CSP 571-Data Preparation and Analysis

Assignment-2

1. Recitation Problems

Chapter-4

Recitation Exercises

Chapter-4

Exercise-4

a) For every observation we use the average value that accounts for 10% of the observations so for test observation $x \in [0.05, 0.95]$ and the observations are in $[x-0.05)$, $[x+0.05]$ intervals respectively. It's given that the observations are uniformly distributed. So for $[x < 0.05]$, the observations in the range $[x-0.05, 1]$ are used. Hence, in order to know the average fraction we use:-

$$\int_0^{0.05} (100x+5)\,dx + \int_{0.95}^{1} (105-100x)\,dx + \int_{0.05}^{0.95} 10x$$

$$0.375 + 0.375 + 9 = 9.75$$

b) Since we are assuming that x_1, x_2 are uniformly distributed on $[0,1] \times [0,1]$, the fractions of observations used to make a prediction is $9.75 \times 9.75 = 0.950625$

c) In (a) & (b) the observations were uniformly distributed. So, similarly in (c) also the observations are uniformly distributed. It can be concluded that the fraction of available observations used to make prediction is $9.75\%^{100}$ which is nearly equivalent to 0, i.e $9.75\%^{100} \simeq 0\%$

d) As in parts a - c the fraction of observations is within 10% range of the test observation value for the predictor. Hence, the fraction of the available observation we use to make prediction is $(9.75\%)^p$. Hence, $p \to \infty \lim (9.75)^p = 0$

i.e.

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

e) For $p=1$ we have $s=0.1$ —(i)
for $p=2$ we have $s=(0.1)^{1/2}$ —(ii)
for $p=100$ we have $s=(0.1)^{1/100}$ —(iii)
This is because, the volume is given by the formula $= V = s^p$ where s = side length of a p dimensional hypercube
$s = V^{1/p}$.
From (i) (ii) & (iii) we can see that the K-nearest neighbours for a given test observation will likely to have predictor values different from predictor values for test observation.

Exercise-6

X_1 = hours studied
X_2 = undergrad GPA
Y = Receive an A
$\hat{\beta}_0 = -6 \quad \hat{\beta}_1 = 0.05, \quad \hat{\beta}_2 = 1$

(a) Logistic Regression with multiple variables

$$P(xy) = \frac{e^{\beta_0 + x_1\beta_1 + x_2\beta_2}}{1 + e^{\beta_0 + x_1\beta_1 + x_2\beta_2}}$$

$x_1 = 40$ hrs $\quad X_2 =$ GPA
$x_2 = 3.5$ GPA $\quad , \quad e = 2.71828$

$$\Rightarrow P(\text{Receive an A} | x) = \frac{e^{\beta_0 + x_1\beta_1 + x_2\beta_2}}{1 + e^{\beta_0 + x_1\beta_1 + x_2\beta_2}} = \frac{e^{-6 + 0.05x_1 + x_2}}{1 + e^{-6 + 0.05x_1 + x_2}}$$

$$\Rightarrow \frac{2.72^{-6 + (40 \times 0.05) + (3.5 \times 1)}}{1 + 2.72^{-6 + (40 \times 0.05) + (3.5 \times 1)}} = 0.3774 = 0.3774$$

Probability of receiving A in Class = $0.3774 = 37.74\%$

b) $P(xy) = 50\% = 0.5 \quad x_2 = 3.5$ GPA $\quad x_1 = ?$

$$0.5 = \frac{2.72^{-6 + (x_1 \times 0.05) + (3.5 \times 1)}}{1 + 2.72^{-6 + (x_1 \times 0.05) + (3.5 \times 1)}} = 50 \text{ hrs}$$

Exercise 7

Exercise 7

$\sigma^2 = 36 \Rightarrow \sigma = 6$

dividends issued by companies = 80%

" not issued by companies = 20%

$\bar{x} = 10 = \mu_{yes}$

$\bar{x} = 0$ (for companies that did not issue dividend $= \mu_{no}$

Percentage profit $= x = 4 \equiv (X = 4)$

$$P_K(x) = (y = k \mid X = x) = \frac{\pi_K \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left[\frac{(x-\mu_K)}{\sigma}\right]^2}}{\sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2}}$$

$$P_{yes}(4) = Pr(Y = yes \mid x = 4)$$

$$\Rightarrow \frac{\pi_{yes} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left[\frac{(4-\mu_{yes})}{\sigma}\right]^2}}{\pi_{yes} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left[\frac{(x-\mu_{yes})}{\sigma}\right]^2} + \pi_{no} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_{no}}{\sigma}\right)^2}}$$

$$\Rightarrow \frac{0.8 \times \frac{1}{\sqrt{2\pi} \times 6} e^{-\frac{1}{2}\left(\frac{4-10}{6}\right)^2}}{\left(0.8 \times \frac{1}{\sqrt{2\pi} \times 6} e^{-\frac{1}{2}\left[\frac{4-10}{6}\right]^2}\right) + \left(0.2 \times \frac{1}{\sqrt{2\pi} \times 6} e^{-\frac{1}{2}\left[\frac{4-0}{6}\right]^2}\right)}$$

$0.7518 = 0.752 = 75.2\%$

$P(\text{yes})(4) = 75.2\%$

Exercise -9

a) People with odds = 0.37

Odds of event $= \frac{p(x)}{1-p(x)}$

$\frac{p(x)}{1-p(x)} = 0.37 = p(x) = 0.37(1-p(x))$

$p(x) = 0.37 - 0.37p(x)$

$p(x) + 0.37p(x) = 0.37 = 1.37p(x) = 0.37$

$p(x) = \frac{0.37}{1.37}$

$= 0.270 = 27\%$

b) $p(x) = 16\% = 0.16$

Odds of event (default) $= \frac{0.16}{1-0.16} = \frac{0.16}{0.84} =$

$\frac{16}{84} \; 4$

21

$= \frac{4}{21} =$ 4 people who defaulted &
21 people who did not.

2. Recitation Problems

Chapter-5

Exercise 2

3. **(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.**

The probability that the first observation is the jth observation is 1/n. Hence, the probability that the first bootstrap observation is not the jth observation is 1-1/n.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

The probability that the second bootstrap is the jth observation is 1/n. Hence, the probability that the second bootstrap observation is not the jth observation is 1-1/n.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1 – 1/n)n

The probability of selecting an observation is 1/n and the probability of not selecting an observation is 1-1/n. So, the probability that the jth observation is not in the bootstrap sample is simply the product of the probabilities that the given observation is not the jth observation. We consider the product of the probabilities since they (probabilities) are independent of each other i.e. 1-1/n* 1-1/n* 1-1/n and so on up to n times $=(1-1/n)^n$.

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample.

