

Chapter-4

Dated: Recitation Exercises

Example-4 $p=1$  feature  $X$ . $X$  = evenly distributed on  $[0, 1]$ 

$\Rightarrow$  predict observation that are within  
 - 10% of the range of  $X$ . closest to test  
 observation.

$\Rightarrow$  for  $x = 0.6$  range  $[0.55, 0.65]$

Answer:-

Available observation we can use for prediction is

For,  $X$  Range =  $[0.55, 0.65]$

$$\Rightarrow [0.65 - 0.55] / (1 - 0) * 100$$

$$\Rightarrow 0.10 * 100$$

$$\Rightarrow 10\%$$

So on average we use 10% of observation for prediction.

(b)

$p=2$  Features  $x_1, x_2$  uniformly distributed on  $[0, 1] \times [0, 1]$

$\Rightarrow$  predict test observation response  
 within 10% of  $x_1$  and 10% of  $x_2$

Dated:

⇒ For  $x_1 = 0.6$        $x_2 = 0.33$   
we use Range  $[0.55, 0.65]$  for  $x_1$   
 $[0.3, 0.4]$  for  $x_2$

Answer:-

$$[(0.65 - 0.55)/(1-0)] * 100] + \\ [(0.4 - 0.3)/(1-0)] * 100]$$

$$\Rightarrow [0.10 * 100] + [0.1 * 100]$$

$$\Rightarrow 10\% + 10\%$$

$$\Rightarrow 20\%$$

So, fraction of the available observation  
can be used for prediction is 50%.

(c)

$$\Rightarrow P = 100$$

⇒ uniformly distributed on each features  
in sample  $[0, 1]^2$

⇒ predict test observation's response  
using 10% of each features.

Answer:-

$$\Rightarrow (\text{Observation feature range})^P \text{ percentage.}$$

$$\Rightarrow ((1/12))^100 + 100\%$$

$$\Rightarrow ((0.1)^{100}) + 100\%$$

$$\approx 0\%$$

Dated:

(d) Argue drawback of KNN for large P

$\Rightarrow$  From above (a-c) result

we can conclude that

As value of P increase the fraction of available observation to make prediction decrease.

$\Rightarrow$  Means, when  $P = 100$  percentage of observation for predict is almost 0 that shows KNN perform poor when number of observation is large.

(e)

$$\Rightarrow P = 1, 2, 100$$

$\Rightarrow$  predict test observation by creating a p-dimensional hypercube centered around test observation.

$\Rightarrow$  1 or 2 of training observations.

Answer:-

length of each side of the hypercube

$$\text{For } P = 1$$

hypercube is a line

$$\text{length} = (0.1)^1 = 0.1$$

$$P = 2$$

hypercube is square

$$\text{length} = (0.1)^{\sqrt{2}} = 0.316$$

Dated:

For  $P = 100$   
hypercube is a dimensional cube.  
length =  $(0.1)^{(1/100)}$   
= 0.977

⇒ When we predict test observation that contains lot of training observation means we have large number of features so it will be better to use all the features.

### (6) Chapter-4 Exercise - 6

$x_1$  = hours       $x_2$  = undergraduate GPA  
 $y$  = A grade.

⇒ logistic Regression

⇒ Estimated coefficients     $B_0 = -6$   
 $B_1 = 0.05$   
 $B_2 = 1$ .

### (9) Probability for students who study

for  $x_1 = 40$  hour  
 $x_2 = 3.5$  GPA  
gets an A ?

Answer :-

Dated:

$$\begin{aligned} P(X) &= e^{B_0 + B_1 X_1 + B_2 X_2} / 1 + e^{B_0 + B_1 X_1 + B_2 X_2} \\ &= \frac{e^{-6 + (0+0.5)(40) + (1)(3.5)}}{1 + e^{-6 + (0+0.5)(40) + (1)(3.5)}} \\ &\Rightarrow \frac{e^{-6 + 2 + 3.5}}{1 + e^{6.5}} = \frac{e^{-0.5}}{1 + e^{-0.5}} \\ &= \frac{0.6065}{1.6065} \\ &= 0.3775 \\ P(X) &= 37.75\% \end{aligned}$$

(b) How many hours student ( $x$ ) need to study to have 50% of chance of getting A?

$$P(X) = 50\% \quad X_2 = 3.5$$

$$X_1 = ?$$

$$\begin{aligned} P(X) &= e^{B_0 + B_1 X_1 + B_2 X_2} / 1 + e^{B_0 + B_1 X_1 + B_2 X_2} \\ 0.5 &= \frac{e^{-6 + 0.05 X_1 + 3.5}}{1 + e^{-6 + 0.05 X_1 + 3.5}} \end{aligned}$$

Dated:

1/8/05

$$= 0.5 = e^{2.278}$$

$$= 0.5 = \frac{e^{0.05x_1 - 2.05}}{1 + e^{0.05x_1 - 2.05}}$$

$$x_1 = 50$$

so, student need to study for 50 hours  
to have 50% of chance of getting A

Dated:

## Chapter 4 Exercise-7

mean value for company issued stock dividend  
was  $x = 10$

those who didn't was  $x = 0$

$\Rightarrow$  variance was  $S^2 = 36$

$\rightarrow$  so  $x$  company issued dividend

$\Rightarrow$   $x$  follows normal distribution.

Answer Bayes' theorem

$$P(Y=1 | X=x) = \frac{\prod_k f_k(x)}{\sum_{i=1}^k \prod_i f_i(x)}$$

$$= f_{yes}(x) = \left[ \pm / \sqrt{2\pi S^2} \right] + e^{-(x-\mu)^2/(2S^2)}$$

$$= f_{yes}(x) = \frac{1}{\sqrt{2\pi S^2}} \cdot e^{-\frac{1}{2S^2}(x-\mu)^2}$$

$$= \frac{1}{\sqrt{2\pi(36)}} e^{-\frac{1}{72}(x-10)^2} \quad ; \quad (\underline{x_{yes} = 10})$$

$$\boxed{f_{yes}(x) = \frac{e^{-\frac{1}{72}(x-10)^2}}{\sqrt{72\pi}}} \quad ; \quad x = 4$$

Dated:  $f_{no}(x) = \frac{1}{\sqrt{2\pi} 6^2} \cdot e^{-\frac{(x-N)^2}{48}}$

$$\left[ f_{no}(x) = \frac{1}{\sqrt{72\pi}} \cdot e^{-\frac{x^2}{72}} \right] \quad \begin{array}{l} \text{if } x=9 \\ \text{if } N=0 \Rightarrow x_{no}=0 \\ \dots \end{array}$$

$$P(Y_{yes}/x=9) = \frac{(\pi_{yes} f_{yes}(x))}{(\pi_{no} f_{no}(4) + \pi_{yes} f_{yes}(x))}$$

$$= \frac{0.8 \times e^{-\frac{81}{144}} / \sqrt{72\pi}}{0.2 \times e^{-\frac{81}{72}} / \sqrt{72\pi} + 0.8 \times e^{-\frac{81}{144}} / \sqrt{72\pi}}$$

$$P(Y_{yes}/x=9) = 0.75$$

for  $x=9$  probability is 75%.

## Chapter - 4      Exercise - 9

- a) On average what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

Answer

$$\text{odds} = \frac{P(X)}{[1-P(X)]}$$

$$\text{odds} = 0.37$$

Dated:

$$0.37 = \frac{P(X)}{1 - P(X)}$$

$$\Rightarrow 0.37 - 0.37 P(X) = P(X)$$

$$\Leftrightarrow P(X) + 0.37 P(X) = 0.37$$

$$\Rightarrow P(X)(1.37) = 0.37$$

$$\Rightarrow P(X) = \frac{0.37}{1.37}$$

$$\boxed{P(X) = 0.2700} \Rightarrow 27\%$$

(b) individual has a 16% chance of defaulting on her credit card payment  
what are the odds that she will default?

$$P(X) = 16\% = 0.16$$

$$\text{Odds} = \frac{P(X)}{1 - P(X)}$$

$$= \frac{0.16}{1 - 0.16}$$

$$= \frac{0.16}{0.84}$$

$$\boxed{\text{Odds} = 0.19}$$

Dated: Chapter-5

### Exercise-2

bootstrap sample from a set of n observations.

- 9) What is the probability that the first bootstrap observation is not  $j$ th observation from the original sample?

Answer: number of observation =  $n$ , Every observation in original sample is independent.

$\downarrow p \Rightarrow$  Probability that first bootstrap observation is  $j$ th observation from original sample.

$$= \frac{1}{n}$$

$p' \Rightarrow$  First bootstrap observation is not  $j$ th observation

$$= 1 - p$$

$$= 1 - \frac{1}{n}$$

Answer =  $p' \Rightarrow \frac{n-1}{n}$

- (b) Probability that second bootstrap observation is not the  $i$ th observation from the original sample?

Dated:

Answer

→ total number of observation =  $n$

→ Observation is part of bootstrap

→ In original sample, every observation has equal probability of appearing in each observation.

→ So Probability that second

$p \Rightarrow$  bootstrap observation is  $j$ th observation from original sample is  $= \frac{1}{n}$

$p' \Rightarrow$  not  $j$ th observation from original sample  $= 1 - \frac{1}{n}$

$$= \frac{n-1}{n}$$

(c)  $j$ th Observation is not in bootstrap sample  $\Rightarrow \left(1 - \frac{1}{n}\right)^n$ .

Answer:-

→ We obtain bootstrap Sample from  $n$  Observation

⇒  $P_2$  -  $j$ th observation not in First bootstrap Sample  $= 1 - \frac{1}{n}$

Dated:

$P_2$  = Probability that observation not in  
Second bootstrap Sample =  $1 - \frac{1}{n}$

$P_n$  = P jth observation not in nth  
bootstrap sample =  $1 - \frac{1}{n}$

So

Answer :-

$P(\text{observation is not in bootstrap sample}) =$

$$\Rightarrow P_1 \cdot P_2 \cdot \dots \cdot P_n$$

$$= \left(1 - \frac{1}{n}\right) \cdot \left(1 - \frac{1}{n}\right) \cdot \dots \cdot \left(1 - \frac{1}{n}\right) \dots n \text{ times}$$

$$\Rightarrow \boxed{\left(1 - \frac{1}{n}\right)^n}$$

(1)

$$n = 5$$

$P(j\text{th observation is in bootstrap sample})$

$$\Rightarrow \left(1 - \frac{1}{5}\right)^5$$

$$\Rightarrow \left(1 - \frac{1}{5}\right)^5$$

$$= \left(\frac{4}{5}\right)^5 \Rightarrow 0.672$$

Dated:

(e)  $n = 100$

$P(\text{jth observation is in bootstrap sample})$

$$\Rightarrow \left(1 - \frac{1}{n}\right)^n$$

$$\Rightarrow \left(1 - \frac{1}{100}\right)^{100}$$

$$\Rightarrow 0.634$$

(f)  $n = 10,000$

$P(\text{jth observation is in bootstrap sample})$

$$\Rightarrow \left(1 - \frac{1}{n}\right)^n$$

$$\Rightarrow \left(1 - \frac{1}{10,000}\right)^{10,000}$$

$$\Rightarrow 0.63214$$

(g) Create plot that display,

for each integer value of  $n$  from 1 to 100,000 the probability that the jth observation is in the bootstrap sample.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

