^2}+0.2e^{-\left(\frac{1}{72}\right)(4-0)^2}} = 0.752 = 75.2\%$

Therefore, if a company's percentage return from the previous year was X=4, there is a 0.752 percent chance that it will declare a dividend this year.

Exercise 9 Answers:

a. $\frac{P(x)}{1-P(x)} = 0.37$

P(x) = 0.37 – 0.37*P(x)

P(x) = 0.27

We have on average 27% people with an odds of 0.37 of defaulting on their credit card payment.

b. $\frac{P(x)}{1-P(x)} = \frac{0.16}{1-0.16} = 0.19$
She has a 19% chance of default.

Chapter 5
Exercise 2 Answers:

a. $1 - \frac{1}{n} = \frac{(n-1)}{n}$
Justification: There are n observations to choose from, which is a given. As a result, 1/n is the likelihood that the j^{th} observation will be the initial bootstrap observation.

b. There will always be n observations for selection since bootstrap sampling is sampling with replacement, and the likelihood that the j^{th} observation will not be the second bootstrap observation is the same as it is in part (a).
$1 - \frac{1}{n} = \frac{(n-1)}{n}$

c. Bootstrap sampling is replacement sampling. Therefore, the likelihood that observation j will not be included in the final sample.
It can be calculated as follows: = Probability that observation j will not be the first observation, second observation, third observation, etc.
= (1 – (1/n)) * (1 – (1/n)) * (1 – (1/n)) *…….. up to n times
$= (1 - (1/n))^n$

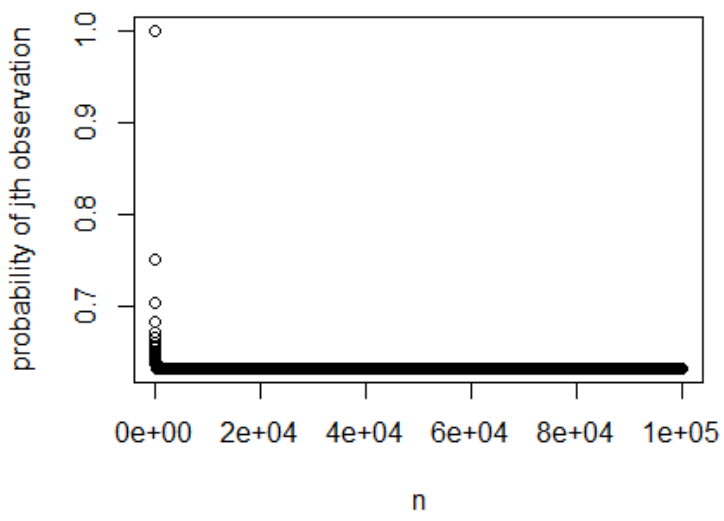
d. The probability that an observation will not be included in the bootstrap sample is given by the formula $(1 - (1/n))^n$ as demonstrated in part (c). On the other hand, we can provide the following formula for the likelihood that the j^{th} observation is in the bootstrap sample:
$= 1 - (1 - (1/n))^n$
$= 1 - (1 - (1/5))^5$
= 1 - 0.32768
= 0.67232

e. Probability that the j^{th} observation is in the bootstrap sample = $1 - (1 - (1/n))^n$
$= 1 - (1 - (1/100))^{100}$
= 1 – 0.36604
= 0.63396

f. Probability that the j^{th} observation is in the bootstrap sample = $1 - (1 - (1/n))^n$
$= 1 - (1 - (1/10000))^{10000}$
= 1 – 0.36786
= 0.63214

g. Code:
```
n = 1:100000
y=sapply(n,function(n){1-((1-(1/n))^n)})
plot(n,y, xlab="n",ylab = "probability of jth observation")
```

```r
n = 1:100000
y=sapply(n,function(n){1-((1-(1/n))^n)})
plot(n,y, xlab="n",ylab = "probability of jth observation")
```



Comment: According to the plot, the probability that the jth observation will be included in the bootstrap sample is roughly equal to 0.63, and it stays linear as n increases.

h. Code:

```r
obs <- 10000
n <- 1:100
strap <- rep(NA, obs)
for (i in 1:obs) {
  strap[i] <- sum(sample(n, rep = T) == 4) > 0
}
mean(strap)
```

Output:

```
> mean(strap)
[1] 0.6392
```

Comment:

The mean indicates the percentage of bootstrap samples that include the fourth observation. As we saw in question part (e), the mean provides a proportion that is roughly equal to $(1 - (1/100))^{100}$, or 0.63.

