Homework 2

Sohaib Syed

2023-02-12

Rectitation Exercises

1.1 Chapter 4

Exercise 4

a First we declare that 10% of the range of X is 0.1, as we assume that X is evenly distributed along [0,1]. So for the example test observation in the problem the values 0.55 and 0.65 come from going 0.05 below 0.6 and 0.05 above 0.6. This range is 0.1 which is 10% of range of X.

using this we can see that if x falls below 0.05 (ex. 0.04), then the lower bound would be adjusted to be 0, not -0.01, and upper bound would be calculated as usual x+0.05 as we assume X is [0,1], no negatives. Similarly, if x is above 0.95 (ex 0.96), the upper bound is set to 1, not 1.01.

This shows that even if we are allowed to use 10% of the range of X, we will be left with even smaller fraction of available observations, as the boundary will eliminate some points.

We can retrieve the average by integrating on the three boundaries that make the range.

$\left(\int_0^{0.05} x + 0.05 \, dx\right) + \left(\int_{0.0}^{0.95} 10 \, dx\right) + \left(\int_{0.95}^{1} 1.05 - x \, dx\right)$

$= 0.0975 = 9.75\%$

b Both X1 and X2 are evenly distributed on [0,1], from part a we know that for a single feature the fraction was 9.75%. With X1 and X2 assumed to be independent, they have same distribution and we also use 10% for both, then the product of the two fractions .0975*.0975= 0.00950625

c Using same method and assumptions from part b, we will do 0.0975^100=

```
0.0975^100
```

```
## [1] 7.951729e-102
```

d parts earlier have shown that as p gets large the fraction of points available to use for prediction will eventually be 0. So when using KNN with large p to predict a new point, there is likely to be 0 points near that test observation.

e We are now given the average fraction. So for p=1, the average length would be 0.1^1= 0.1; For p=2, 0.1^(1/2)= 0.3162 and p=100, 0.1 ^ 1/100 ~ .977. This comes from solving for the side of a cube given its volume of 0.1. We were told that p=1 is a line and p=2 is a square. For volume of square to be given as 0.1, i know that the length of side raised to power 2 (for a square) will result in 0.1, so rearrange s^p=V to V^1/p= s, for side length.

This further shows how most of the observations are on the edges of the hyper cube which is why KNN on high dimension data sets is not going to perform well

Exercise 6

a Equation 4.7 gives P(x)= e^B0+B1x1+...+Bpxp/ 1+e^B0+B1x1+...+Bpxp

so for this problem: e^-6+0.05(40)+1(3.5)/ 1+e^-6+0.05(40)+1(3.5)=

```
exp(-6+0.05*40+3.5)/ (1+exp(-6+0.05*40+3.5))
```

```
## [1] 0.3775407
```

37.75% chance student receives an A

b We use the same equation but instead solve for X1:

e^-6+0.05(X1)+1(3.5)/ 1+e^-6+0.05(X1)+1(3.5)=0.5

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

0.5= e^-6+0.05(X1)+1(3.5)- 0.5 (e^-6+0.05(X1)+1(3.5))

0.5=0.5 (e^-6+0.05(X1)+1(3.5))

1= (e^-6+0.05(X1)+1(3.5))

0=-6+.05(x1)+3.5 2.5=.05(x1)

x1=2.5/0.5=50

Study 50 hours for a 50% chance of getting an A

Exercise 7

Equation 4.15 P(Y=k|X=x)= pi_k * f_k(X)/ sum (pi_l *f_l(x))

P(Yes|4)= (P(Yes) * (1/(2 * pi * $simga^{2)}$1/2) * $e^{-(x-u)}$2/2* simga^2)/ (P(yes) * (1/(2 * pi * $simga^{2)}$1/2) * $e^{-(x-u)}$2/2* simga^2)