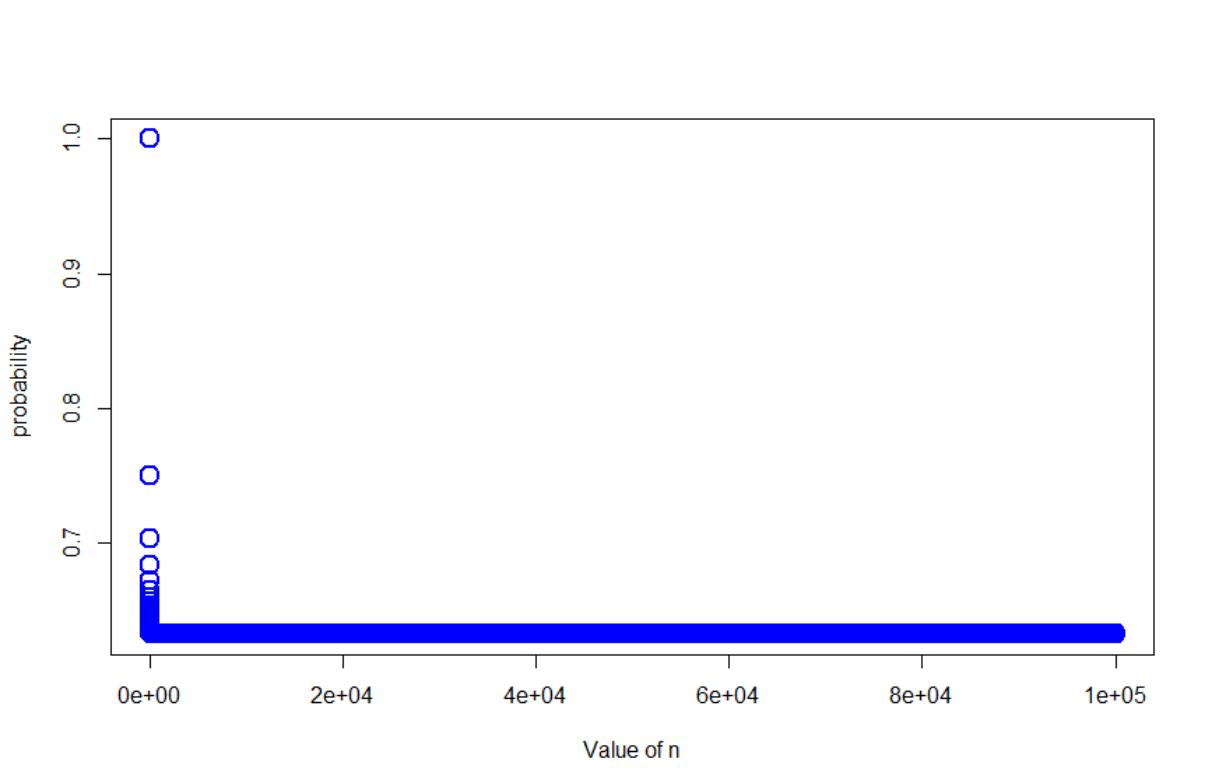
Go to file/function Addins

Source

Console Terminal Background Jobs

R 4.2.1 · ~/

```
> plot(1:100000,1-(1-1/1:100000)^(1:100000),xlab="Value of n",ylab="probability",col="blue",cex=2,lwd=2)
> |
```



$$n = \text{seq}(1, 10000)$$
$$\text{prob} > 6m^2/3 = 62/72$$

Dated:

(h) Numerically the probability that bootstrap sample of size  $n=100$  contains the  $j^{th}$  observation.

$j=4$ . We repeatedly create bootstrap sample, and each time we record whether or not the forth observation is contained in bootstrap sample.

$\Rightarrow$  Comment on the results

$\Rightarrow$  Store = ~~rep~~ rep (NA, 10000)

$\Rightarrow$  for ( $i$  in  $1:10000$ ) {

Store[ $i$ ] = sum (sample(1:100, rep=TRUE  
= = 4) > 0

$\Rightarrow$  mean(store) = 0.6303

Answer We got mean value 0.6303

the observed probability of

$$1 - (1 - 1/100)^4 \cdot 100 = 63.4\% \text{ is}$$

Comparable to our prediction.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal Background Jobs

```
R 4.2.1 · ~/Α  
> store=rep(NA,10000)  
> for (i in 1:10000) { store[i]=sum(sample(1:100,rep=TRUE)==4)>0  
+  
+ }  
> mean(store)  
[1] 0.6303  
> |
```

Dated:

### Exercise - 3

- a) Explain how k-fold cross validation is implemented.

Answer:

- ⇒ k-Fold Cross Validation, divides the n observations into k groups/folds of approximately equal size randomly.
- ⇒ First Fold is treated as validation set
- ⇒ Remaining  $k-1$  folds are used for fitting model. And  $mse_i$  is calculated
- 3 Each group considered as validation set once and a training set for  $(k-1)$  times
- ⇒ This process is repeated  $k$  times
- ⇒ We compute Test error by

$$cv(k) = \left(\frac{1}{k}\right) \sum_{i=1}^k mse_i$$

for  $i = 1$  to  $k$ .

- (b) What are the advantages and disadvantages of k-fold cross validation relative to

Dated:

## 1) Validation set approach?

Advantages:-

This approach is easy to understand and is also easy to implement.

Disadvantages:-

- ⇒ Estimation of test error rate is highly vary depending on which observation are included in training set and which are included in validation set.
- ⇒ We use observation included in training & fit model.
- ⇒ The same Training set is used for fitting model, so if few observation for training, the model will perform worse and overestimate test error.

## 2) LOOCV?

Advantages:-

- ⇒ This model is subset of k-fold cross-validation with  $k=n$ .
- ⇒ So in this model, each observation considered once in validation set and  $(n-1)$  times in training set.
- ⇒ This approach gives approximately unbiased test error estimation.

Dated:

### Disadvantages:-

- ⇒ This model requires fitting of statistical learning method n times.
- ⇒ k-fold cross validation requires k times.
- ⇒ So LOOCV can be expensive.
- ⇒ LOOCV has greater variance than k-fold cross validation model.

# Assignment-2

Shraddha

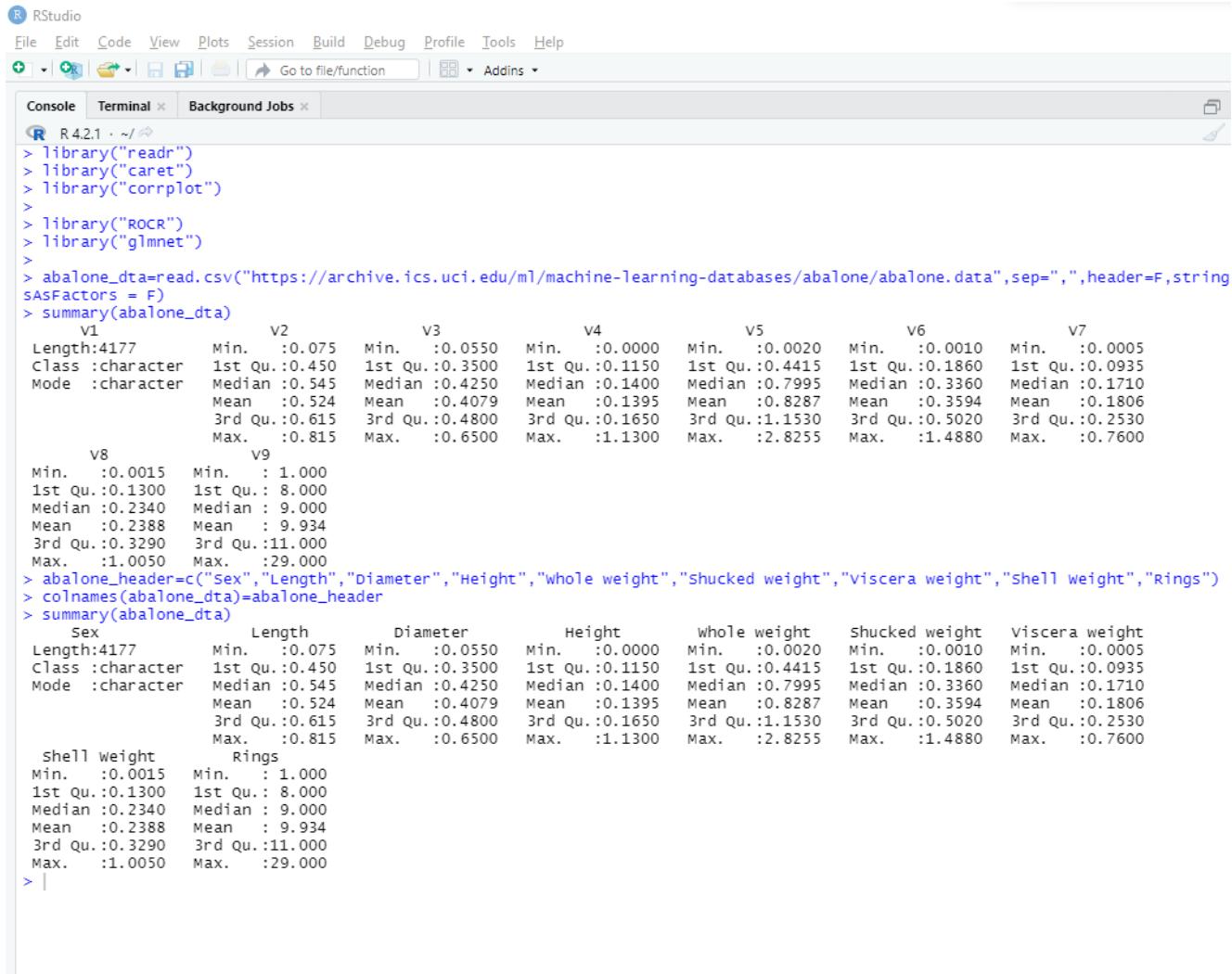
## Practicum Problems

### Problem 1:

Install packages caret, corrplot, glmnet , readr , tidyverse , ROCR using command install. Packages("packages")

Answer:

#### Load abalone Sample Dataset



The screenshot shows the RStudio interface with the console tab selected. The console window displays the R code used to load the abalone dataset and its summary statistics. The code includes library calls for readr, caret, corrplot, ROCR, and glmnet, followed by reading the abalone data from a URL and summarizing it.

```
R4.2.1 - ~/R
> library("readr")
> library("caret")
> library("corrplot")
>
> library("ROCR")
> library("glmnet")
>
> abalone_dta<-read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data",sep=",",header=F,stringsAsFactors = F)
> summary(abalone_dta)
   V1          V2          V3          V4          V5          V6          V7
Length:4177    Min. :0.075    Min. :0.0550   Min. :0.0000   Min. :0.0020   Min. :0.0010   Min. :0.0005
Class :character 1st Qu.:0.450  1st Qu.:0.3500  1st Qu.:0.1150  1st Qu.:0.4415  1st Qu.:0.1860  1st Qu.:0.0935
Mode  :character Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995   Median :0.3360   Median :0.1710
                           Mean :0.524   Mean :0.4079   Mean :0.1395   Mean :0.8287   Mean :0.3594   Mean :0.1806
                           3rd Qu.:0.615  3rd Qu.:0.4800  3rd Qu.:0.1650  3rd Qu.:1.1530  3rd Qu.:0.5020  3rd Qu.:0.2530
                           Max. :0.815   Max. :0.6500   Max. :1.1300   Max. :2.8255   Max. :1.4880   Max. :0.7600
   V8          V9
Min. :0.0015  Min. : 1.000
1st Qu.:0.1300 1st Qu.: 8.000
Median :0.2340 Median : 9.000
Mean   :0.2388 Mean   : 9.934
3rd Qu.:0.3290 3rd Qu.:11.000
Max.   :1.0050 Max.   :29.000
> abalone_header<-c("Sex","Length","Diameter","Height","whole weight","shucked weight","Viscera weight","Rings")
> colnames(abalone_dta)<-abalone_header
> summary(abalone_dta)
   Sex          Length         Diameter        Height       whole weight     shucked weight   Viscera weight
Length:4177    Min. :0.075    Min. :0.0550   Min. :0.0000   Min. :0.0020   Min. :0.0010   Min. :0.0005
Class :character 1st Qu.:0.450  1st Qu.:0.3500  1st Qu.:0.1150  1st Qu.:0.4415  1st Qu.:0.1860  1st Qu.:0.0935
Mode  :character Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995   Median :0.3360   Median :0.1710
                           Mean :0.524   Mean :0.4079   Mean :0.1395   Mean :0.8287   Mean :0.3594   Mean :0.1806
                           3rd Qu.:0.615  3rd Qu.:0.4800  3rd Qu.:0.1650  3rd Qu.:1.1530  3rd Qu.:0.5020  3rd Qu.:0.2530
                           Max. :0.815   Max. :0.6500   Max. :1.1300   Max. :2.8255   Max. :1.4880   Max. :0.7600
   shell weight      Rings
Min. :0.0015  Min. : 1.000
1st Qu.:0.1300 1st Qu.: 8.000
Median :0.2340 Median : 9.000
Mean   :0.2388 Mean   : 9.934
3rd Qu.:0.3290 3rd Qu.:11.000
Max.   :1.0050 Max.   :29.000
> |
```