Exercise 3 Answers:

a. In K fold cross validation, n observations are randomly divided into k groups or folds. Each fold has a length of roughly n/k, and the groups do not overlap. This method treats the first fold

as a validation set and the method is fit on the remaining k-1 folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This process iterates k times, and each iteration treats a new group as a validation set. This procedure results in k estimates of the test error, MSE_1, MSE_2, ..., MSE_k. The k-fold CV estimate is computed by averaging these values,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$ **.... (For Quantitative Response)**

b.

i. Advantages: The validation set approach is easy to understand and to implement.

Disadvantages: Comparing the validation set approach to k-fold cross-validation, there are two major downsides. First off, there can be a lot of variation in the validation estimate of the test error rate (depending on precisely which observations are included in the training set and which observations are included in the validation set). Second, the model is fitted using only a portion of the observations. This shows that the validation set error rate may have a tendency to overstate the test error rate for the model fit on the complete data set because statistical methods typically perform poorly when trained on fewer observations.

ii. Advantages: Due to the randomness of the splitting procedure, the validation approach results in different MSE when used repeatedly, whereas LOOCV has minimal bias because we split based on just one observation each time.

Disadvantages: The LOOCV cross-validation method is an exception to the general rule of k-fold cross-validation when k=n. Comparing this method to k-fold cross-validation has two disadvantages. In contrast to k-fold cross-validation, which simply calls for fitting the model k times, it necessitates fitting the possibly computationally expensive model n times. Due to the fact that each training set has n1 observations, the LOOCV cross-validation approach may provide roughly unbiased estimates of the test error; nonetheless, this method has a higher variance than k-fold cross-validation (since we are averaging the outputs of n fitted models trained on an almost identical set of observations, these outputs are highly correlated, and the mean of highly correlated quantities has higher variance than less correlated ones)

2. Practicum Problems

(I'm attaching the pdf markdown files I created from r script.)