The probability that the jth observation is in the bootstrap sample is 1-((1 – 1/n)^n). Thus, if n =5 we have 1-((1 – 1/5)^5) = 0.67232 .

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

The probability that the jth observation is in the bootstrap sample is 1-((1 – 1/n)^n). Thus, if n =100 we have 1-((1 – 1/100)^100) = 0.63396 .

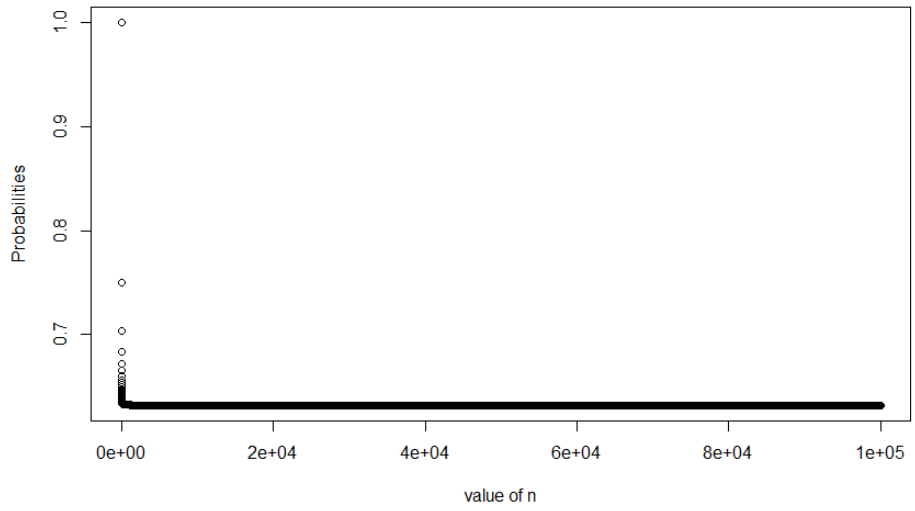
(f) When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?

The probability that the jth observation is in the bootstrap sample is 1-((1 – 1/n)^n). Thus, if n =10000 we have 1-((1 – 1/10000) ^10000).

(g) Create a plot that displays, for each integer value of n from 1 to 100, 000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
plot(1:100000, 1-(1-1/1:100000)^(1:100000), xlab = "value of n", ylab = "Probabilities")
```

```
> plot(1:100000, 1-(1-1/1:100000)^(1:100000), xlab = "value of n", ylab = "Probabilities")
>
```



Answer- It is observed that the probability is 0.632 and after that, it remains constant as n →∞.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

```
store <- rep(NA, 10000)
for (i in 1:10000) {
  store[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
}
mean(store)
```

```
> store <- rep(NA, 10000)
> for (i in 1:10000) {
+   store[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
+ }
> mean(store)
[1] 0.6303
>
```

As we can see the result of mean is 0.6303. We can say that lim (1 + $x\ n$)^n = e ^x, thus, if we use this for our
$n \to \infty$
Problem we get that the probability that the bootstrap of size n containing jth observation will be 0.632 as n→∞.

Exercise 3

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

In k-fold cross validation, the set of observations is divided randomly into k groups of equal sizes or folds. The first fold is treated as a validation set and the rest of the folds are fitted according to k-1. The MSE (Mean Squared Error) is then computed using the observations in the held-out fold. This procedure is repeated k times and each time a different set of observations is treated as a validation set. The process results in k estimates of test error i.e., MSE_1, MSE_2, MSE_3, and so on. The k-fold CV estimate is given by computing the averages of values.

$CV_k = 1/k\sum^k MSEi$
i=1

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

i. The validation set approach?

ii. LOOCV?

(i)Validation Set Approach- It has 2 drawbacks compared to k-fold cross-validation. First, the validation estimate of the test error rate can be variable i.e., depending on which observations are included in the training set and which are included in the validation set. Second, only a subset of observations is used to fit the model.

The advantage of the validation set approach is the computation cost since the data needs to be trained and tested only once unlike the k-fold CV where it needs to be run k times.

(ii)LOOCV- This approach has a larger computational cost than k fold CV model since the model is trained and tested n times instead of k. The LOOCV model has higher variance because they are correlated. Performing a k-fold CV for example say on K=5 will lead to an intermediate level of bias as each training set contains (k-1) n/k. Therefore, from the view of bias reduction, LOOCV can be preferred over k-fold CV.

2. Practicum Problems

Problem 1

```
library(ggplot2)

library(corrplot)
```

```
Loading required package: ROCR
> library(ggplot2)
> library(corrplot)
corrplot 0.92 loaded
>
```

```
AbaloneData1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data",

sep = ",", header = FALSE, col.names = c('Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',

'Viscera weight', 'Shell weight', 'Rings'), as.is = 2&6, stringsAsFactors=T)

colnames(AbaloneData1)


summary(AbaloneData1)
```

```
> AbaloneData1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data",
+ sep = ",", header = FALSE, col.names = c('Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
+ 'Viscera weight', 'Shell weight', 'Rings'), as.is = 2&6, stringsAsFactors=T)
>
> colnames(AbaloneData1)
[1] "Sex"            "Length"         "Diameter"       "Height"         "Whole.weight"   "Shucked.weight"
[7] "Viscera.weight" "Shell.weight"   "Rings"
>
```

```
> summary(AbaloneData1)
     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
> |
```

Removing all observations in the infant category

AbaloneData2 = AbaloneData1[AbaloneData1$Sex != "I",]

View(AbaloneData2)