#### Remove all observations in the Infant category

view(abalone data)

# Assignment-2

Shraddha

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

abalone_dta									
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell Weight	Rings
1	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
2	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
3	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
4	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
5	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
6	I	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.1200	8
7	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.3300	20
8	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.2600	16
9	M	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.1650	9
10	F	0.550	0.440	0.150	0.6945	0.3145	0.1510	0.3200	19
11	F	0.525	0.380	0.140	0.6065	0.1940	0.1475	0.2100	14
12	M	0.430	0.350	0.110	0.4060	0.1675	0.0810	0.1350	10
13	M	0.490	0.380	0.135	0.5415	0.2175	0.0950	0.1900	11
14	F	0.535	0.405	0.145	0.6845	0.2725	0.1710	0.2050	10
15	F	0.470	0.355	0.100	0.4755	0.1675	0.0805	0.1850	10
16	M	0.500	0.400	0.130	0.6645	0.2580	0.1330	0.2400	12
17	I	0.355	0.280	0.085	0.2905	0.0950	0.0395	0.1150	7
18	F	0.440	0.340	0.100	0.4510	0.1880	0.0870	0.1300	10

## Keeping Male/Female Classes

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Source

Console Terminal Background Jobs

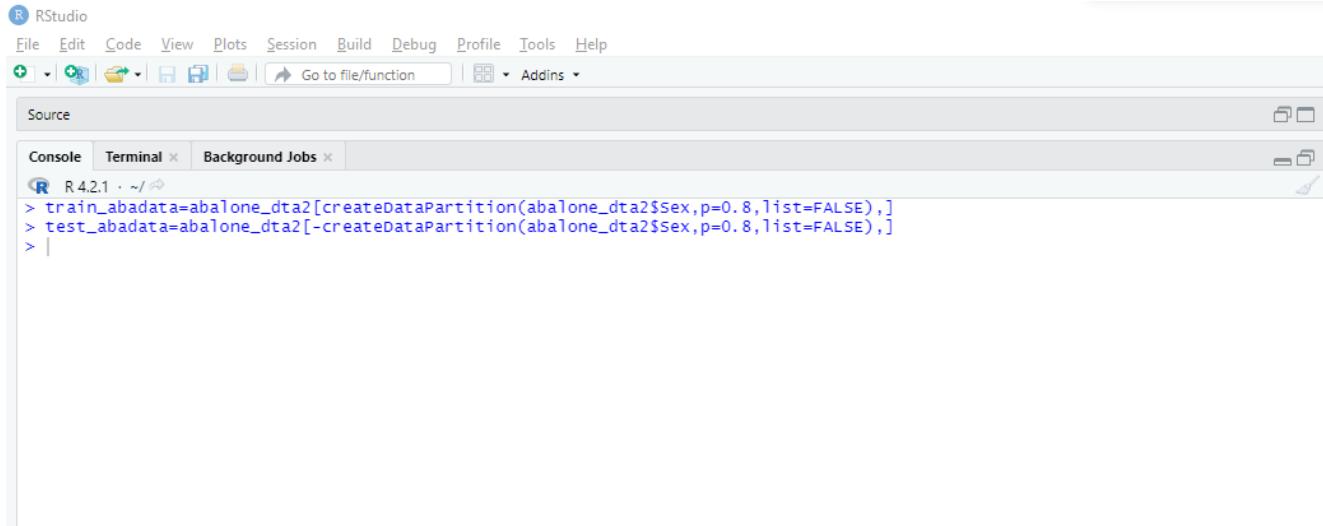
R 4.2.1 - /

```
> unique(abalone_dta$Sex)
[1] "M" "F" "I"
> abalone_dta2=abalone_dta[abalone_dta$Sex!="I",]
> unique(abalone_dta2$Sex)
[1] "M" "F" "I"
> unique(abalone_dta2$Sex)
[1] "M" "F"
> abalone_dta2$sex=factor(abalone_dta2$Sex)
> str(abalone_dta2)
'data.frame': 2835 obs. of 9 variables:
 $ Sex      : Factor w/ 2 levels "F","M": 2 2 1 2 1 1 2 1 1 2 ...
 $ Length   : num  0.455 0.35 0.53 0.44 0.53 0.545 0.475 0.55 0.525 0.43 ...
 $ Diameter : num  0.365 0.265 0.42 0.365 0.415 0.425 0.37 0.44 0.38 0.35 ...
 $ Height   : num  0.095 0.09 0.135 0.125 0.15 0.125 0.125 0.15 0.14 0.11 ...
 $ Whole weight : num  0.514 0.226 0.677 0.516 0.777 ...
 $ Shucked weight: num  0.2245 0.0995 0.2565 0.2155 0.237 ...
 $ Viscera weight: num  0.101 0.0485 0.1415 0.114 0.1415 ...
 $ Shell weight : num  0.15 0.07 0.21 0.155 0.33 0.26 0.165 0.32 0.21 0.135 ...
 $ Rings     : int  15 7 9 10 20 16 9 19 14 10 ...
```

## Assignment-2

Shraddha

Using the caret package, use `createDataPartition` to perform an 80/20 test-train split (80% training and 20% testing)



The screenshot shows the RStudio interface with the following details:

- Header:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Addins.
- Source:** Source tab is selected.
- Console Tab:** Active tab, showing the R session output.
- Output:** R 4.2.1 · ~/ ↻  
> train\_abadata=abalone\_dta2[createDataPartition(abalone\_dta2\$sex,p=0.8,list=FALSE),]  
> test\_abadata=abalone\_dta2[-createDataPartition(abalone\_dta2\$sex,p=0.8,list=FALSE),]  
> |

Fit a logistic regression using all feature variables via `glm`, and observe which predictors are relevant

# Assignment-2

Shraddha

The screenshot shows an RStudio interface with the following details:

- Header:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Addins.
- Source Editor:** Shows R code for creating training and testing datasets, fitting a logistic regression model, and summarizing it. The summary output includes:

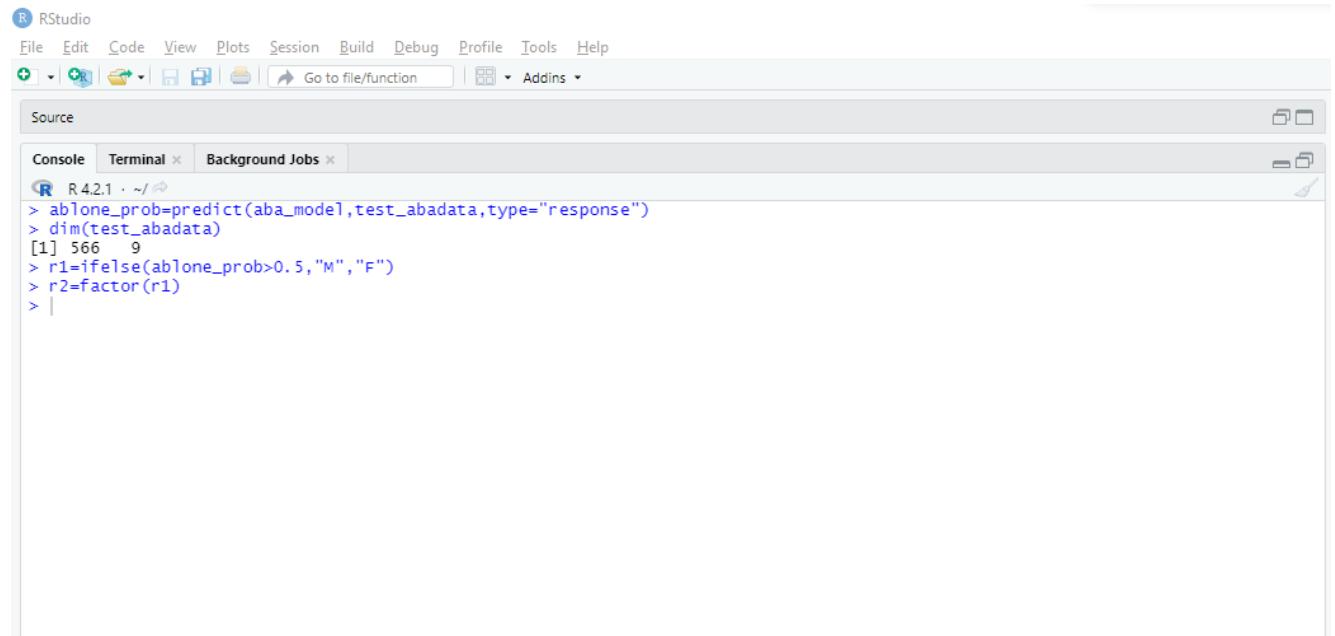
  - Call:** `glm(formula = sex ~ ., family = binomial, data = train_abadata)`
  - Deviance Residuals:** Summary statistics for deviance residuals.
  - Coefficients:** A table showing estimated coefficients, standard errors, z-values, and p-values for each predictor. The predictors listed are Intercept, Length, Diameter, Height, `whole weight`, `shucked weight`, `viscera weight`, and Rings.
  - Signif. codes:** A legend for significance levels: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1.
  - Dispersion parameter for binomial family taken to be 1)**
  - Null deviance:** 3131.7 on 2268 degrees of freedom
  - Residual deviance:** 3072.9 on 2260 degrees of freedom
  - AIC:** 3090.9
  - Number of Fisher Scoring iterations:** 4

- Console Output:** Shows the results of `coef(aba_model)` and `confint(aba_model)`. The `confint` command is noted as "Waiting for profiling to be done..."

The confidence interval result in above image shows that, only "Shucked Weight" predictor has value 0 in confidence interval range which is align with the basic assumption of Null Hypothesis which represent there is no relationship between X and Y. So Null Hypothesis holds true for all the predictors except "Shucked Weight" means this predictor is relevant. Other predictors do not hold relationship with response variable.