Abalone

Shiva Sankar Modala

2023-02-09

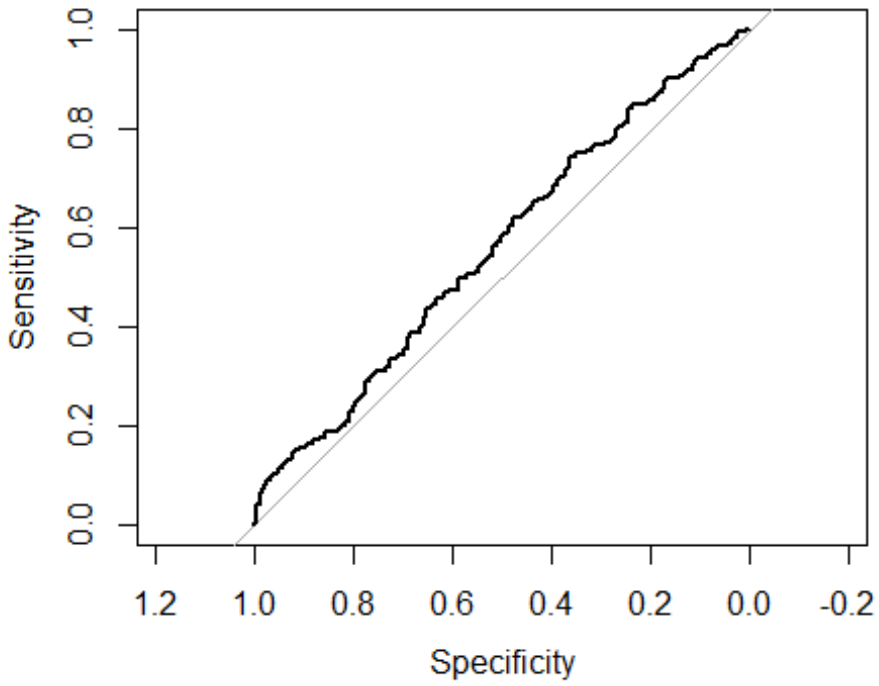
```
## Installing the necessary packages for the problem ##

#install.packages('readr')     ## abalone.data is a large dataset. So, I used
readr package in handling that data
#install.packages('knitr')     ## To convert the r script into the markdown
ans later for presentation, Knit is used for documentation.
#install.packages('stringr')   ## It provides a cohesive set of functions
designed to work with strings easily
#install.packages('caret')     ## To use machine learning models, I used
caret package to fit our model
#install.packages('corrplot')  ## With the corrplot, I can provide the
correlation matrix for our data.
#install.packages('pROC')      ## For the ROC curves and analysis.

## These are the libraries that I used for the abalone data.
library(readr)
library(knitr)
library(stringr)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(corrplot)

## corrplot 0.92 loaded

library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

# Reading the abalone data as the csv format
data_abalone= read.csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/abalone/abalone.data',header = FALSE,sep = ",",stringsAsFactors =
TRUE)
```

```r
# Remove the Infants in the observations by keeping the Male/Female classes
infant_remove = subset(data_abalone,V1!='I')
infant_remove$V1 = factor(infant_remove$V1)
set.seed(1)

# With the help of createDataPartition() in the caret package, we split the
data into 80% and 20%.
partition_data = createDataPartition(infant_remove$V1,p=0.2,list=FALSE)

# Dividing the test data and train data by separating the columns.
# test data has the infant data with the data part
test_data = infant_remove[partition_data,]
# Train data is without that data part
train_data = infant_remove[-partition_data,]

# Fit a logistic regression using all feature variables using the generalized
linear models
# I used glm to apply that model to the data
log_regression = glm(V1~V2+V3+V4+V5+V6+V7+V8+V9,data=train_data,family =
binomial)

# Summary for the above logistic regression
summary(log_regression)
```

```
##
## Call:
## glm(formula = V1 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9, family =
binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8773  -1.1995   0.8723   1.1165   1.5184
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.858543   0.520622   5.491 4.00e-08 ***
## V2          -0.629068   2.292027  -0.274   0.7837
## V3          -6.633627   2.709837  -2.448   0.0144 *
## V4          -3.732314   2.249421  -1.659   0.0971 .
## V5          -0.745165   0.854026  -0.873   0.3829
## V6           4.055672   1.027483   3.947 7.91e-05 ***
## V7          -1.041244   1.442155  -0.722   0.4703
## V8           1.368821   1.299135   1.054   0.2920
## V9           0.001171   0.018057   0.065   0.9483
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3128.9  on 2266  degrees of freedom
```

```