```
 Max.    :1.0050   Max.    :29.000
> #Removing all observations in the infant category
> AbaloneData2 = AbaloneData1[AbaloneData1$Sex != "I", ]
> View(AbaloneData2)
> |
```

	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
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	0.8945	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
18	F	0.440	0.340	0.100	0.4510	0.1880	0.0870	0.1300	10
19	M	0.365	0.295	0.080	0.2555	0.0970	0.0430	0.1000	7
20	M	0.450	0.320	0.100	0.3810	0.1705	0.0750	0.1150	9
21	M	0.355	0.280	0.095	0.2455	0.0955	0.0620	0.0750	11
23	F	0.565	0.440	0.155	0.9395	0.4275	0.2140	0.2700	12

270	F	0.450	0.360	0.125	0.4995	0.2035	0.1000	0.1700	13
271	F	0.640	0.525	0.215	1.7790	0.4535	0.2655	0.5500	22
272	M	0.590	0.500	0.200	1.1870	0.4120	0.2705	0.3700	16
273	M	0.620	0.485	0.205	1.2190	0.3875	0.2505	0.3850	14
274	M	0.630	0.505	0.225	1.5250	0.5600	0.3335	0.4500	15
275	M	0.630	0.515	0.155	1.2590	0.4105	0.1970	0.4100	13
276	M	0.655	0.540	0.215	1.8440	0.7425	0.3270	0.5850	22
277	F	0.660	0.530	0.185	1.3485	0.4930	0.2450	0.4900	12
278	M	0.610	0.500	0.240	1.6420	0.5320	0.3345	0.6900	18
279	M	0.635	0.525	0.205	1.4840	0.5500	0.3115	0.4300	20
280	F	0.515	0.425	0.135	0.7120	0.2665	0.1605	0.2500	11
281	F	0.535	0.415	0.185	0.8415	0.3140	0.1585	0.3000	15
283	F	0.455	0.355	0.120	0.4495	0.1770	0.1040	0.1500	9
284	M	0.485	0.395	0.140	0.6295	0.2285	0.1270	0.2250	14
285	M	0.515	0.380	0.175	0.9565	0.3250	0.1580	0.3100	14
286	F	0.535	0.415	0.170	0.8790	0.2950	0.1965	0.2850	10
287	M	0.530	0.435	0.155	0.6990	0.2680	0.1595	0.2050	10
288	F	0.495	0.400	0.155	0.6445	0.2420	0.1325	0.2050	17

library(caret)

creating 80-20 test train split with createDataPartition

AbTrainData <- createDataPartition(y = AbaloneData2$Sex, p = 0.8, list = FALSE)

TrainingData = AbaloneData2[AbTrainData,]

TestingData = AbaloneData2[-AbTrainData,]

```
> library(caret)
> AbTrainData <- createDataPartition(y = AbaloneData2$Sex, p = 0.8, list = FALSE)
> TrainingData = AbaloneData2[AbTrainData,]
> TestingData = AbaloneData2[-AbTrainData,]
> |
```

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

AbaloneData2$Sex = factor(AbaloneData2$Sex)

FittedModel <- glm(Sex~.,family=binomial,data=TrainingData)

summary(FittedModel)

```
> AbaloneData2$Sex = factor(AbaloneData2$Sex)
> FittedModel <- glm(Sex~.,family=binomial,data=TrainingData)
> summary(FittedModel)
```

```
> summary(FittedModel)

Call:
glm(formula = Sex ~ ., family = binomial, data = TrainingData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7852  -1.1968   0.8725   1.1199   1.4229

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     2.886355   0.505414   5.711 1.12e-08 ***
Length         -1.875850   2.282582  -0.822   0.4112
Diameter       -5.426780   2.720960  -1.994   0.0461 *
Height         -2.553556   1.925024  -1.327   0.1847
Whole.weight    0.772836   0.853476   0.906   0.3652
Shucked.weight  2.129342   1.005438   2.118   0.0342 *
Viscera.weight -2.422110   1.448527  -1.672   0.0945 .
Shell.weight   -0.275745   1.268362  -0.217   0.8279
Rings           0.005275   0.017836   0.296   0.7674
---
Signif. codes:  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: 3076.6  on 2260  degrees of freedom
AIC: 3094.6

Number of Fisher Scoring iterations: 4

> |
```

Do the confidence intervals for the predictors contain 0 within the range? How does this relate to the null hypothesis?

confint(FittedModel)

```
> confint(FittedModel)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)      1.91063887  3.89329706
Length          -6.35205948  2.60203454
Diameter       -10.77944727 -0.10456961
Height          -6.96674598  0.75855741
Whole.weight    -0.89848720  2.45829289
Shucked.weight   0.15726368  4.10694831
Viscera.weight  -5.27277254  0.41163441
Shell.weight    -2.77338807  2.21087863
Rings           -0.02967909  0.04029393
>
```

Answer- From the above observations we can see that Shucked weight does not contain 0 within the range of confidence intervals but all other predictors have it in their confidence interval ranges. The null hypothesis assumes no relationship between X and Y; thus, we can conclude that the Null hypothesis holds valid for all predictors except Shucked weight. Thus, Shucked weight is a relevant predictor.

Use the confusionMatrix function in caret to observe testing results (use a 50% cutoff to tag Male/Female)

#Taking type of parameter as response, the output is of the form P[y=1/x]

PredictData <-predict(FittedModel, newdata = TestingData, type = "response")

use a 50% cutoff to tag Male/Female

Result <-ifelse(PredictData > 0.5, "M", "F")

Result <-factor(Result1)

confusionMatrix(Result, TestingData$Sex)

```
> PredictData <-predict(FittedModel, newdata = TestingData, type = "response")
> Result <-ifelse(PredictData > 0.5, "M", "F")
> Result <-factor(Result1)
> confusionMatrix(Result, TestingData$Sex)
```

```
> confusionMatrix(Result, TestingData$Sex)
Confusion Matrix and Statistics

          Reference
Prediction   F   M
         F  93  87
         M 168 218

               Accuracy : 0.5495
                 95% CI : (0.5074, 0.591)
    No Information Rate : 0.5389
    P-Value [Acc > NIR] : 0.3217

                  Kappa : 0.0727

 Mcnemar's Test P-Value : 5.449e-07
```

```
                 Kappa : 0.0727

Mcnemar's Test P-Value : 5.449e-07

            Sensitivity : 0.3563
            Specificity : 0.7148
         Pos Pred Value : 0.5167
         Neg Pred Value : 0.5648
             Prevalence : 0.4611
         Detection Rate : 0.1643
   Detection Prevalence : 0.3180
      Balanced Accuracy : 0.5355