# Assignment-2

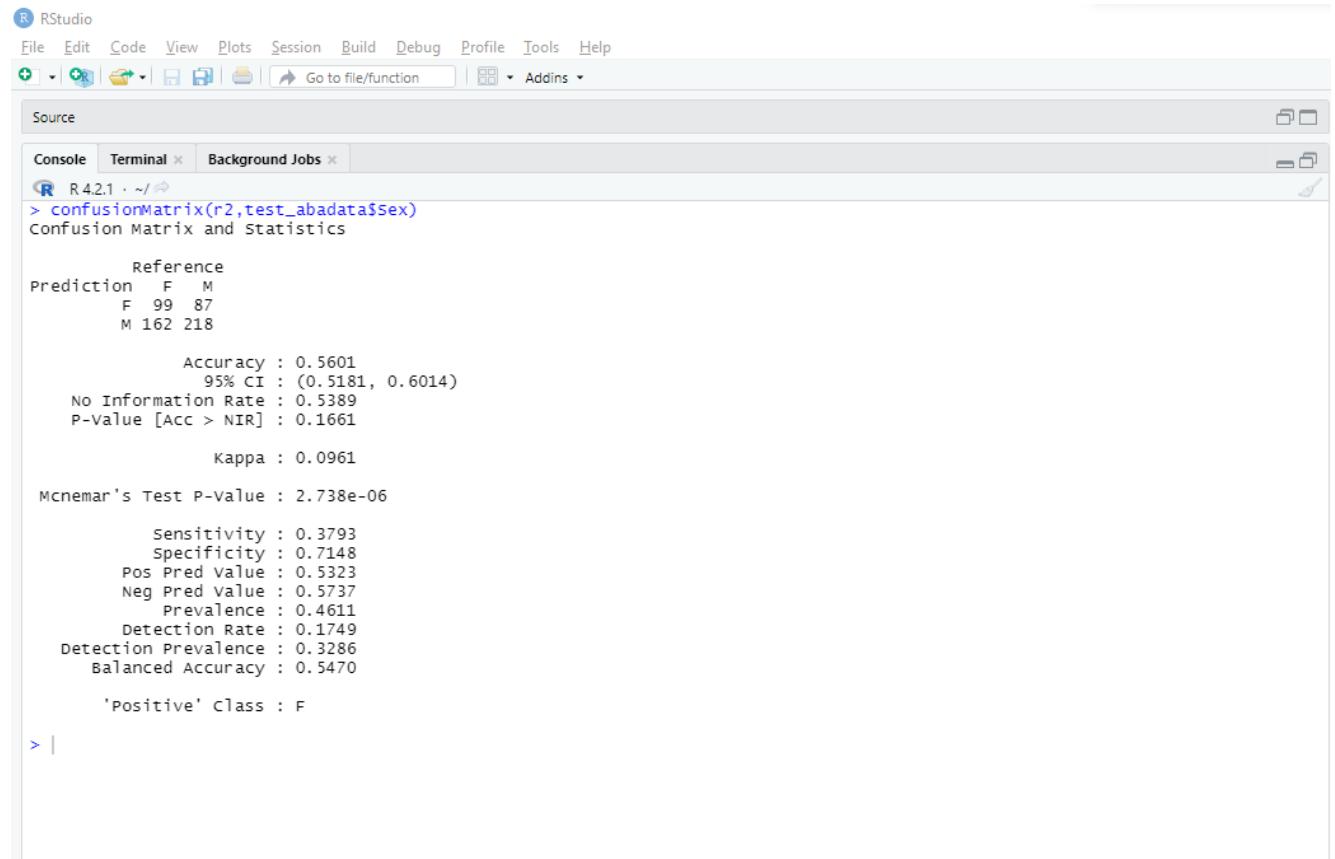
Shraddha



RStudio interface showing R code in the console tab:

```
R 4.2.1 · ~/◊
> abalone_prob=predict(aba_model,test_abadata,type="response")
> dim(test_abadata)
[1] 566   9
> r1=ifelse(abalone_prob>0.5,"M","F")
> r2=factor(r1)
> |
```

## Create Confusion Matrix



RStudio interface showing R code in the console tab:

```
R 4.2.1 · ~/◊
> confusionMatrix(r2,test_abadata$Sex)
Confusion Matrix and Statistics

Reference
Prediction   F   M
      F    99   87
      M   162  218

          Accuracy : 0.5601
          95% CI : (0.5181, 0.6014)
          No Information Rate : 0.5389
          P-value [Acc > NIR] : 0.1661

          Kappa : 0.0961

McNemar's Test P-value : 2.738e-06

          Sensitivity : 0.3793
          Specificity : 0.7148
          Pos Pred Value : 0.5323
          Neg Pred Value : 0.5737
          Prevalence : 0.4611
          Detection Rate : 0.1749
          Detection Prevalence : 0.3286
          Balanced Accuracy : 0.5470

          'Positive' Class : F

> |
```

# Assignment-2

Shraddha

## Plotting ROC Curve

RStudio

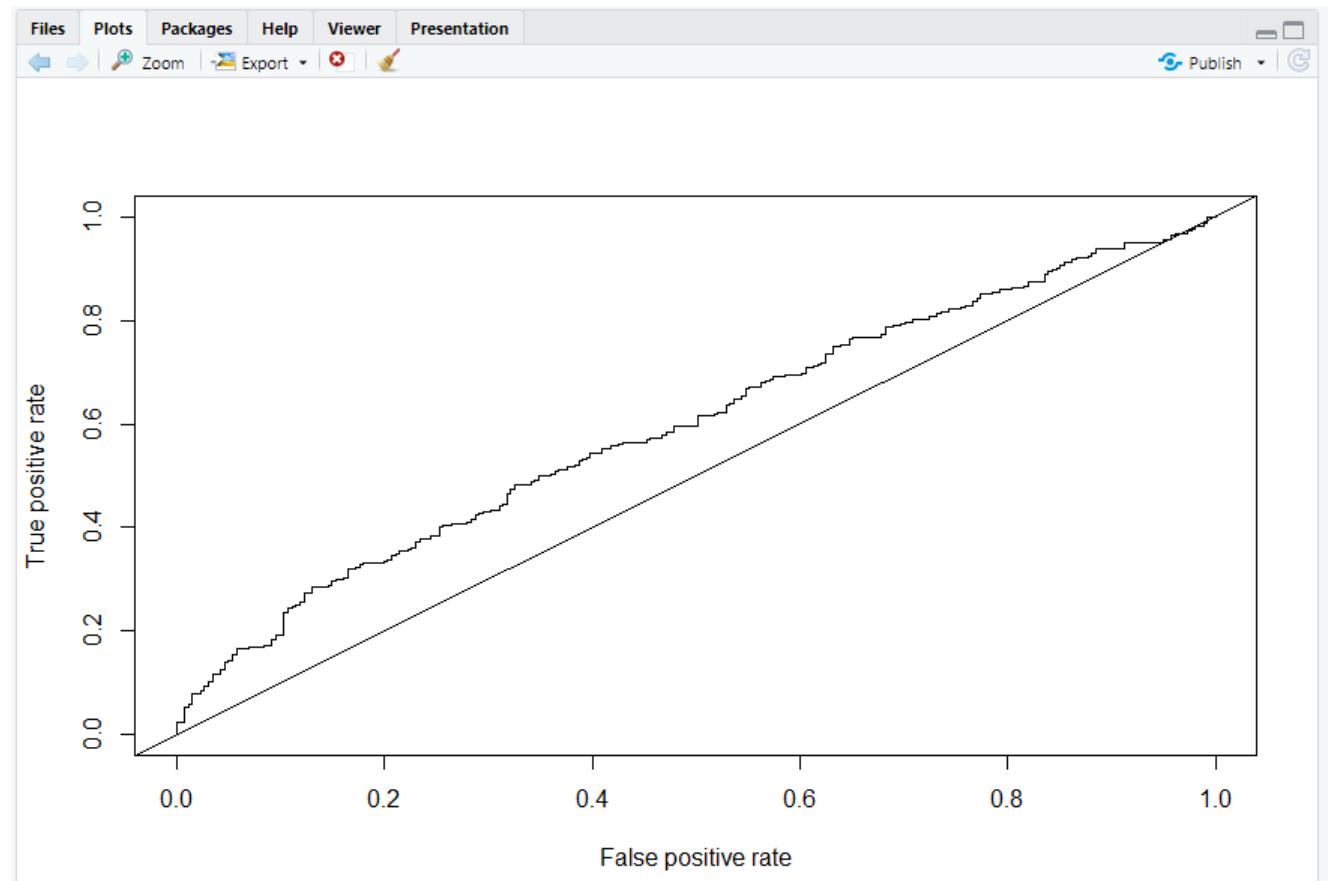
File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal x Background Jobs x

R 4.2.1 · ~/

```
> Roc_prediction=prediction(abalone_prob,test_abadata$Sex)
> Roc_Performance=performance(Roc_prediction,measure = "tpr",x.measure = "fpr")
> plot(Roc_Performance)
> abline(0,1)
> auc_performance=performance(Roc_prediction,measure = "auc")
> auc_performance@y.values
[[1]]
[1] 0.595955
> |
```



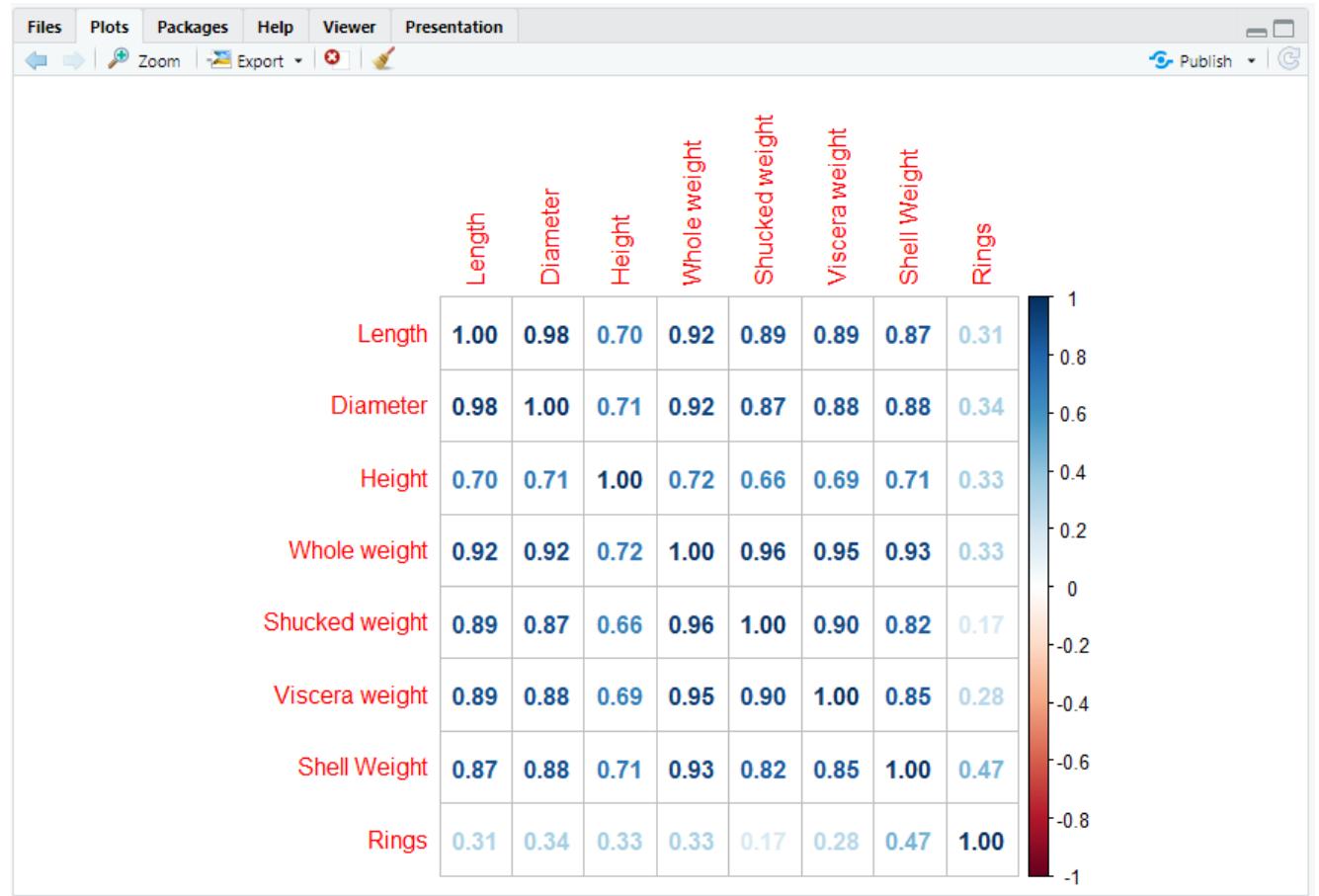
# Assignment-2

Shraddha

## Plotting Correlation between predictors

RStudio interface showing the R console output:

```
R 4.2.1 · ~/r
> cm=cor(abalone_dta2[,-1])
> corrplot(cm,method = "number")
> |
```