## Residual deviance: 3064.5  on 2258  degrees of freedom
## AIC: 3082.5
##
## Number of Fisher Scoring iterations: 4

# Coefficient for the above logistic regression
coef(log_regression)

##  (Intercept)           V2           V3           V4           V5
V6
##  2.858543140 -0.629067560 -6.633626737 -3.732313567 -0.745165230
4.055671602
##           V7           V8           V9
## -1.041243887  1.368820970  0.001170528

cat("\n The null hypothesis can be avoided for the variables for which the
predictions have a lower p-value")

##
##  The null hypothesis can be avoided for the variables for which the
predictions have a lower p-value

cat("\n We can tell from the output that V3 and V6 are the important
predictors.")

##
##  We can tell from the output that V3 and V6 are the important predictors.

# Now we have to present the confidence intervals for the logistic regression
confint(log_regression)

## Waiting for profiling to be done...

##                    2.5 %      97.5 %
## (Intercept)   1.85352256  3.89549890
## V2           -5.12145416  3.86968345
## V3          -11.96790846 -1.33704996
## V4           -8.56672822 -0.04177129
## V5           -2.44078468  0.91942538
## V6            2.05920944  6.09531362
## V7           -3.86758608  1.79335994
## V8           -1.17091097  3.93439914
## V9           -0.03424511  0.03659261

cat("\n Confidence interval does not contain 0 for V6 but it does for V3. V6
has 95% chance that +  predictor V6 falls between range 2.05920944 &
6.09531362 and we can reject the null hypothesis.")

##
##  Confidence interval does not contain 0 for V6 but it does for V3. V6 has
95% chance that +  predictor V6 falls between range 2.05920944 & 6.09531362
and we can reject the null hypothesis.
```

```r
# The type as response provides the predicted probabilities
predic1= predict(log_regression,test_data,type="response")

# Create a new variable for the male and female and this can help us in
making the confusion matrix
predic = ifelse(predic1>=0.5,'M','F')

# Confusion matrix for predictor for the test dataset.
confusionMatrix(as.factor(predic),as.factor(test_data$V1))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   F   M
##          F  97  85
##          M 165 221
##
##                Accuracy : 0.5599
##                  95% CI : (0.5179, 0.6012)
##     No Information Rate : 0.5387
##     P-Value [Acc > NIR] : 0.1665
##
##                   Kappa : 0.0945
##
##  Mcnemar's Test P-Value : 5.841e-07
##
##             Sensitivity : 0.3702
##             Specificity : 0.7222
##          Pos Pred Value : 0.5330
##          Neg Pred Value : 0.5725
##              Prevalence : 0.4613
##          Detection Rate : 0.1708
##    Detection Prevalence : 0.3204
##       Balanced Accuracy : 0.5462
##
##        'Positive' Class : F
##

# plotting the ROC curve for the predictor
plot(roc(test_data$V1,predic1))

## Setting levels: control = F, case = M

## Setting direction: controls < cases
```




```
cat("\n As we can see ROC curve is better for our model")
```

```
##
##  As we can see ROC curve is better for our model
```