       'Positive' Class : F
```

Accuracy is 0.5495

how does the accuracy compare to a random classifier ROC curve?

#classifier ROC curve

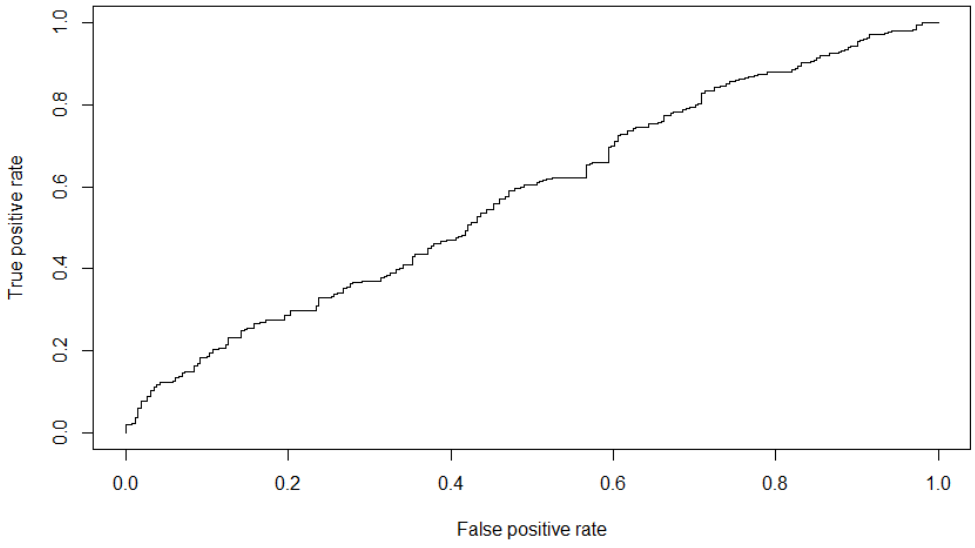
library(ROCR)

pred = prediction(PredictData,TestingData$Sex)

perf = performance(pred, measure = "tpr", x.measure = "fpr")

plot(perf)

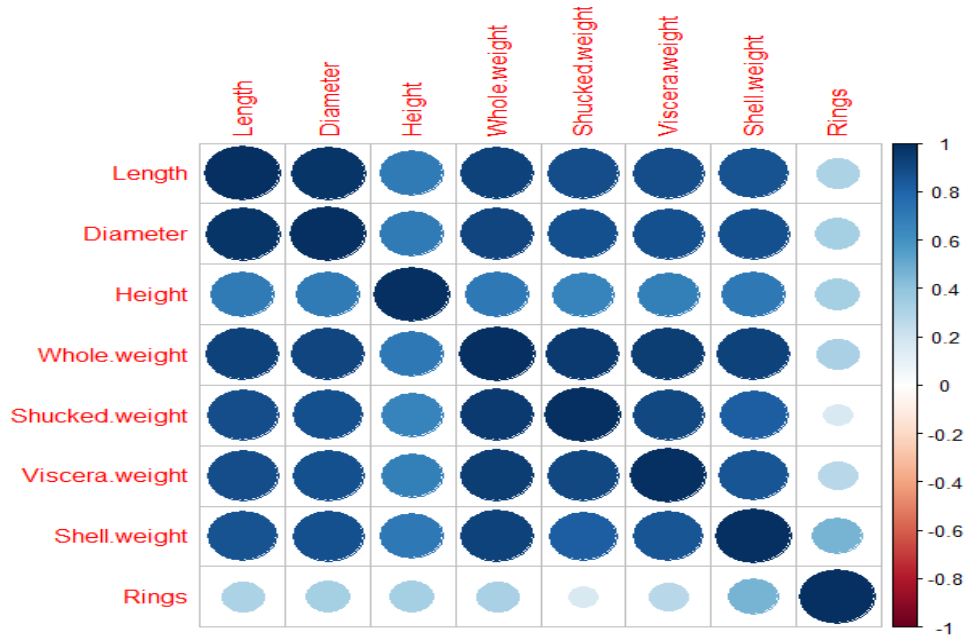
```
> library(ROCR)
> pred = prediction(PredictData,TestingData$Sex)
> perf = performance(roc.pred, measure = "tpr", x.measure = "fpr")
> pred = prediction(PredictData,TestingData$Sex)
> perf = performance(pred, measure = "tpr", x.measure = "fpr")
> plot(perf)
>
```



#Use the corrplot package to plot correlations between the predictors. How does this help explain the classifier performance?

Corelate <-cor(AbaloneData2[, -1])

corrplot(Corelate)



Answer- In the figure above, it can be seen that all the predictors exhibit a positive relationship except the Rings predictor. The Rings predictor has a low positive relationship while all other predictors have a high positive relationship.

Problem#2

library(data.table) **#import of data**

library(e1071) **#importing e1071**

```
> library(data.table)
data.table 1.14.2 using 8 threads (see ?getDTthreads).  Latest news: r-datatable.com
> library(e1071)
```

MushroomData1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data",

sep = ",", header = FALSE, col.names = c('class-name', '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'), as.is = 2&6, stringsAsFactors=T)

```
> MushroomData1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data",
+ sep = ",", header = FALSE, col.names = c('class-name', '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'), as.is = 2&6, stringsAsFactors=T)
>
```

colnames(MushroomData1)

```
> colnames(MushroomData1)
 [1] "class.name"               "cap.shape"                "cap.surface"              "cap.color"
 [5] "bruises."                 "odor"                     "gill.attachment"          "gill.spacing"
 [9] "gill.size"                "gill.color"               "stalk.shape"              "stalk.root"
[13] "stalk.surface.above.ring" "stalk.surface.below.ring" "stalk.color.above.ring"   "stalk.color.below.ring"
[17] "veil.type"                "veil.color"               "ring.number"              "ring.type"
[21] "spore.print.color"        "population"               "habitat"
>
```

table(MushroomData1$class.name)

```
> table(MushroomData1$class.name)

   e    p
4208 3916
>
```

str(MushroomData1)

```
> str(MushroomData1)
'data.frame':	8124 obs. of  23 variables:
 $ class.name              : chr  "p" "e" "e" "p" ...
 $ 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.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" ...
>
```

#Removing missing values

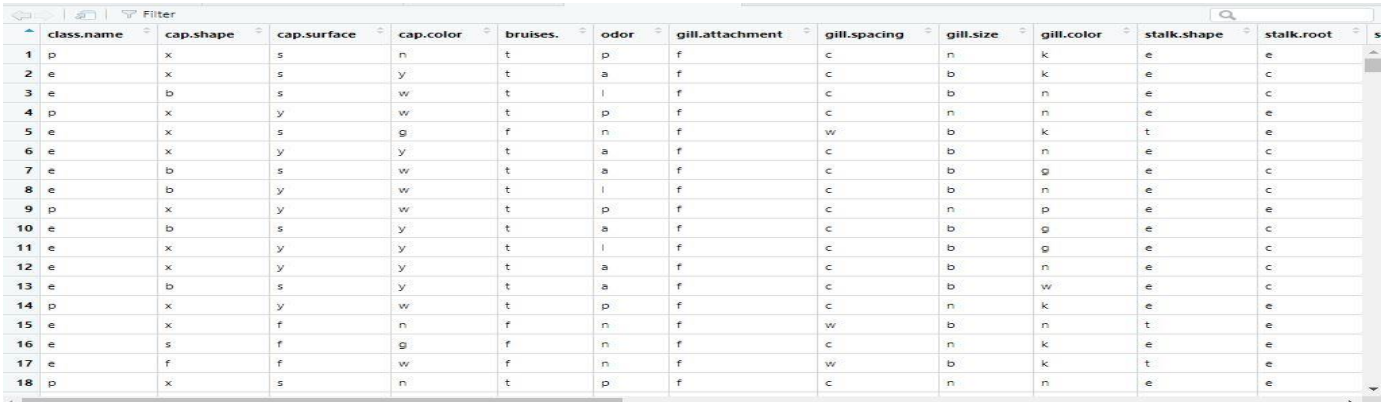
MushroomData1[MushroomData1=="?"]