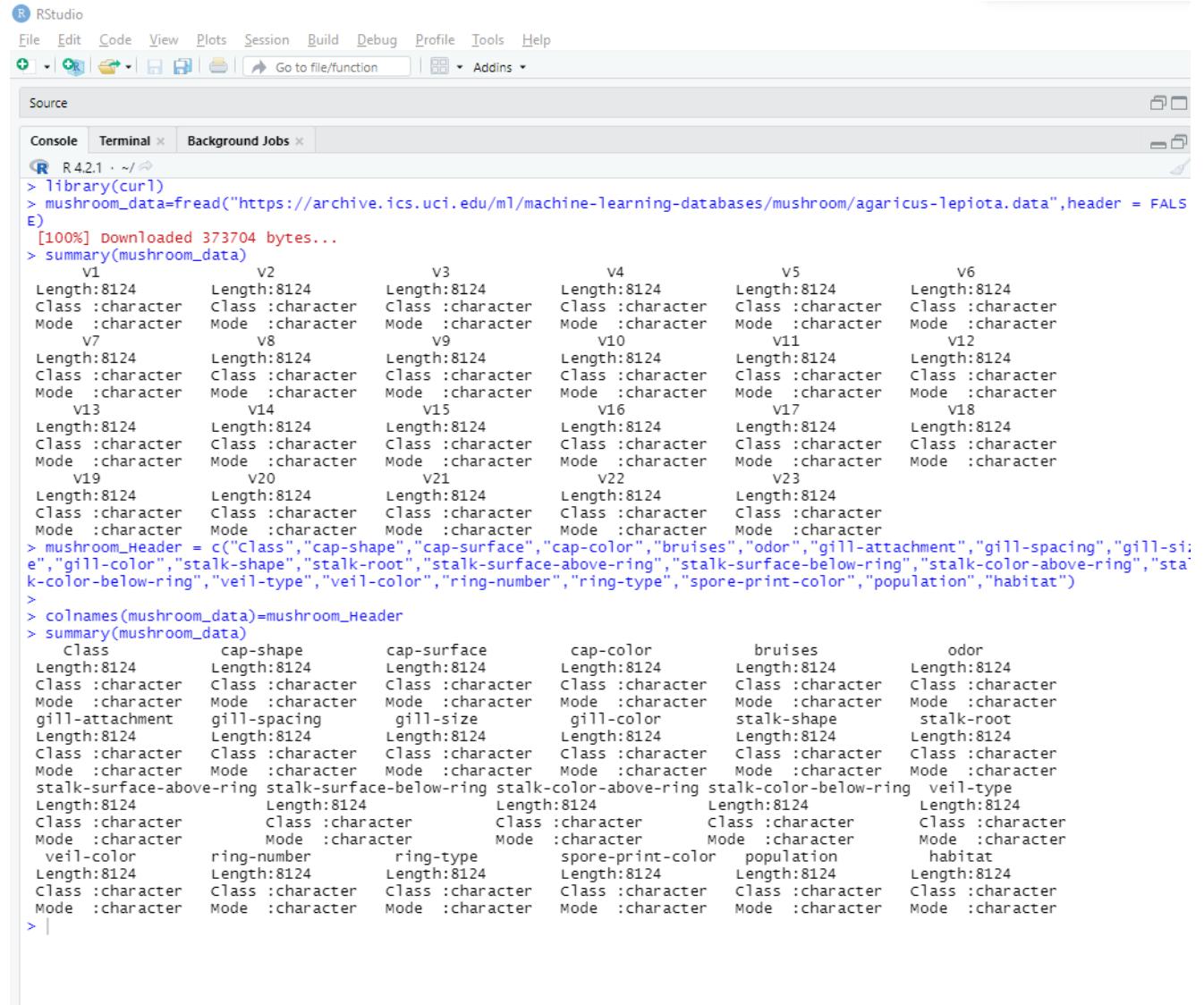
## Assignment-2

Shraddha

From the above result we can see that there is a positive linear relation for all factors except the rings predictor which has a weak uphill relationship, while others have a high uphill connection.

## Problem-2

Load the mushroom sample dataset from the UCI Machine Learning Repository using R Dataframe.



The screenshot shows the RStudio interface with the console tab selected. The console window displays the R code used to load the dataset and the resulting summary information for each variable.

```
R 4.2.1 · ~/ ↵
> library(curl)
> mushroom_data=fread("https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agricus-lepiota.data",header = FALSE)
[100%] Downloaded 373704 bytes...
> summary(mushroom_data)
   V1          V2          V3          V4          V5          V6
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   V7          V8          V9          V10         V11         V12
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   V13         V14         V15         V16         V17         V18
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   V19         V20         V21         V22         V23
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
> mushroom_header = c("Class","cap-shape","cap-surface","cap-color","bruises","odor","gill-attachment","gill-spacing","gill-size","gill-color","stalk-shape","stalk-root","stalk-surface-above-ring","stalk-surface-below-ring","stalk-color-above-ring","stalk-color-below-ring","veil-type","veil-color","ring-number","ring-type","spore-print-color","population","habitat")
>
> colnames(mushroom_data)=mushroom_header
> summary(mushroom_data)
   Class      cap-shape      cap-surface      cap-color      bruises      odor
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   gill-attachment      gill-spacing      gill-size      gill-color      stalk-shape      stalk-root
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   stalk-surface-above-ring      stalk-surface-below-ring      stalk-color-above-ring      stalk-color-below-ring      veil-type
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
   veil-color      ring-number      ring-type      spore-print-color      population      habitat
Length:8124    Length:8124    Length:8124    Length:8124    Length:8124    Length:8124
Class :character Class :character Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character Mode  :character
```

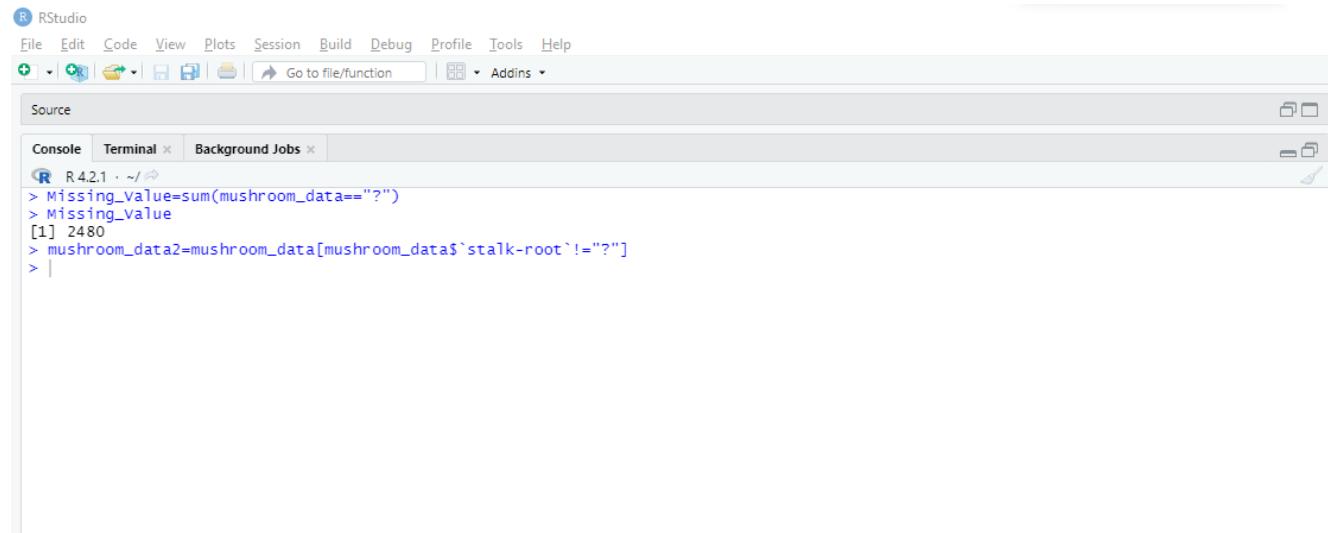
# Assignment-2

Shraddha

## Class Distribution and convert class attribute to a factor

```
> table(mushroom_data$Class)
e   p
4208 3916
> mushroom_data$Class=as.factor(mushroom_data$Class)
> dim(mushroom_data)
[1] 8124  23
> str(mushroom_data)
Classes 'data.table' and 'data.frame': 8124 obs. of  23 variables:
 $ Class      : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
 $ cap-shape   : chr  "x" "x" "b" "x" ...
 $ cap-surface  : chr  "s" "s" "s" "y" ...
 $ cap-color   : chr  "n" "y" "w" "w" ...
 $ bruises     : chr  "t" "t" "t" "t" ...
 $ odor        : chr  "p" "a" "l" "p" ...
 $ gill-attachment: chr  "f" "f" "f" "f" ...
 $ gill-spacing: chr  "c" "c" "c" "c" ...
 $ gill-size    : chr  "n" "b" "b" "n" ...
 $ gill-color   : chr  "k" "k" "n" "n" ...
 $ stalk-shape  : chr  "e" "e" "e" "e" ...
 $ stalk-root   : chr  "e" "c" "c" "e" ...
 $ stalk-surface-above-ring: chr  "s" "s" "s" "s" ...
 $ stalk-surface-below-ring: chr  "s" "s" "s" "s" ...
 $ stalk-color-above-ring: chr  "w" "w" "w" "w" ...
 $ stalk-color-below-ring: chr  "w" "w" "w" "w" ...
 $ veil-type    : chr  "p" "p" "p" "p" ...
 $ veil-color   : chr  "w" "w" "w" "w" ...
 $ ring-number  : chr  "o" "o" "o" "o" ...
 $ ring-type    : chr  "p" "p" "p" "p" ...
 $ spore-print-color: chr  "k" "n" "n" "k" ...
 $ population   : chr  "s" "n" "n" "s" ...
 $ habitat      : chr  "u" "g" "m" "u" ...
 - attr(*, ".internal.selfref")=<externalptr>
> |
```

## Number of Missing Values:



Removing the data with missing values do not impact on our dataset.

Create a Naive Bayes classifier using the e1071 package, using the sample function to split the data between 80% for training and 20% for testing.