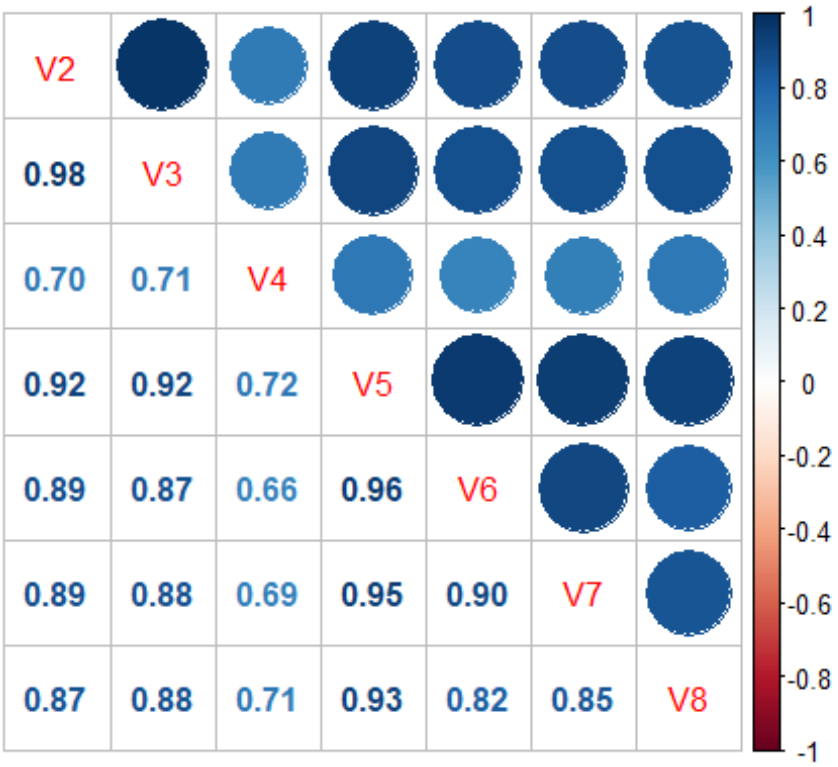
```
cat("hence it will predict better than selecting random value")
```

```
## hence it will predict better than selecting random value
```

```
cat("Accuracy of the model is 0.5599")
```

```
## Accuracy of the model is 0.5599
```

```
# plotting the mixed Correlation plot for the model
corrplot.mixed(cor(infant_remove[,2:8]))
```




```
# Conclusion
cat("\n Given that the above plot donesn't explain much, the strong
correlation between all the variables demonstrates the classifier's poor
performance")

##
##  Given that the above plot donesn't explain much, the strong correlation
between all the variables demonstrates the classifier's poor performance

cat("\n A good model has uncorrelated variables.")

##
##  A good model has uncorrelated variables.
```