= .8 * (1/ 2 * pi * 36) * e^ -(4-10)^2/72 = .8 * (1/ $(72pi)^{1/2)*}$ ^{e}-2

```
numer<-0.8 * (1/sqrt(72*pi))*exp((-(4-10)**2)/72)
denom= (numer) + .2* (1/sqrt(72*pi))*exp((-(4-0)**2)/72)
(numer/denom)*100
```

```
## [1] 75.18525
```

There is a 75.19% chance that a company will issue dividends given its percent profit from last year was X=4

Exercise 9

a p(x)/1-p(x)=.37 p(x)=.37-.37p(x) 1.37p=.37 .37/1.37=p(x) = .27

On average .27 of people with odds of .37 of defaulting on credit cards payments will default

b p(x)=.16

odds=p(x)/1-p(x)

.16/1-.16= 0.1905

.1905 odds of this person defaulting

1.2 Chapter 5

Exercise 2

a With bootstrapping with replacement we can assume each observation is independent and equal probability of being chosen so each observation has 1/n chance of being chose, thus it has a 1-(1/n) chance of not being chosen.

b As stated in part(a) each observation has 1/n chance of being chosen thus a 1-1/n chance of not being chose so once again the probability is 1-1/n

c To not be in a bootstrap sample at all that that has n observation means that the jth sample had to not be chosen n times, which means (1-1/n)* ... *(1-1/n)= (1-1/n)^n.

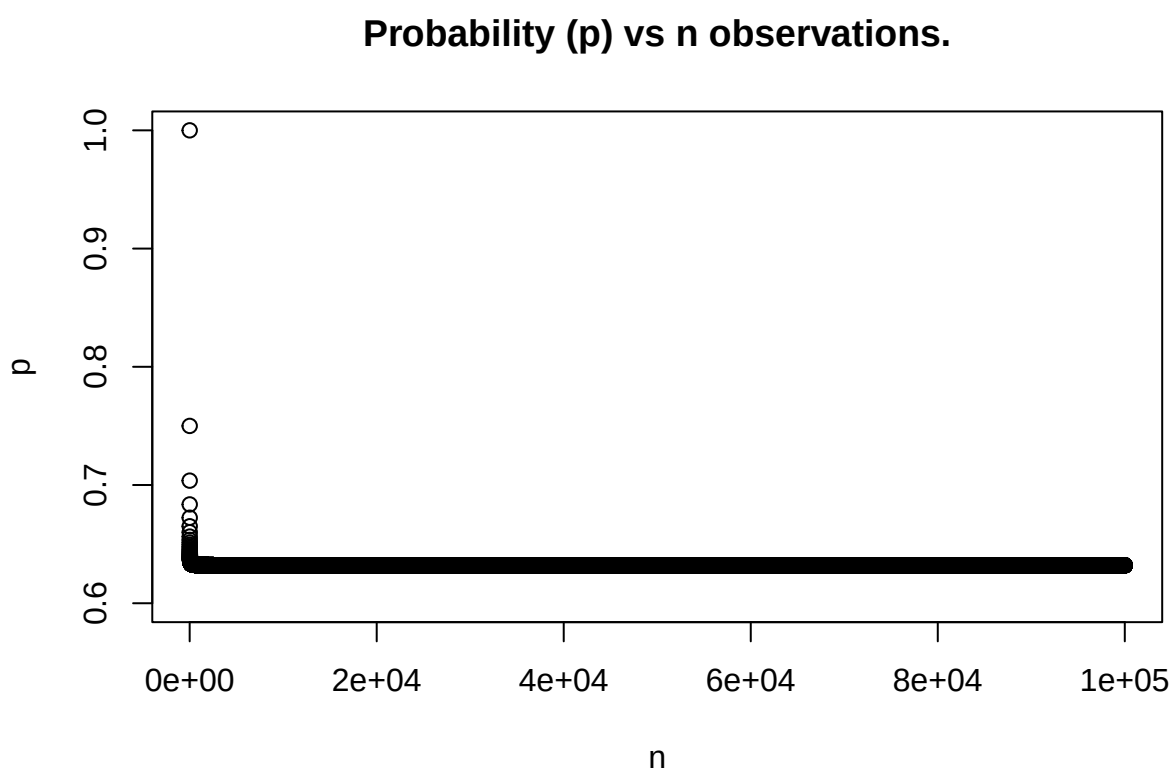
d To be in the bootstrap sample we take the complement of equation in part c: 1- (1-1/n)^n