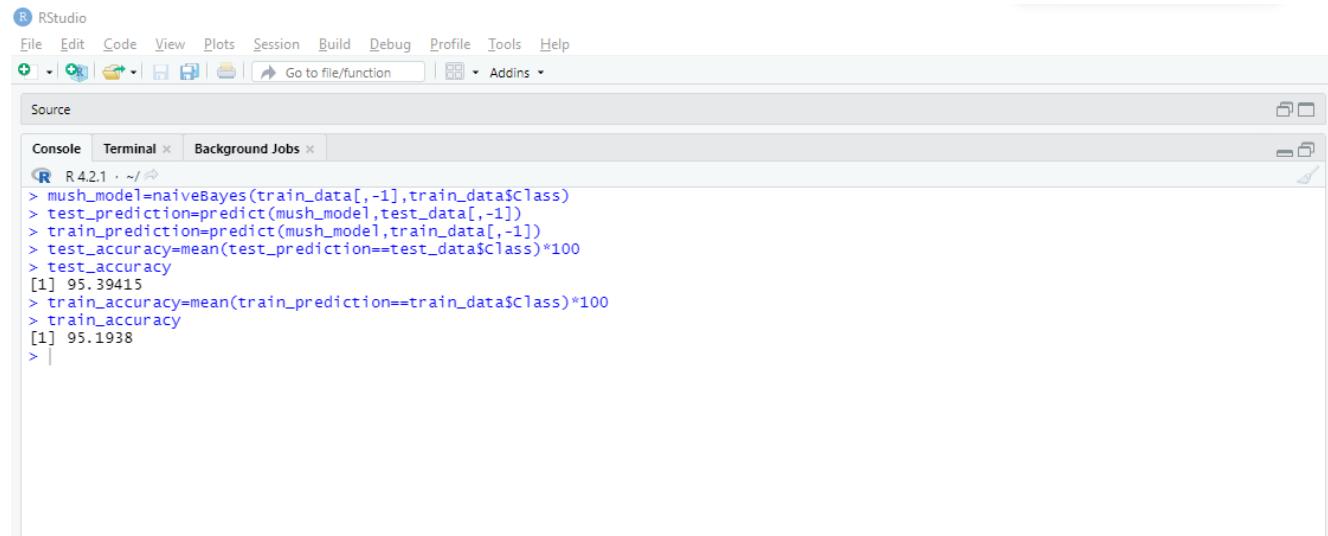
# Assignment-2

Shraddha



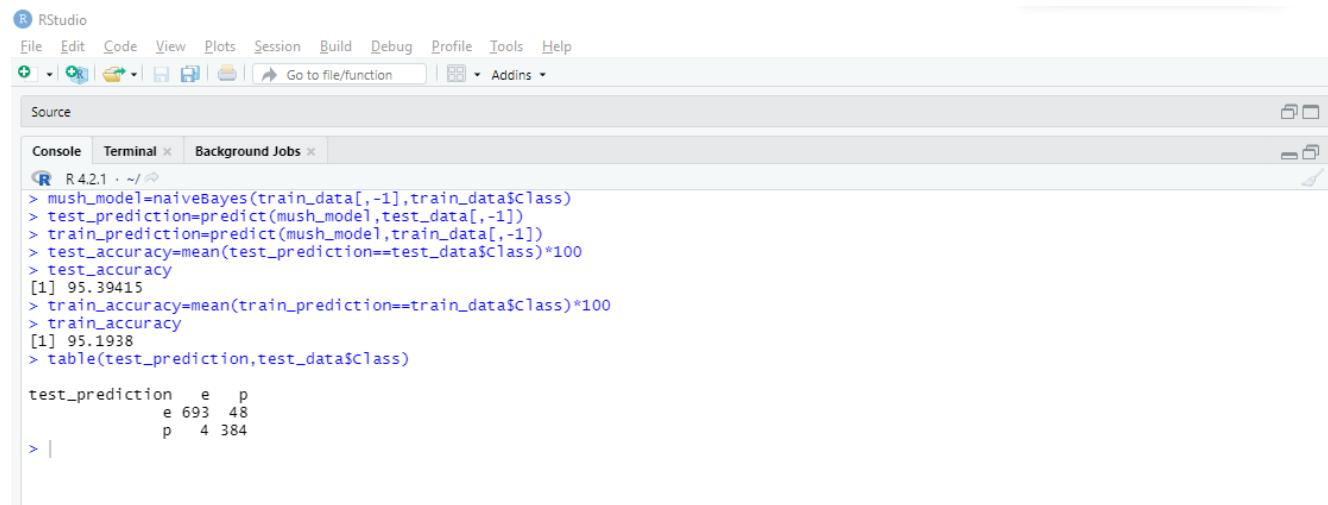
```
R 4.2.1 ~/RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ - Go to file/function | Addins
Source
Console Terminal Background Jobs
R 4.2.1 ~/RStudio
> library(e1071)
> train_size=floor(0.80*nrow(mushroom_data2))
> train_Index=sample(nrow(mushroom_data2),size = train_size)
> train_data=mushroom_data2[train_Index,]
> test_data=mushroom_data2[-train_Index,]
```

## Prediction of testing and training data



```
R 4.2.1 ~/RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ - Go to file/function | Addins
Source
Console Terminal Background Jobs
R 4.2.1 ~/RStudio
> mush_model=naiveBayes(train_data[,-1],train_data$class)
> test_prediction=predict(mush_model,test_data[,-1])
> train_prediction=predict(mush_model,train_data[,-1])
> test_accuracy=mean(test_prediction==test_data$class)*100
> test_accuracy
[1] 95.39415
> train_accuracy=mean(train_prediction==train_data$class)*100
> train_accuracy
[1] 95.1938
> |
```

Use the table function to create a confusion matrix of predicted vs. actual classes - how many false positives did the model produce?



```
R 4.2.1 ~/RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ - Go to file/function | Addins
Source
Console Terminal Background Jobs
R 4.2.1 ~/RStudio
> mush_model=naiveBayes(train_data[,-1],train_data$class)
> test_prediction=predict(mush_model,test_data[,-1])
> train_prediction=predict(mush_model,train_data[,-1])
> test_accuracy=mean(test_prediction==test_data$class)*100
> test_accuracy
[1] 95.39415
> train_accuracy=mean(train_prediction==train_data$class)*100
> train_accuracy
[1] 95.1938
> table(test_prediction,test_data$class)

test_prediction   e   p
                  e 693  48
                  p   4 384
> |
```

The result in above image shows TP = 693, FP = 48, FN = 4 and TN = 384

# Assignment-2

Shraddha

## Problem-3

Load the Yacht Hydrodynamics sample dataset from the UCI Machine Learning Repository into R using Dataframe.

RStudio interface showing the R console output:

```
R 4.2.1 · ~/ ↵
> library(readr)
> library(caret)
Loading required package: ggplot2
Use suppressPackageStartupMessages() to eliminate package startup messages
Loading required package: lattice
> library(data.table)
data.table 1.14.2 using 4 threads (see ?getDTthreads). Latest news: r-datarable.com
> hydro_data<-readr("https://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht_hydrodynamics.data",header = FALSE)
[100%] Downloaded 11487 bytes...
> summary(hydro_data)
   V1          V2          V3          V4          V5          V6          V7 
Min. :-5.000  Min. :0.5300  Min. :4.340  Min. :2.810  Min. :2.730  Min. :0.1250  Min. : 0.0100 
1st Qu.:-2.400 1st Qu.:0.5460 1st Qu.:4.770 1st Qu.:3.750 1st Qu.:3.150 1st Qu.:0.2000 1st Qu.: 0.7775 
Median : -2.300 Median :0.5650 Median :4.780 Median :3.955 Median :3.150 Median :0.2875 Median : 3.0650 
Mean   : -2.382 Mean   :0.5641 Mean   :4.789 Mean   :3.937 Mean   :3.207 Mean   :0.2875 Mean   :10.4954 
3rd Qu.:-2.300 3rd Qu.:0.5740 3rd Qu.:5.100 3rd Qu.:4.170 3rd Qu.:3.510 3rd Qu.:0.3750 3rd Qu.:12.8150 
Max.   : 0.000  Max.   :0.6000 Max.   :5.140 Max.   :5.350 Max.   :3.640 Max.   :0.4500 Max.   :62.4200 
> colnames(hydro_data)=c("longitudinalPos","prismaticcoef","LDR","BDR","LBR","froudeNo","Residuary")
>
> summary(hydro_data)
  longitudinalPos prismaticcoef        LDR          BDR          LBR          froudeNo        Residuary  
Min. :-5.000  Min. :0.5300  Min. :4.340  Min. :2.810  Min. :2.730  Min. :0.1250  Min. : 0.0100 
1st Qu.:-2.400 1st Qu.:0.5460 1st Qu.:4.770 1st Qu.:3.750 1st Qu.:3.150 1st Qu.:0.2000 1st Qu.: 0.7775 
Median : -2.300 Median :0.5650 Median :4.780 Median :3.955 Median :3.150 Median :0.2875 Median : 3.0650 
Mean   : -2.382 Mean   :0.5641 Mean   :4.789 Mean   :3.937 Mean   :3.207 Mean   :0.2875 Mean   :10.4954 
3rd Qu.:-2.300 3rd Qu.:0.5740 3rd Qu.:5.100 3rd Qu.:4.170 3rd Qu.:3.510 3rd Qu.:0.3750 3rd Qu.:12.8150 
Max.   : 0.000  Max.   :0.6000 Max.   :5.140 Max.   :5.350 Max.   :3.640 Max.   :0.4500 Max.   :62.4200 
> attach(hydro_data)
> |
```

Use the caret package to perform a 80/20 test-train split (via the `createDataPartition` function), and obtain a training fit for a linear model

RStudio interface showing the R console output:

```
R 4.2.1 · ~/ ↵
> train_Index=createDataPartition(y=hydro_data$Residuary,p=0.8,list = FALSE)
> training_data=hydro_data[train_Index,]
> testing_data=hydro_data[-train_Index,]
> LM1=lm(hydro_data$Residuary~hydro_data$longitudinalPos + hydro_data$prismaticcoef + hydro_data$LDR + hydro_data$BDR + hydro_data$LBR + hydro_data$froudeNo , data = training_data)
> |
```

What is the training MSE/RMSE and R2 results?

## Assignment-2

Shraddha

RStudio interface showing R code in the console:

```
R 4.2.1 · ~/d
> mse=function(ya,yp) { return(mean((ya-yp)^2)) }
> Training_mse1=MSE(hydro_data$Residuary,LM1$fitted.values)
> Training_mse1
[1] 78.45015
> Training_RMSE=sqrt(Training_mse1)
> Training_RMSE
[1] 8.857209
> Training_rsquared=summary(LM1)$r.sq
> Training_rsquared
[1] 0.6575638
>
```

Next, use the caret package to perform a bootstrap from the full sample dataset with N=1000 samples for fitting a linear model (via the train Control method), resulting in a training MSE/RMSE and R2 for each resample.