```
> #Removing missing values
> MushroomData1[MushroomData1=="?"]
  [1] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [29] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [57] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [85] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[113] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[141] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[169] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[197] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[225] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[253] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[281] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[309] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[337] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[365] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[393] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[421] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[449] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[477] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[505] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
[533] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
```

```
 [505] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [533] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [561] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [589] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [617] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [645] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [673] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [701] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [729] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [757] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [785] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [813] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [841] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [869] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [897] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [925] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [953] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [981] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
 [ reached getOption("max.print") -- omitted 1480 entries ]
>
```

View(MushroomData1)

summary(MushroomData1)



	class.name	cap.shape	cap.surface	cap.color	bruises.	odor	gill.attachment	gill.spacing	gill.size	gill.color	stalk.shape	stalk.root
1	p	x	s	n	t	p	f	c	n	k	e	e
2	e	x	s	y	t	a	f	c	b	k	e	c
3	e	b	s	w	t	l	f	c	b	n	e	c
4	p	x	y	w	t	p	f	c	n	n	e	e
5	e	x	s	g	f	n	f	w	b	k	t	e
6	e	x	y	y	t	a	f	c	b	n	e	c
7	e	b	s	w	t	a	f	c	b	g	e	c
8	e	b	y	w	t	l	f	c	b	n	e	c
9	p	x	y	w	t	p	f	c	n	p	e	e
10	e	b	s	y	t	a	f	c	b	g	e	c
11	e	x	y	y	t	l	f	c	b	g	e	c
12	e	x	y	y	t	a	f	c	b	n	e	c
13	e	b	s	y	t	a	f	c	b	w	e	c
14	p	x	y	w	t	p	f	c	n	k	e	e
15	e	x	f	n	f	n	f	w	b	n	t	e
16	e	s	f	g	f	n	f	c	n	k	e	e
17	e	f	f	w	f	n	f	w	b	k	t	e
18	p	x	s	n	t	p	f	c	n	n	e	e

```
> view(MushroomData1)
> summary(MushroomData1)
  class.name         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
 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
>
```

TrainingIndex =sample(floor(0.80*nrow(MushroomData1)))

TrainingData = MushroomData1[TrainingIndex,]

TestingData = MushroomData1[-TrainingIndex,]

```
> TrainingIndex =sample(floor(0.80*nrow(MushroomData1)))
> TrainingData = MushroomData1[TrainingIndex,]
> TestingData = MushroomData1[-TrainingIndex,]
>
```

#naive Bayes

Model12 = naiveBayes(class.name~.,data=TrainingData)

Model12

```
> TestingData = MushroomData1[-TrainingIndex,]
> #naive Bayes
> Model12 = naiveBayes(class.name~.,data=TrainingData)
> Model12

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        e         p
0.5694722 0.4305278

Conditional probabilities:
   cap.shape
Y              b            c            f            k            s            x
  e 0.0710618752 0.0000000000 0.4128613888 0.0235071602 0.0086463118 0.4839232640
  p 0.0164403145 0.0007147963 0.4388849178 0.0164403145 0.0000000000 0.5275196569

   cap.surface
Y              f            g            s            y
  e 0.384760875 0.000000000 0.215347203 0.399891921
  p 0.271622588 0.001429593 0.288777698 0.438170122

   cap.color
Y              b            c            e            g            n            p            r            u            w
  e 0.012969468 0.006754931 0.168603080 0.239124561 0.284517698 0.012969468 0.004323156 0.004323156 0.158335585
  p 0.042887777 0.000000000 0.106862044 0.288777698 0.178341673 0.031451036 0.000000000 0.000000000 0.114367405
   cap.color
Y              y
  e 0.108078898
  p 0.237312366

   bruises.
Y            f          t
  e 0.2647933 0.7352067
  p 0.7769836 0.2230164

   odor
Y              a            c            f            l            m            n            p            s            y
  e 0.1080788976 0.0000000000 0.0000000000 0.1080788976 0.0000000000 0.7838422048 0.0000000000 0.0000000000 0.0000000000
  p 0.0000000000 0.0686204432 0.6454610436 0.0000000000 0.0003573981 0.0400285919 0.0914939242 0.0732666190 0.0807719800

   gill.attachment
Y              a            f
  e 0.001350986 0.998649014
  p 0.000000000 1.000000000

   gill.spacing
Y              c          w
  e 0.74952715 0.25047285
  p 0.96283059 0.03716941
```

#Prediction on Testing Data

testPred = predict(Model12,TestingData[,-1])

#Prediction on Training Data

trainPred = predict(Model12,TrainingData[,-1])

```
> #Prediction on Testing Data
> testPred = predict(Model12,TestingData[,-1])
> #Prediction on Training Data
> trainPred = predict(Model12,TrainingData[,-1])
> |
```

#Accuracy of Testing model

Mean1 = mean(testPred == TestingData$class.name)*100

c(Mean1)

```
> #Accuracy of Testing model
> Mean1 = mean(testPred == TestingData$class.name)*100
> c(Mean1)
[1] 88.92308
> |
```

#Accuracy of Training Model

Mean2 = mean(trainPred == TrainingData$class.name)*100

c(Mean2)

```
> #Accuracy of Training Model
> Mean2 = mean(trainPred == TrainingData$class.name)*100
> c(Mean2)
[1] 94.27604
> |
```

#Confusion Matrix

table(testPred, TestingData$class.name)

```
> #Confusion Matrix
> table(testPred, TestingData$class.name)

testPred    e    p
       e  370   43
       p  137 1075
> |
```

Problem #3

library(caret)

library(ggplot2)

```
> library(caret)
> library(ggplot2)
```

Yacht1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht_hydrodynamics.data",

sep = "", header = FALSE, col.names = c('long pos', 'pris coeff', 'len dis', 'beam rat', 'len beam', 'froude num',

'residue res'), as.is = 2&6, stringsAsFactors=T)

colnames(Yacht1)

```
> Yacht1 = read.csv(file = "https://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht_hydrodynamics.data",
+ sep = "", header = FALSE, col.names = c('long pos', 'pris coeff', 'len dis', 'beam rat', 'len beam', 'froude num',
+ 'residue res'), as.is = 2&6, stringsAsFactors=T)
> colnames(Yacht1)
[1] "long.pos"    "pris.coeff"  "len.dis"     "beam.rat"    "len.beam"    "froude.num"  "residue.res"
> |
```

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.

#80-20 test- train split

HydroIndex = createDataPartition(y = Yacht1$residue.res, p = 0.8, list = FALSE)

TrainingData = Yacht1[HydroIndex,]

TestingData = Yacht1[-HydroIndex,]