agaricus_lepiota

Shiva Sankar Modala

2023-02-10

```
# Installing the package caret
# install.packages('caret')

# Reading the data in csv format for the agaricus lepiota from the mushroom
package
mush_data = read.csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/mushroom/agaricus-
lepiota.data',header=FALSE,sep=",",stringsAsFactors = TRUE)

# Loading the library for the statistics and the probability package.
library(e1071)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

# Finding the missing values in our dataset.
values_missing = which(mush_data$V12=='?')

# Assigning a new variable for the dataset excluding the missing data
MisingsVal_Mushroom = mush_data[-c(values_missing)]

# We can replace the missing values with either removing those
# or we can replace missing values with mode of the data
mush_mode = (table(as.vector(MisingsVal_Mushroom$V12)))
replaceVal_mush = mush_data

# Replacing the missing value with the character 'b'
replaceVal_mush$V12[values_missing]= 'b'

# To train the data, we need to split the given dataset.
# So, I have applied 80% for the training and the rest for the testing.
miss_index = sample(1:nrow(MisingsVal_Mushroom),size =
0.8*nrow(MisingsVal_Mushroom))

# This is the train and test for the data without replacing
miss_train = mush_data[miss_index,]
miss_test = mush_data[-miss_index,]

# To train the data we need to split the given dataset with the replaced
data.
# So, I have applied 80% for the training and the rest for the testing.
Replace_index=sample(1:nrow(replaceVal_mush),size =
```

```
0.8*nrow(replaceVal_mush))
Replace_train = mush_data[Replace_index,]
Replace_test = mush_data[-Replace_index,]

# Apply the naive bayes classifier for both our data with missing values
miss_naiveBayes = naiveBayes(V1~.,data=miss_train)
# Apply the naive bayes classifier for both our data with replaced values
replace_naiveBayes = naiveBayes(V1~.,data=Replace_train)

# Apply the predict function for our classifier in both the test and train
data.
Miss_test_pred= predict(miss_naiveBayes,miss_test)
Miss_train_pred = predict(miss_naiveBayes,miss_train)
# Similarly apply the same predict function for the train and test for the
replaced data.
Replace_test_pred = predict(replace_naiveBayes,Replace_test)
Replace_train_pred = predict(replace_naiveBayes,Replace_train)

# With the confusion matrix we can find the false positives that the model
produced.

### These output values are subjective and can change when we re-run the
program.
# So, there can be a slight change in the values everytime we re-run the
program.

# Confusion Matrix
confusionMatrix(table(Miss_test_pred,miss_test$V1),dnn=c("Predicted","Actual"
))
```

```
## Confusion Matrix and Statistics
##
##
## Miss_test_pred   e   p
##              e 867  87
##              p   1 670
##
##                Accuracy : 0.9458
##                  95% CI : (0.9337, 0.9563)
##     No Information Rate : 0.5342
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8904
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9988
##             Specificity : 0.8851
##          Pos Pred Value : 0.9088
##          Neg Pred Value : 0.9985
##              Prevalence : 0.5342
```

```
##          Detection Rate : 0.5335
##    Detection Prevalence : 0.5871
##       Balanced Accuracy : 0.9420
##
##        'Positive' Class : e
##
```

```
cat("\n The accuracy for the missing values for the test is 0.9458")
```

```
##
##  The accuracy for the missing values for the test is 0.9458
```

```
cat("\n The false positive for the test is 94.")
```

```
##
##  The false positive for the test is 94.
```

```
# Similarly for the training data
confusionMatrix(table(Miss_train_pred,miss_train$V1),dnn=c("Predicted","Actua
l"))
```

```
## Confusion Matrix and Statistics
##
##
## Miss_train_pred    e    p
##               e 3315  361
##               p   25 2798
##
##                Accuracy : 0.9406
##                  95% CI : (0.9346, 0.9462)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8808
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9925
##             Specificity : 0.8857
##          Pos Pred Value : 0.9018
##          Neg Pred Value : 0.9911
##              Prevalence : 0.5139
##          Detection Rate : 0.5101
##    Detection Prevalence : 0.5656
##       Balanced Accuracy : 0.9391
##
##        'Positive' Class : e
##
```

```
cat("\n The accuracy for the missing values for the training is 0.9406")
```

```
##
##  The accuracy for the missing values for the training is 0.9406
```

```
cat("\n The false positive for the train is 358.")
```

```
##
##  The false positive for the train is 358.
```

```
# With the confusion matrix we can find the false positives that the model
produced.
```

```
confusionMatrix(table(Replace_test_pred,Replace_test$V1),dnn=c("Predicted","A
ctual"))
```

```
## Confusion Matrix and Statistics
##
##
## Replace_test_pred   e   p
##                 e 841  81
##                 p   8 695
##
##                Accuracy : 0.9452
##                  95% CI : (0.933, 0.9558)
##     No Information Rate : 0.5225
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8898
##
##  Mcnemar's Test P-Value : 2.312e-14
##
##             Sensitivity : 0.9906
##             Specificity : 0.8956
##          Pos Pred Value : 0.9121
##          Neg Pred Value : 0.9886
##              Prevalence : 0.5225
##          Detection Rate : 0.5175
##    Detection Prevalence : 0.5674
##       Balanced Accuracy : 0.9431
##
##        'Positive' Class : e
##
```

```
cat("\n The accuracy for the replaced values for the test is 0.9452.")
```

```
##
##  The accuracy for the replaced values for the test is 0.9452.
```

```
cat("\n The false positive for the test is 103.")
```

```
##
##  The false positive for the test is 103.
```

```r
# With the confusion matrix we can find the false positives that the model produced.
confusionMatrix(table(Replace_train_pred,Replace_train$V1),dnn=c("Predicted", "Actual"))
```

```
## Confusion Matrix and Statistics
##
##
## Replace_train_pred    e    p
##                  e 3335  367
##                  p   24 2773
##
##                Accuracy : 0.9398
##                  95% CI : (0.9338, 0.9455)
##     No Information Rate : 0.5168
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8791
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9929
##             Specificity : 0.8831
##          Pos Pred Value : 0.9009
##          Neg Pred Value : 0.9914
##              Prevalence : 0.5168
##          Detection Rate : 0.5132
##    Detection Prevalence : 0.5696
##       Balanced Accuracy : 0.9380
##
##        'Positive' Class : e
##
```

```r
cat("\n The accuracy for the replaced values for the training is 0.9398.")
```

```
##
##  The accuracy for the replaced values for the training is 0.9398.
```

```r
cat("\n The false positive for the train is 350.")
```

```
##
##  The false positive for the train is 350.
```

```r
## Once again these output values are subjective and can change when we re-run the program
# So, there can be a slight change in the values that are different from what you can see in the cat statement.
```