RStudio interface showing R code in the console:

```
R 4.2.1 · ~/d
> training_Control=trainControl(method = "boot",number = 1000)
> LM2=train(Residuary~. ,data = training_data , method = "lm" ,trcontrol = training_Control)
> summary(LM2$resample$RMSE)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
6.897 8.750 9.169 9.195 9.632 11.948
> summary(LM2$resample$Rsquared)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
0.5194 0.6139 0.6349 0.6332 0.6522 0.7086
>
```

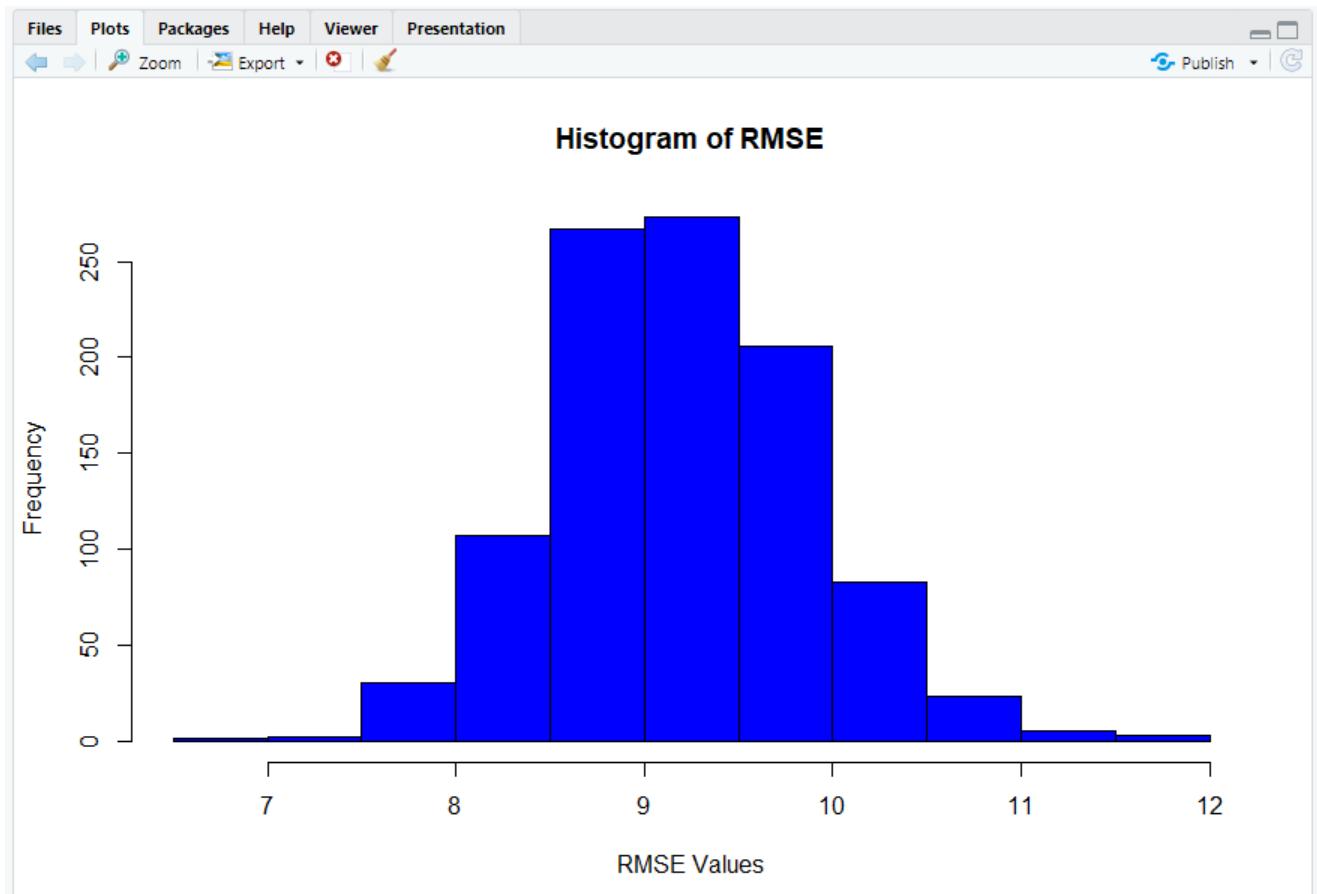
Plot a histogram of the RMSE values and provide a mean RMSE and R2 for the fit.

RStudio interface showing R code in the console:

```
R 4.2.1 · ~/d
> hist(LM2$resample$RMSE,xlab = "RMSE values" , main = "Histogram of RMSE" ,col = "blue")
>
```

## Assignment-2

Shraddha

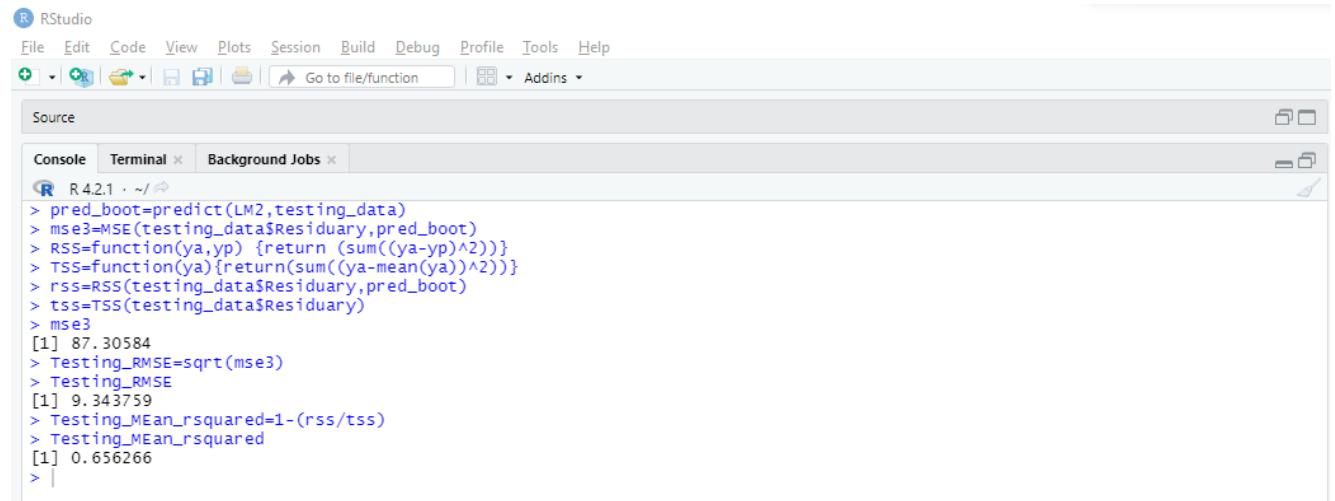


Screenshot of RStudio showing the R console output:

```
R 4.2.1  ~ / 
> Training_mse2=mean(LM2$resample$RMSE)^2
> Training_mse2
[1] 84.54655
> Training_Mean_RMSE=mean(LM2$resample$RMSE)
> Training_Mean_RMSE
[1] 9.19492
> Training_Mean_rsquared=mean(LM2$resample$Rsquared)
> Training_Mean_rsquared
[1] 0.6331882
>
```

## Assignment-2

Shraddha



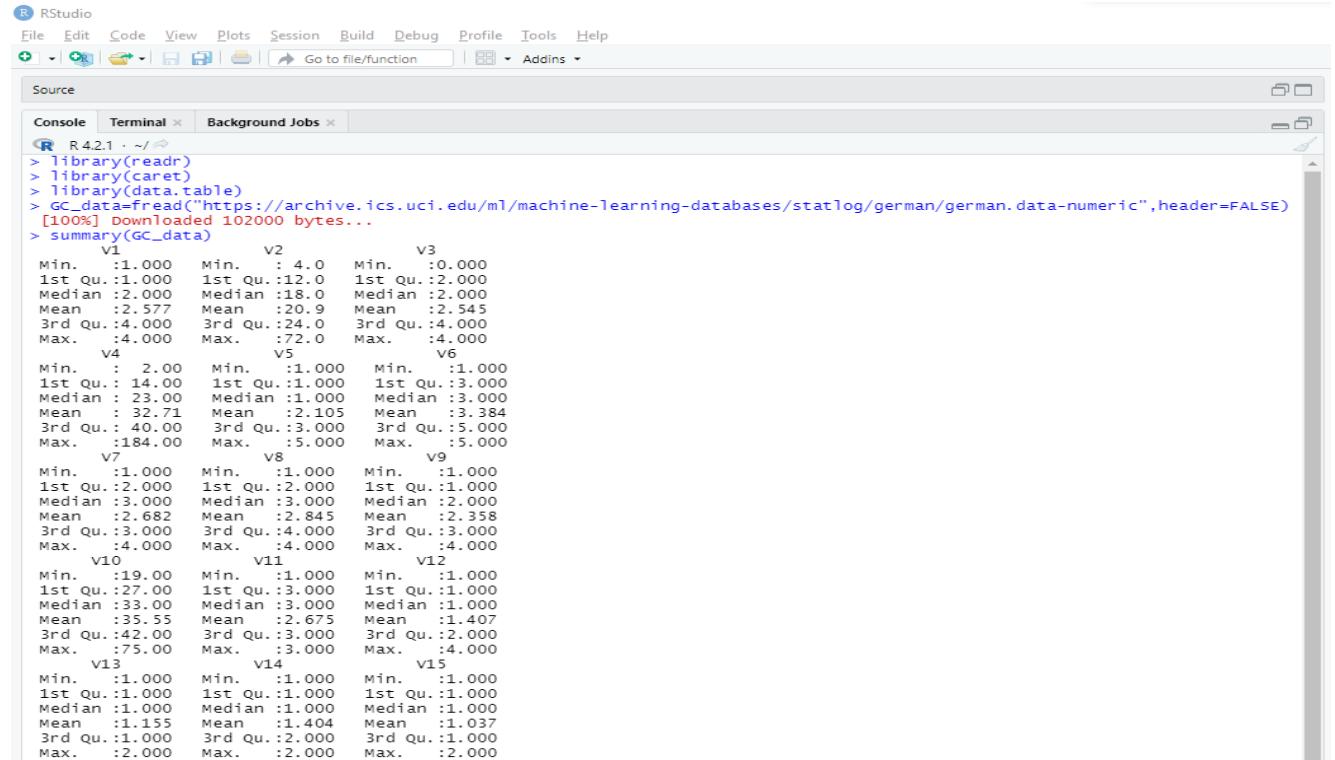
The screenshot shows the RStudio interface with the Source tab selected. The console output displays R code for calculating RMSE and R-squared values for a bootstrap model compared to a testing data set. The results show no significant difference between the two models.

```
R 4.2.1 · ~/d
> pred_boot=predict(LM2,testing_data)
> mse3=MSE(testing_data$Residuary,pred_boot)
> RSS=function(ya,yp) {return (sum((ya-yp)^2))}
> TSS=function(ya){return(sum((ya-mean(ya))^2))}
> rss=RSS(testing_data$Residuary,pred_boot)
> tss=TSS(testing_data$Residuary)
> mse3
[1] 87.30584
> Testing_RMSE=sqrt(mse3)
> Testing_RMSE
[1] 9.343759
> Testing_MEan_rsquared=1-(rss/tss)
> Testing_MEan_rsquared
[1] 0.656266
> |
```

The result of testing data set shows there is no difference between the actual and bootstrap models.

## Problem 4

Load the German Credit Data sample dataset from the UCI Machine Learning Repository (German. Data-numeric) into R using a dataframe.