```
> HydroIndex = createDataPartition(y = Yacht1$residue.res, p = 0.8, list = FALSE)
> TrainingData = Yacht1[HydroIndex,]
> TestingData = Yacht1[-HydroIndex,]
>
```

#training fit for linear model

LModel1 = lm(residue.res~., data = TrainingData)

```
> #training fit for linear model
> LModel1 = lm(residue.res~., data = TrainingData)
>
```

What are the training MSE/RMSE and R2 results?

#Training MSE

TrainingPred = predict(LModel1, newdata = TrainingData)

TestingPred = predict(LModel1, newdata = TestingData)

TrainMSE1 = mean((TrainingData$residue.res - TrainingPred)^2)

c(TrainMSE1)

```
 #training MSE
 TrainingPred = predict(LModel1, newdata = TrainingData)
 TestingPred = predict(LModel1, newdata = TestingData)
 TrainMSE1 = mean((TrainingData$residue.res - TrainingPred)^2)
 c(TrainMSE1)
1] 77.90954
```

#training RMSE

TrainRMSE1 = sqrt(TrainMSE1)

c(TrainRMSE1)

#R2 results

summary(LModel1)$r.sq

```
> #training RMSE
> TrainRMSE1 = sqrt(TrainMSE1)
> c(TrainRMSE1)
[1] 8.826638
> #R2 results
> summary(LModel1)$r.sq
[1] 0.6637175
>
```

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 trainControl method), resulting in a training MSE/RMSE and R2 for each resample.

```
> TrainingControl = trainControl(method = "boot", number = 1000)
> LModel2 = train(residue.res~., data = TrainingData, method = "lm", trControl = TrainingControl)
```

#summary for RMSE and R2 for each sample

summary(LModel2$resample$RMSE)

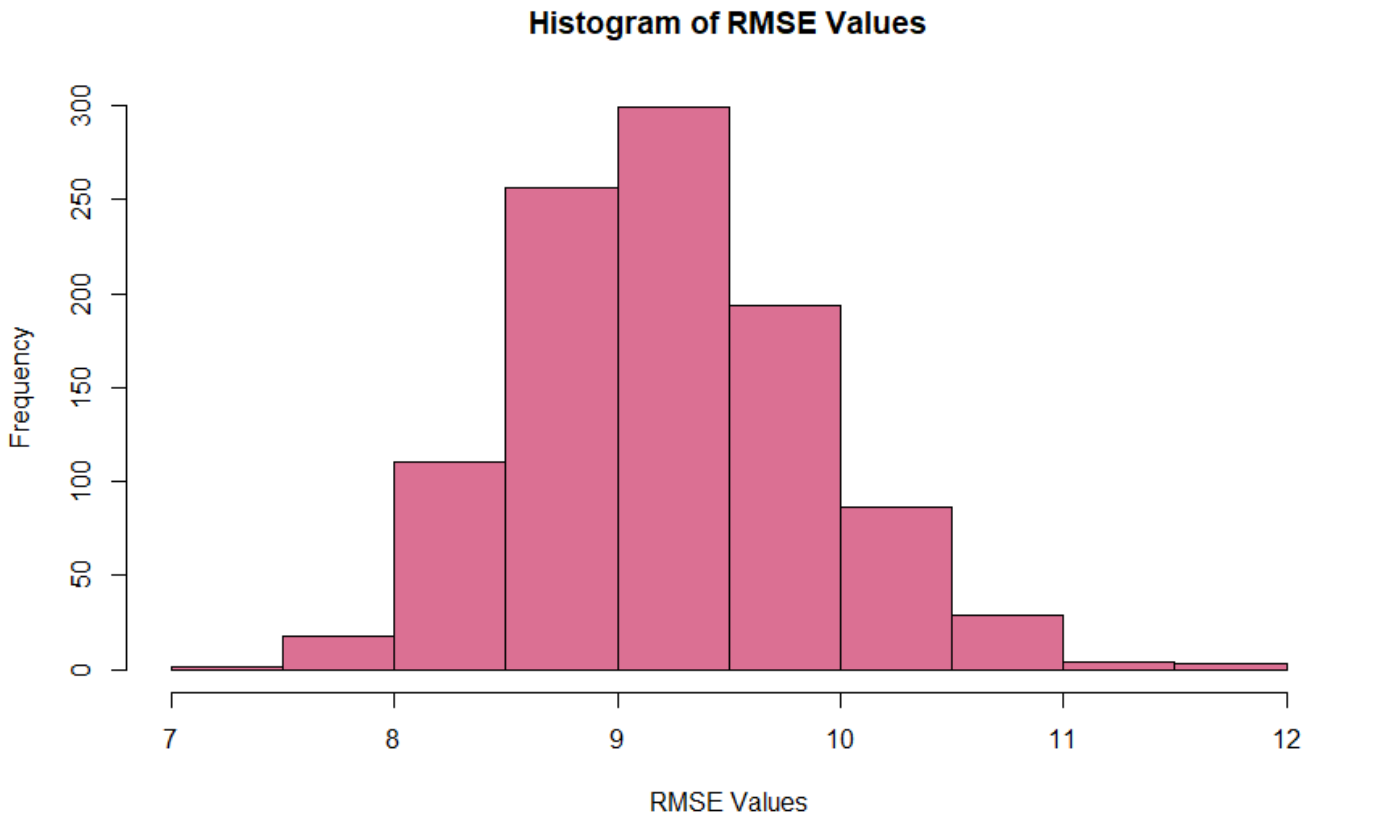
summary(LModel2$resample$Rsquared)

```
 0.5321  0.6247  0.6430  0.6425  0.6634  0.7218
> #summary for RMSE and R2 for each sample
> summary(LModel2$resample$RMSE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.131   8.776   9.208   9.230   9.640  11.804
> summary(LModel2$resample$Rsquared)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5321  0.6247  0.6430  0.6425  0.6634  0.7218
> |
```

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

hist(LModel2$resample$RMSE, xlab = "RMSE Values", main = "Histogram of RMSE Values", col="pale violet red")

```
> #Plot a histogram of the RMSE values, and provide a mean RMSE and R2 for the fit
> hist(LModel2$resample$RMSE, xlab = "RMSE Values", main = "Histogram of RMSE Values", col="pale violet red")
> |
```



#calculating mean RMSE and R2

RmseMean = mean(LModel2$resample$RMSE)

c(RmseMean)

R2Mean = mean(LModel2$resample$Rsquared)

c(R2Mean)

```
> #calculating mean RMSE and R2
> RmseMean = mean(LModel2$resample$RMSE)
> c(RmseMean)
[1] 9.229556
> R2Mean = mean(LModel2$resample$Rsquared)
> c(R2Mean)
[1] 0.6424604
>
```

#How do these values compare to the basic model?

mean(TrainRMSE1)

mean(Rsquare2)