yacht_hydrodynamics

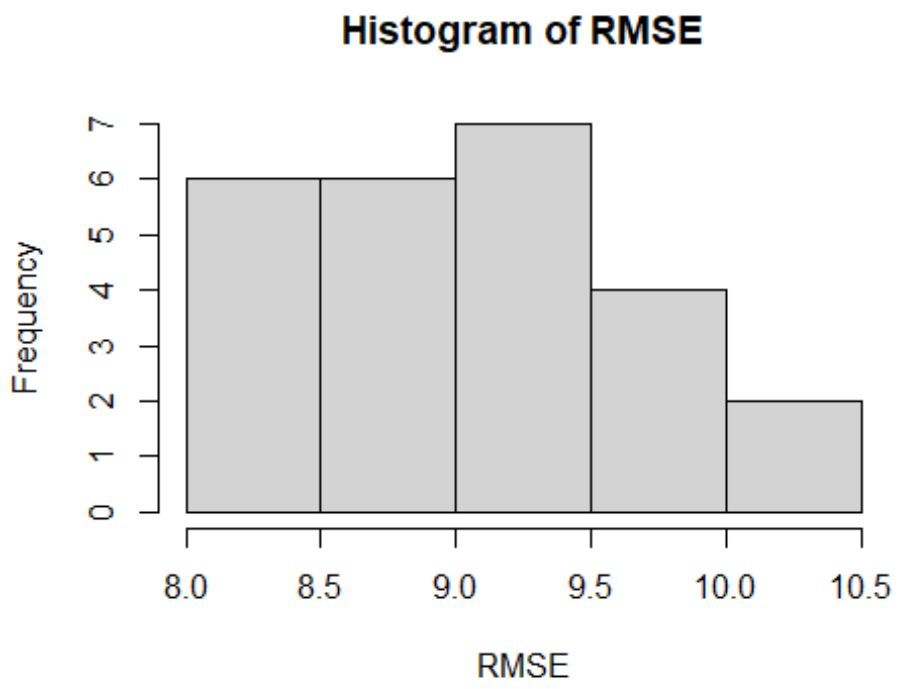
Shiva Sankar Modala

2023-02-10

```
## Installing the necessary packages for the problem ##

#install.packages('readr')     ## yacht_hydrodynamics.data is a large
dataset. So, I used readr package in handling the data
#install.packages('caret')     ## To use machine learning models, I used
caret to fit our model
#install.packages('ggplot2')   ## used ggplot2 for better visualizations of
data
#install.packages('lattice')   ## Lattice is used to implement the trellis
graphics for our data

# Loading the libraries
library(readr)
library(data.table)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)
library(lattice)

# Reading the yacht_hydrodynamics.data as the table without the header
yacht_hydrodynamics = read.table("https://archive.ics.uci.edu/ml/machine-
learning-databases/00243/yacht_hydrodynamics.data", header = F)

# Assigning the column names for our dataset
names(yacht_hydrodynamics) = c("longitude","Prismatic","displacement","beam-
draught","beamlenght","fraude","residuary")
head(yacht_hydrodynamics)

##   longitude Prismatic displacement beam-draught beamlenght fraude
residuary
## 1      -2.3     0.568         4.78         3.99       3.17  0.125
0.11
## 2      -2.3     0.568         4.78         3.99       3.17  0.150
0.27
## 3      -2.3     0.568         4.78         3.99       3.17  0.175
0.47
## 4      -2.3     0.568         4.78         3.99       3.17  0.200
0.78
## 5      -2.3     0.568         4.78         3.99       3.17  0.225
```

```
1.18
## 6      -2.3     0.568         4.78         3.99       3.17  0.250
1.82

# Creating the data partition for our data having 80% our data for the
training. So the rest 20% is for testing.
#  I used the caret package to perform a 80/20 test-train split
cd = createDataPartition(y = yacht_hydrodynamics$residuary , p = 0.8, list =
FALSE)

# Separating the dataset for the train data
train_data = yacht_hydrodynamics[cd,]

# Separating the test data without the output label data.
test_data = yacht_hydrodynamics[-cd,]

# Applying the linear regression model for the dataset
# Applying the multiple linear regression
lm1 = lm(yacht_hydrodynamics$residuary~yacht_hydrodynamics$longitude +
yacht_hydrodynamics$Prismatic +
             yacht_hydrodynamics$displacement + yacht_hydrodynamics$`beam-
draught` + yacht_hydrodynamics$`beam-draught` +
             yacht_hydrodynamics$displacement + yacht_hydrodynamics$fraude,
              data = train_data)

# creating a function for the mean square error
mse = function(y, yt){
  return (mean((y - yt)^2))
}

# Applying the mean square error for the residuary and the fitted values for
the linear regression model.
msee = mse(yacht_hydrodynamics$residuary, lm1$fitted.values )
msee

## [1] 78.47651

cat("\n The MSE for the training data is = ", msee)

##
##  The MSE for the training data is =  78.47651

cat("\n The Root mean square error for the train data is = ", sqrt(msee))

##
##  The Root mean square error for the train data is =  8.858697

cat("\n The summary for the r-squared data for the linear model is  =
",summary(lm1)$r.sq)

##
##  The summary for the r-squared data for the linear model is  =  0.6574487
```

```r
# train control specify the resampling scheme
# I used the caret package to perform a bootstrap from the full sample
dataset with N=1000 samples
train = trainControl(method = "boot", number = 1000)

lm2 = train(residuary~., data = train_data, method = "lm" )

# summary of the model
summary(lm2$resample$RMSE)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.243   8.679   9.306   9.109   9.478  10.337

summary(lm2$resample$Rsquared)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5690  0.6301  0.6483  0.6457  0.6580  0.7025

# Plotting a histogram for the resampled data and the root mean square error
hist(lm2$resample$RMSE, xlab = "RMSE", main = "Histogram of RMSE")
```




```r
# applying the mean for the resampled data as the mse2
mse2 = mean(lm2$resample$RMSE)^2
mse2

## [1] 82.96775

cat("\n Training MSE for the bootstrap model  is = ", mse2)
```

```
##
##  Training MSE for the bootstrap model  is =  82.96775
```

```
cat("\n Training RMSE for the bootstrap model is  ", mean(lm2$resample$RMSE))
```

```
##
##  Training RMSE for the bootstrap model is   9.108663
```

```
cat("\n Training Mean R-squared for the bootstrap model is
",mean(lm2$resample$Rsquared))
```

```
##
##  Training Mean R-squared for the bootstrap model is  0.6457281
```

```
predVals_boot = predict(lm2,test_data)
```

```
cat("\n From the above observations, there is no difference in performance
between the original and bootstrap models.")
```

```
##
##  From the above observations, there is no difference in performance
between the original and bootstrap models.
```