1(1-1/5)^5= 0.672

e 1 - (1-1/100)^100 = 0.634

f 1 - (1-1/10000)^10000 = 0.632

```
n = 1:100000
p = 1-(1-1/n)^n
plot(n,p, main="Probability (p) vs n observations.",ylim=c(.6,1))
```



g

It is observable that there is a very quick decay of probability to the asymptotic value just above 0.6. From the previous parts this asymptotic value is likely 0.63

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

h

```
## [1] 0.6417
```

The result is further evidence for the previous answers that the probability of a jth observation not being in a bootstrap sample with n observations is around .63 as was shown in part (f).

Exercise 3

a n observations are split in to k non-overlapping groups (n/k). The rest of the groups are used as training set. This is done k times, as a different group is chosen as validation. Example 50 observations with 10 groups. First 10 points are used as validation the rest 40 as training, and we get one point of cross-validation error, then next we get points 11:20 and get another CVerror. This is done k times and the test error is estimated by taking average of CVerrors from each group.

b

i. Kfold has less variance than validation estimate approach, has better estimate of test error since all data is used, not just a subset.

ii. LOOCV may pose computational problems, especially if n is extremely large, k-fold often gives more accurate estimates of the test error rate than does LOOCV. This has to do with a bias-variance trade-off. LOOCv has high correlation between observations

Practicum Problems

2.1 Problem 1

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
df<-read.table('abalone.data',header=FALSE,sep= ',', col.names=c('sex','length','diameter','height','wh
df<-subset(df,sex!='I')
df$sex<-as.factor(df$sex)
myIndex<-createDataPartition(df$sex,p=0.8,list=F)

trainSet<-df[myIndex,]
testSet<-df[-myIndex,]

fit1<-glm(sex~.,data=trainSet,family=binomial)
summary(fit1)
```

```
##
## Call:
## glm(formula = sex ~ ., family = binomial, data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8683  -1.1979   0.8411   1.1168   1.5609
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     3.11577    0.52442   5.941 2.83e-09 ***
## length         -2.73838    2.27499  -1.204 0.228710
## diameter       -4.19472    2.69202  -1.558 0.119184
## height         -4.48834    2.35123  -1.909 0.056270 .
## whole.weight   -0.09386    0.81162  -0.116 0.907933
## shucked.weight  3.53197    0.97912   3.607 0.000309 ***
## viscera.weight -2.55266    1.41490  -1.804 0.071211 .
## shell.weight    0.54155    1.25798   0.430 0.666840
## rings           0.01183    0.01770   0.668 0.504041
## ---
```

```
## 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: 3055.9  on 2260  degrees of freedom
## AIC: 3073.9
##
## Number of Fisher Scoring iterations: 4
```

From the summary only shucked weight looks to be relevant as it has lowest p-value