```
> #How do these values compare to the basic model?
> mean(TrainRMSE1)
[1] 8.826638
> mean(Rsquare2)
[1] 0.6407169
>
```

Ans- We can see that RMSE and R2 values for the basic model are quite similar to that of the bootstrap model. The values have only a little difference.

#How does the performance on the test set for the original and boostrap model compare?

#Original Data

TestingPredOrig= predict(LModel2, newdata = TestingData)

TestMSEOrig = mean((TestingData$residue.res - TestingPredOrig)^2)

c(TestMSEOrig)

```
> TestingPredOrig= predict(LModel2, newdata = TestingData)
> TestMSEOrig = mean((TestingData$residue.res - TestingPredOrig)^2)
> c(TestMSEOrig)
 [1] 82.28057
>
```

BootRMSEOrig = sqrt(TestMSEOrig)

c(BootRMSEOrig)

summary(LModel2)$r.sq

```
[1] 82.28057
> BootRMSEOrig = sqrt(TestMSEOrig)
> c(BootRMSEOrig)
[1] 9.070864
>
> summary(LModel2)$r.sq
[1] 0.6637175
>
```

 # Performance on test set for Bootstrap Model

TestingControl = trainControl(method = "boot", number = 1000)

LModel2 = train(residue.res~., data = TestingData, method = "lm", trControl = TestingControl)

```
# Performance on test set for Bootstrap Model
TestingControl = trainControl(method = "boot", number = 1000)
LModel2 = train(residue.res~., data = TestingData, method = "lm", trControl = TestingControl)
```

summary(LModel2$resample$RMSE)

summary(LModel2$resample$Rsquared)

```
> summary(LModel2$resample$RMSE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.226   8.927   9.886  10.190  11.281  19.917
> summary(LModel2$resample$Rsquared)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1651  0.5574  0.6178  0.6101  0.6685  0.8755
>
```

RmseBootMean1 = mean(LModel2$resample$RMSE)

c(RmseBootMean1)

R2BootMean1 = mean(LModel2$resample$Rsquared)

c(R2BootMean1)

```
> RmseBootMean1 = mean(LModel2$resample$RMSE)
> c(RmseBootMean1)
[1] 10.18956
> R2BootMean1 = mean(LModel2$resample$Rsquared)
> c(R2BootMean1)
[1] 0.6101048
>
```

Ans-In this also, there is a little difference between the performance of the original test set and the performance of the bootstrap model.

#Problem4

library(caret)

GermanCredit1 = read.csv(file =

"https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data-numeric", sep = "",

header = FALSE, col.names = c("check_acc", "duration", "credit_his", "purpose",

"amount", "saving_acct", "present_emp", "installment_rate", "sex", "other_debtor",

"present_resid", "property", "age", "other_install", "housing", "n_credits", "job",

"n_people", "telephone", "foreign", "response", "v22", "v23", "v24", "v25"),as.is = 2&6, stringsAsFactors=T)

```
> library(caret)
> GermanCredit1 = read.csv(file =
+ "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data-numeric", sep = "",
+ header = FALSE, col.names = c("check_acc", "duration", "credit_his", "purpose",
+   "amount", "saving_acct", "present_emp", "installment_rate", "sex", "other_debtor",
+ "present_resid", "property", "age", "other_install", "housing", "n_credits", "job",
+ "n_people", "telephone", "foreign", "response", "v22", "v23", "v24", "v25"),as.is = 2&6, stringsAsFactors=T)
>
```

colnames(GermanCredit1)

```
> colnames(GermanCredit1)
 [1] "check_acc"       "duration"        "credit_his"      "purpose"         "amount"          "saving_acct"
 [7] "present_emp"     "installment_rate" "sex"            "other_debtor"    "present_resid"   "property"
[13] "age"             "other_install"   "housing"         "n_credits"       "job"             "n_people"
[19] "telephone"       "foreign"         "response"        "v22"             "v23"             "v24"
[25] "v25"
>
```

#converting category v25 to factor

GermanCredit1$v25 = factor(GermanCredit1$v25)

#Use the caret package to perform a 80/20 test-train split (via the createDataPartition function),

#and obtain a training fit for a logistic model via the glm package.

#80-20 test-train split

GemTrainIndex = createDataPartition(y = GermanCredit1$v25, p = 0.8, list = FALSE)

TrainingData = GermanCredit1[GemTrainIndex,]

TestingData = GermanCredit1[-GemTrainIndex,]

```
> #converting category v25 to factor
> GermanCredit1$v25 = factor(GermanCredit1$v25)
> GemTrainIndex = createDataPartition(y = GermanCredit1$v25, p = 0.8, list = FALSE)
> TrainingData = GermanCredit1[GemTrainIndex,]
> TestingData = GermanCredit1[-GemTrainIndex,]
>
```

LogisticModel1 = glm(v25~., family = binomial, data = TrainingData)

```
> LogisticModel1 = glm(v25~., family = binomial, data = TrainingData)
>
```

#What are the training Precision/Recall and F1 results?

#Training Set of Original

ActualVal = ifelse(LogisticModel1$fitted.values > 0.5,2,1)

ActualVal = factor(ActualVal)

ConMat = confusionMatrix(ActualVal, TrainingData$v25)

```
> ActualVal = ifelse(LogisticModel1$fitted.values > 0.5,2,1)
> ActualVal = factor(ActualVal)
> ConMat = confusionMatrix(ActualVal, TrainingData$v25)
>
```

#Training Precision

TrainingPrecision = ConMat$byClass[5] * 100

c(TrainingPrecision)

#Training Recall

TrainingRecall = ConMat$byClass[6] * 100

c(TrainingRecall)

#Training F1

TrainingF1 = ConMat$byClass[7] * 100

c(TrainingF1)

```
> #Training Precision
> TrainingPrecision = ConMat$byClass[5] * 100
> c(TrainingPrecision)
Precision
 80.79365
> #Training Recall
> TrainingRecall = ConMat$byClass[6] * 100
> c(TrainingRecall)
  Recall
90.89286
> #Training F1
> TrainingF1 = ConMat$byClass[7] * 100
> c(TrainingF1)
       F1
85.54622
> |
```

#Testing set of original

PredictData = predict(LogisticModel1, TestingData, type = "response")

ActualValTest = ifelse(PredictData > 0.5,2,1)

ActualValTest = factor(ActualValTest)

```
> PredictData = predict(LogisticModel1, TestingData, type = "response")
> ActualValTest = ifelse(PredictData > 0.5,2,1)
> ActualValTest = factor(ActualValTest)
> |
```

ConMatTest = confusionMatrix(ActualValTest, TestingData$v25)

c(ConMatTest)

```
> ConMatTest = confusionMatrix(ActualValTest, TestingData$v25)
> c(ConMatTest)
$positive
[1] "1"