german

Shiva Sankar Modala

2023-02-10

```
# Loading the necessary libraries
library(readr)
library(data.table)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

#Load the German Credit Data sample dataset from the UCI Machine Learning
Repository (german.data-numeric) into R using a dataframe in the table format
creditGermData<-read.table("https://archive.ics.uci.edu/ml/machine-learning-
databases/statlog/german/german.data-numeric",header = FALSE)
set.seed(100)
creditGermData$V25 = factor(creditGermData$V25)

# I used the caret package to perform a 80/20 test-train split using the
createDataPartition()
train_Index = createDataPartition(y = creditGermData$V25 , p = 0.8, list =
FALSE)

# Separating the Training data
train_Data = creditGermData[train_Index,]

# Separating the Testing data
testData = creditGermData[-train_Index,]

# obtain a training fit for a logistic model via the glm()
logisticModel = glm(V25~.,family=binomial,data=train_Data)
actualVals = train_Data$V25

# 50% cut-off factor so that the  probabilities  > 0.5 are 2 and rest are 1
fittedVals = ifelse(logisticModel$fitted.values >  0.5,2,1)
fittedVals = factor(fittedVals)

# Gives the confusion matrix for the fitted and train data
cm = confusionMatrix(fittedVals, train_Data$V25)

# The training Precision/Recall and F1 results are:

cat("\n Training Precision: ", cm$byClass[5] * 100, "%")

##
##  Training Precision:  82.16039 %
```

```r
cat("\n Training Recall: ", cm$byClass[6] * 100, "%")
```

```
##
##  Training Recall:  89.64286 %
```

```r
cat("\n Training F1-Score: ", cm$byClass[7] * 100, "%")
```

```
##
##  Training F1-Score:  85.73868 %
```

```r
probs = predict(logisticModel, testData, type = "response")

fittedVals_test = ifelse(probs > 0.5,2,1)
fittedVals_test = factor(fittedVals_test)

cm_test = confusionMatrix(fittedVals_test, testData$V25)
cm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 124  36
##          2  16  24
##
##                Accuracy : 0.74
##                  95% CI : (0.6734, 0.7993)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.122775
##
##                   Kappa : 0.3158
##
##  Mcnemar's Test P-Value : 0.008418
##
##             Sensitivity : 0.8857
##             Specificity : 0.4000
##          Pos Pred Value : 0.7750
##          Neg Pred Value : 0.6000
##              Prevalence : 0.7000
##          Detection Rate : 0.6200
##    Detection Prevalence : 0.8000
##       Balanced Accuracy : 0.6429
##
##        'Positive' Class : 1
##
```

```r
cat("\n Testing Precision: ", cm_test$byClass[5] * 100, "%")
```

```
##
##  Testing Precision:  77.5 %
```

```r
cat("\n Testing Recall: ", cm_test$byClass[6] * 100, "%")
```

```
##
##  Testing Recall:  88.57143 %

cat("\n Testing F1-Score: ", cm_test$byClass[7] * 100, "%")

##
##  Testing F1-Score:  82.66667 %

# use the trainControl and train functions to perform a k=10 fold cross-
validation fit of the same model,
# Define training control
train.control = trainControl(method = "cv", number = 10)

# Training the model
logisticModel2 = train(V25~., data = train_Data, method = "glm", family =
"binomial", trControl =train.control)
fittedVals_cv = ifelse(logisticModel2$finalModel$fitted.values > 0.5,2,1)
fittedVals_cv = factor(fittedVals_cv)

# Confusion matrix
cm_cv = confusionMatrix(fittedVals_cv, train_Data$V25)
cm_cv

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 502 109
##          2  58 131
##
##                Accuracy : 0.7912
##                  95% CI : (0.7614, 0.8189)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 3.653e-09
##
##                   Kappa : 0.4708
##
##  Mcnemar's Test P-Value : 0.0001092
##
##             Sensitivity : 0.8964
##             Specificity : 0.5458
##          Pos Pred Value : 0.8216
##          Neg Pred Value : 0.6931
##              Prevalence : 0.7000
##          Detection Rate : 0.6275
##    Detection Prevalence : 0.7638
##       Balanced Accuracy : 0.7211
##
##        'Positive' Class : 1
##
```

```
cat("\n Training Precision with 10-fold CV: ", cm_cv$byClass[5] * 100, "%")
```

```
##
##  Training Precision with 10-fold CV:  82.16039 %
```

```
cat("\n Training Recall with 10-fold CV: ", cm_cv$byClass[6] * 100, "%")
```

```
##
##  Training Recall with 10-fold CV:  89.64286 %
```

```
cat("\n Training F1-Score with 10-fold CV: ", cm_cv$byClass[7] * 100, "%")
```

```
##
##  Training F1-Score with 10-fold CV:  85.73868 %
```

```
probs_cv = predict(logisticModel2, testData, type = "prob")

# 50% cut-off factor so that the  probabilities  > 0.5 are 2 and rest are 1
fittedVals_cv_test = ifelse(probs > 0.5,2,1)
fittedVals_cv_test = factor(fittedVals_test)
cm_cv_test = confusionMatrix(fittedVals_test, testData$V25)

# cross-validated training Precision/Recall and F1 values.

cat("\n Testing Precision: ", cm_cv_test$byClass[5] * 100, "%")
```

```
##
##  Testing Precision:  77.5 %
```

```
cat("\n Testing Recall: ", cm_cv_test$byClass[6] * 100, "%")
```

```
##
##  Testing Recall:  88.57143 %
```

```
cat("\n Testing F1-Score: ", cm_cv_test$byClass[7] * 100, "%")
```

```
##
##  Testing F1-Score:  82.66667 %
```

```
cat("\n From the above observations, we can observe that both the cross
validation and basic model have same result.")
```

```
##
##  From the above observations, we can observe that both the cross
validation and basic model have same result.
```