The screenshot shows the RStudio interface with the Source tab selected. The console output displays R code for loading the German Credit Data dataset from a URL and summarizing its columns. The summary statistics for each column are provided.

```
R 4.2.1 · ~/d
> library(readr)
> library(caret)
> library(data.table)
> GC_data=readr::read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data-numeric",header=FALSE)
[100%] Downloaded 102000 bytes...
> summary(GC_data)
      V1          V2          V3
Min. :1.000  Min. : 4.0  Min. :0.000
1st Qu.:1.000  1st Qu.:12.0  1st Qu.:2.000
Median :2.000  Median :18.0  Median :2.000
Mean   :2.577  Mean   :20.9  Mean   :2.545
3rd Qu.:4.000  3rd Qu.:24.0  3rd Qu.:4.000
Max.   :4.000  Max.   :72.0  Max.   :4.000
      V4          V5
Min. : 2.00  Min. :1.000  Min. :1.000
1st Qu.:14.00 1st Qu.:1.000  1st Qu.:3.000
Median :23.00  Median :1.000  Median :3.000
Mean   :32.71  Mean   :2.105  Mean   :3.384
3rd Qu.:40.00  3rd Qu.:3.000  3rd Qu.:5.000
Max.   :184.00  Max.   :5.000  Max.   :5.000
      V6          V7          V8          V9
Min. :1.000  Min. :1.000  Min. :1.000  Min. :1.000
1st Qu.:2.000 1st Qu.:2.000  1st Qu.:1.000  1st Qu.:1.000
Median :3.000  Median :3.000  Median :2.000  Median :2.000
Mean   :2.682  Mean   :2.845  Mean   :2.358  Mean   :2.358
3rd Qu.:3.000  3rd Qu.:4.000  3rd Qu.:3.000  3rd Qu.:3.000
Max.   :4.000  Max.   :4.000  Max.   :4.000  Max.   :4.000
      V10         V11         V12
Min. :19.00  Min. :1.000  Min. :1.000
1st Qu.:27.00 1st Qu.:3.000  1st Qu.:1.000
Median :33.00  Median :3.000  Median :1.000
Mean   :35.55  Mean   :2.675  Mean   :1.407
3rd Qu.:42.00  3rd Qu.:3.000  3rd Qu.:2.000
Max.   :75.00  Max.   :3.000  Max.   :4.000
      V13         V14         V15
Min. :1.000  Min. :1.000  Min. :1.000
1st Qu.:1.000 1st Qu.:1.000  1st Qu.:1.000
Median :1.000  Median :1.000  Median :1.000
Mean   :1.155  Mean   :1.404  Mean   :1.037
3rd Qu.:1.000  3rd Qu.:2.000  3rd Qu.:1.000
Max.   :2.000  Max.   :2.000  Max.   :2.000
```

## Assignment-2

Shraddha

Perform a 80/20 test-train split (via the `createDataPartition` function), and obtain a training fit for a logistic model via the `glm` package.

```
R R4.2.1 · ~/d
> GC_data$V25=factor(GC_data$V25)
> train_Index=createDataPartition(y=GC_data$V25,p=0.8,list = FALSE)
> training_data=GC_data[train_Index,]
> testing_data=GC_data[-train_Index,]
> logistic_model1=glm(v25~.,family = binomial,data=training_data)
```

```
R R4.2.1 · ~/d
> logistic_model1=glm(v25~.,family = binomial,data = training_data)
> Actual_value=training_data$V25
> fitted_values=ifelse(logistic_model1$fitted.values>0.5,2,1)
> fitted_values=factor(fitted_values)
> confu_Matrix=confusionMatrix(fitted_values,training_data$V25)
> confu_Matrix
Confusion Matrix and statistics

Reference
Prediction   1   2
      1 507 117
      2  53 123

    Accuracy : 0.7875
    95% CI : (0.7575, 0.8154)
    No Information Rate : 0.7
    P-value [Acc > NIR] : 1.558e-08

    Kappa : 0.4523

McNemar's Test P-Value : 1.353e-06

    Sensitivity : 0.9054
    Specificity : 0.5125
    Pos Pred Value : 0.8125
    Neg Pred Value : 0.6989
    Prevalence : 0.7000
    Detection Rate : 0.6338
    Detection Prevalence : 0.7800
    Balanced Accuracy : 0.7089

'Positive' Class : 1
```

What is the training Precision/Recall and F1 results?

# Assignment-2

Shraddha

```
R 4.2.1 · ~/ ◁
> Training_Precision=(confu_Matrix$byClass[5]*100)
> Training_Precision
Precision
  81.25
> Training_recall=confu_Matrix$byClass[6]*100
> Training_recall
Recall
90.53571
> Training_F1_score=confu_Matrix$byClass[7]*100
> Training_F1_score
      F1
85.64189
> |
```

```
R 4.2.1 · ~/ ◁
> probability=predict(logistic_model1,testing_data,type="response")
> Test_fitted_values=ifelse(probability>0.5,2,1)
> Test_fitted_values=factor(Test_fitted_values)
> Test_Conf_Matrix=confusionMatrix(Test_fitted_values,testing_data$v25)
> Test_Conf_Matrix
Confusion Matrix and Statistics

             Reference
Prediction    1     2
      1 118   21
      2   22   39

          Accuracy : 0.785
          95% CI : (0.7215, 0.8398)
          No Information Rate : 0.7
          P-Value [Acc > NIR] : 0.004475

          Kappa : 0.4905

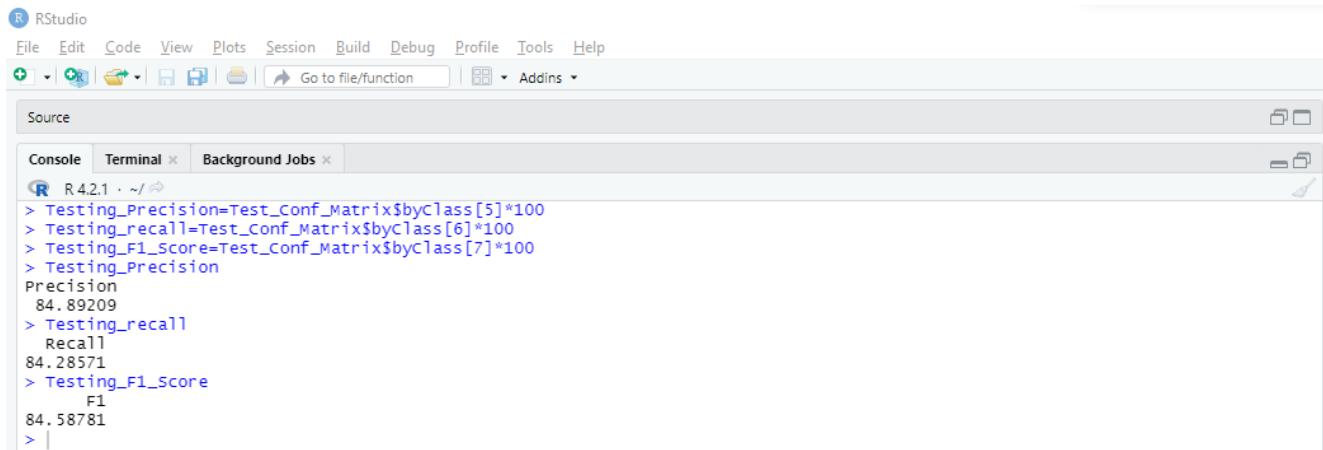
McNemar's Test P-Value : 1.000000

          Sensitivity : 0.8429
          Specificity : 0.6500
          Pos Pred Value : 0.8489
          Neg Pred Value : 0.6393
          Prevalence : 0.7000
          Detection Rate : 0.5900
          Detection Prevalence : 0.6950
          Balanced Accuracy : 0.7464

          'Positive' Class : 1
> |
```

## Assignment-2

Shraddha



RStudio interface showing R code in the console tab:

```
R 4.2.1 · ~/ ◁
> Testing_Precision=Test_Conf_Matrix$byClass[5]*100
> Testing_recall=Test_Conf_Matrix$byClass[6]*100
> Testing_F1_Score=Test_Conf_Matrix$byClass[7]*100
> Testing_Precision
Precision
84.89209
> Testing_recall
Recall
84.28571
> Testing_F1_Score
F1
84.58781
> |
```

use the **trainControl** and **train** functions to perform a k=10 fold cross-validation fit of the same model, and obtain cross-validated training Precision/Recall and F1 values



RStudio interface showing R code in the console tab:

```
R 4.2.1 · ~/ ◁
> training_control=trainControl(method = "cv",number = 10)
> logistic_model2=train(v25~.,data = training_data,method="glm",family="binomial",trControl=training_control)
```

# Assignment-2

Shraddha

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal x Background Jobs x

R 4.2.1 · ~/

```
> fitted_values_cv=ifelse(logistic_model2$finalModel$fitted.values>0.5,2,1)
> fitted_values_cv=factor(fitted_values_cv)
> confu_Matrix_cv=confusionMatrix(fitted_values_cv,training_data$v25)
> confu_Matrix_cv
Confusion Matrix and statistics

      Reference
Prediction   1   2
      1 507 117
      2  53 123

      Accuracy : 0.7875
      95% CI : (0.7575, 0.8154)
      No Information Rate : 0.7
      P-Value [Acc > NIR] : 1.558e-08

      Kappa : 0.4523

McNemar's Test P-Value : 1.353e-06

      Sensitivity : 0.9054
      Specificity : 0.5125
      Pos Pred Value : 0.8125
      Neg Pred Value : 0.6989
      Prevalence : 0.7000
      Detection Rate : 0.6338
      Detection Prevalence : 0.7809
      Balanced Accuracy : 0.7089

      'Positive' Class : 1

> |
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Source

Console Terminal x Background Jobs x

R 4.2.1 · ~/

```
> Training_Precision_cv=confu_Matrix_cv$byClass[5]*100
> Training_recall_cv=confu_Matrix_cv$byClass[6]*100
> Training_F1_score_cv=confu_Matrix_cv$byClass[7]*100
> Training_Precision_cv
Precision
81.25
> Training_recall_cv
Recall
90.53571
> Training_F1_score_cv
F1
85.64189
> |
```

## Assignment-2

Shraddha

RStudio interface showing a console output for a logistic regression model evaluation. The code uses the `logistic\_model2` for prediction and `testing\_data` for testing. It calculates various performance metrics including Accuracy, 95% CI, No Information Rate, P-Value [Acc > NIR], Kappa, McNemar's Test P-Value, Sensitivity, Specificity, Pos Pred Value, Neg Pred Value, Prevalence, Detection Rate, Detection Prevalence, and Balanced Accuracy. The 'Positive' class is identified as 1.

```
R 4.2.1 · ~/◊
> probability_cv=predict(logistic_model2,testing_data,type = "prob")
> Test_fitted_values_cv=ifelse(probability>0.5,2,1)
> Test_fitted_values_cv=factor(Test_fitted_values)
> Test_cv_confu_matrix=confusionMatrix(Test_fitted_values,testing_data$v25)
> Test_cv_confu_matrix
Confusion Matrix and Statistics

Reference
Prediction 1 2
      1 118 21
      2  22 39

Accuracy : 0.785
95% CI : (0.7215, 0.8398)
No Information Rate : 0.7
P-Value [Acc > NIR] : 0.004475

Kappa : 0.4905

McNemar's Test P-Value : 1.000000

Sensitivity : 0.8429
Specificity : 0.6500
Pos Pred Value : 0.8489
Neg Pred Value : 0.6393
Prevalence : 0.7000
Detection Rate : 0.5900
Detection Prevalence : 0.6950
Balanced Accuracy : 0.7464

'Positive' Class : 1
```

RStudio interface showing a console output for calculating precision, recall, and F1 score from a confusion matrix. The code uses the `Test\_cv\_confu\_matrix` object, specifically its `byClass` component, scaled by 100. The calculated values are Precision: 84.89209, Recall: 84.28571, and F1: 84.58781.

```
R 4.2.1 · ~/◊
> Testing_Precision_CM_CV=Test_cv_confu_matrix$byClass[5]*100
> Testing_Recall_CM_CV=Test_cv_confu_matrix$byClass[6]*100
> Testing_F1_Score_CM_CV=Test_cv_confu_matrix$byClass[7]*100
> Testing_Precision_CM_CV
Precision
84.89209
> Testing_Recall_CM_CV
Recall
84.28571
> Testing_F1_Score_CM_CV
F1
84.58781
> |
```

By looking at testing data we can say that there is no difference between original and bootstrap models.