$table
          Reference
Prediction   1   2
         1 124  30
         2  16  30

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
    0.77000000     0.41326531     0.70539361     0.82641914     0.70000000     0.01686939     0.05527028

$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value            Precision
```

```
Prediction   1   2
         1 124  30
         2  16  30

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
    0.77000000     0.41326531     0.70539361     0.82641914     0.70000000     0.01686939     0.05527028

$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value            Precision
           0.8857143            0.5000000            0.8051948            0.6521739            0.8051948
              Recall                   F1           Prevalence       Detection Rate Detection Prevalence
           0.8857143            0.8435374            0.7000000            0.6200000            0.7700000
   Balanced Accuracy
           0.6928571

$mode
[1] "sens_spec"

$dots
list()
```

#Testing Precision

TestingPrecision = ConMatTest$byClass[5] * 100

c(TestingPrecision)

#Testing Recall

TestingRecall = ConMatTest$byClass[6] * 100

c(TestingRecall)

#Testing F1

TestingF1 = ConMatTest$byClass[7] * 100

c(TestingF1)

```
> #Testing Precision
> TestingPrecision = ConMatTest$byClass[5] * 100
> c(TestingPrecision)
Precision
 80.51948
> #Testing Recall
> TestingRecall = ConMatTest$byClass[6] * 100
> c(TestingRecall)
  Recall
88.57143
> #Testing F1
> TestingF1 = ConMatTest$byClass[7] * 100
> c(TestingF1)
       F1
84.35374
>
```

Next, 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.

#Training the model

TrainingControl = trainControl(method = "cv", number = 10)

LogisticModel2 = train(v25~., data = TrainingData, family = "binomial", method = "glm", trControl = TrainingControl)

ActualValCV = ifelse(LogisticModel2$finalModel$fitted.values > 0.5,2,1)

ActualValCV = factor(ActualValCV)

```
> TrainingControl = trainControl(method = "cv", number = 10)
> LogisticModel2 = train(v25~., data = TrainingData, family = "binomial", method = "glm", trControl = TrainingControl)
> ActualValCV = ifelse(LogisticModel2$finalModel$fitted.values > 0.5,2,1)
> ActualValCV = factor(ActualValCV)
>
```

ConMatCV = confusionMatrix(ActualValCV, TrainingData$v25)

c(ConMatCV)

```
> ConMatCV = confusionMatrix(ActualValCV, TrainingData$v25)
> c(ConMatCV)
$positive
[1] "1"

$table
          Reference
Prediction   1   2
         1 509 121
         2  51 119

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
  7.850000e-01   4.415584e-01   7.548824e-01   8.130000e-01   7.000000e-01   3.949089e-08   1.431177e-07

$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value            Precision
```

```
Prediction   1   2
         1 509 121
         2  51 119

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
  7.850000e-01   4.415584e-01   7.548824e-01   8.130000e-01   7.000000e-01   3.949089e-08   1.431177e-07

$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value            Precision
           0.9089286            0.4958333            0.8079365            0.7000000            0.8079365
              Recall                   F1           Prevalence       Detection Rate Detection Prevalence
           0.9089286            0.8554622            0.7000000            0.6362500            0.7875000
   Balanced Accuracy
           0.7023810

$mode
[1] "sens_spec"

$dots
list()
```

#Training Precision with 10 fold CV

TrainingPrecisionFold = ConMatCV$byClass[5] * 100

c(TrainingPrecisionFold)

#Training Recall with 10 fold CV

TrainingRecallFold = ConMatCV$byClass[6] * 100

c(TrainingRecallFold)

#Training F1 with 10 fold CV

TrainingF1Fold = ConMatCV$byClass[7] * 100

c(TrainingF1Fold)

Answer- There is not much difference in the values of the original set and that of the set with k=10 fold cross validation.

```
> #Training Precision with 10 fold CV
> TrainingPrecisionFold = ConMatCV$byClass[5] * 100
> c(TrainingPrecisionFold)
Precision
 80.79365
> #Training Recall with 10 fold CV
> TrainingRecallFold = ConMatCV$byClass[6] * 100
> c(TrainingRecallFold)
   Recall
90.89286
> #Training F1 with 10 fold CV
> TrainingF1Fold = ConMatCV$byClass[7] * 100
> c(TrainingF1Fold)
       F1
85.54622
>
```

#Testing set with 10 fold CV

PredictDataTestFold = predict(LogisticModel2, TestingData, type = "prob")

ActualValTestFold = ifelse(PredictData > 0.5,2,1)

ActualValTestFold = factor(ActualValTest)

```
> PredictDataTestFold = predict(LogisticModel2, TestingData, type = "prob")
> ActualValTestFold = ifelse(PredictData > 0.5,2,1)
> ActualValTestFold = factor(ActualValTest)
> |
```

ConMatTestFold = confusionMatrix(ActualValTest, TestingData$v25)

c(ConMatTestFold)

```
> ConMatTestFold = confusionMatrix(ActualValTest, TestingData$v25)
> c(ConMatTestFold)
$positive
[1] "1"

$table
          Reference
Prediction   1   2
         1 124  30
         2  16  30

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
    0.77000000     0.41326531     0.70539361     0.82641914     0.70000000     0.01686939     0.05527028

$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value            Precision
           0.8857143            0.5000000            0.8051948            0.6521739            0.8051948
              Recall                   F1           Prevalence       Detection Rate Detection Prevalence
           0.8857143            0.8435374            0.7000000            0.6200000            0.7700000
   Balanced Accuracy
           0.6928571

$mode
[1] "sens_spec"

$dots
list()

> |
```

#Testing Precision with 10 fold CV

TestingPrecisionFold = ConMatTestFold$byClass[5] * 100

c(TestingPrecisionFold)

#Testing Recall with 10 fold CV

TestingRecallFold = ConMatTestFold$byClass[6] * 100

c(TestingRecallFold)

#Testing F1 with 10 fold Cv

TestingF1Fold = ConMatTestFold$byClass[7] * 100

c(TestingF1Fold)

```
> #Testing Precision with 10 fold CV
> TestingPrecisionFold = ConMatTestFold$byClass[5] * 100
> c(TestingPrecisionFold)
Precision
 80.51948
> #Testing Recall with 10 fold CV
> TestingRecallFold = ConMatTestFold$byClass[6] * 100
> c(TestingRecallFold)
  Recall
88.57143
> #Testing F1 with 10 fold Cv
> TestingF1Fold = ConMatTestFold$byClass[7] * 100
> c(TestingF1Fold)
      F1
84.35374
>
```

Answer- As seen above, in the test set also, not much difference can be seen between the values of the original set and that of the test set with k=10-fold cross-validation.

Submitted By: -

Aastha Dhir

CWID-A20468022

adhir2@hawk.iit.edu