```
confint(fit1)
```

```
## Waiting for profiling to be done...
```

```
##                       2.5 %      97.5 %
## (Intercept)     2.10419833  4.16075379
## length         -7.20295246  1.72137691
## diameter       -9.48495760  1.07633170
## height         -9.39166616 -0.57474459
## whole.weight   -1.69195255  1.49965088
## shucked.weight  1.62277421  5.46764668
## viscera.weight -5.33696566  0.21579200
## shell.weight   -1.92744477  3.01466950
## rings          -0.02285716  0.04658982
```

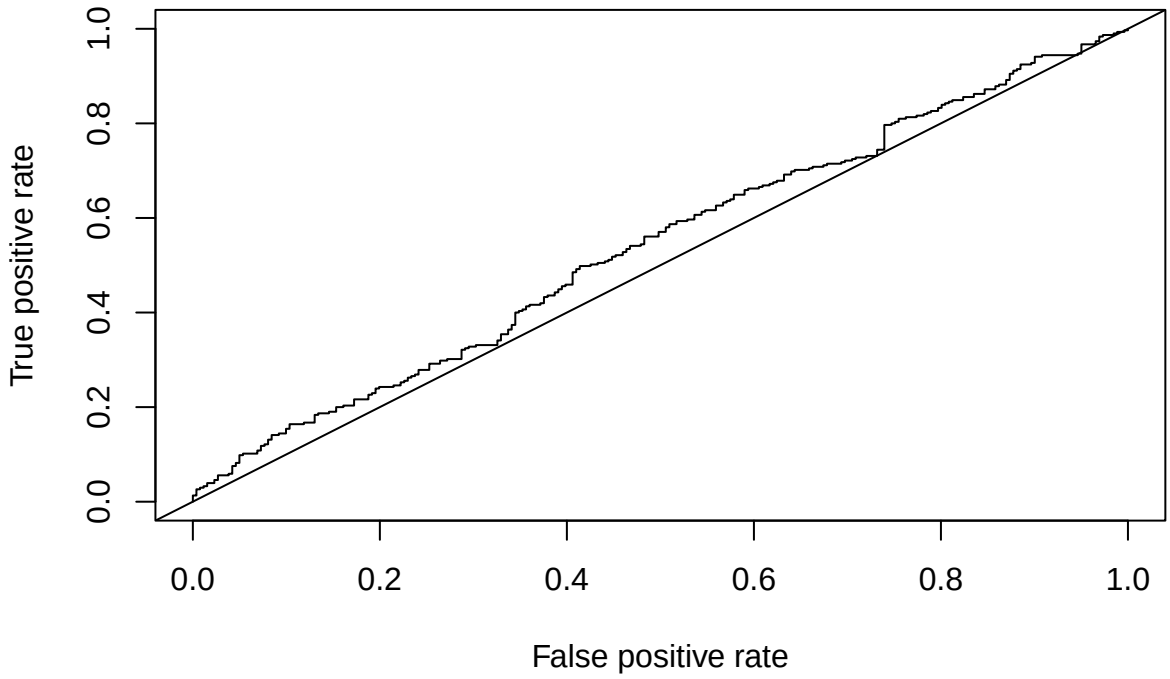
As the intervals show, shucked weight is the only predictor that does not contain 0 in the 95% confidence interval range. This means that for shucked weight we can reject the null hypothesis of it not being relevant to the model. We fail to reject null hypothesis for all other predictors.

```
library(ROCR)
predicted_abalone<-predict(fit1,testSet,type='r')
predicted_abalone_factored <- ifelse(predicted_abalone>0.5,'M','F')
confusionMatrix(as.factor(predicted_abalone_factored), testSet$sex)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   F   M
##          F  96  95
##          M 165 210
##
##                Accuracy : 0.5406
##                  95% CI : (0.4986, 0.5823)
##     No Information Rate : 0.5389
##     P-Value [Acc > NIR] : 0.4836
##
##                   Kappa : 0.0575
##
##  Mcnemar's Test P-Value : 1.876e-05
##
```

```
##             Sensitivity : 0.3678
##             Specificity : 0.6885
##          Pos Pred Value : 0.5026
##          Neg Pred Value : 0.5600
##              Prevalence : 0.4611
##          Detection Rate : 0.1696
##    Detection Prevalence : 0.3375
##       Balanced Accuracy : 0.5282
##
##        'Positive' Class : F
##
```

```
pred <- prediction(predicted_abalone, testSet$sex)
perf <- performance(pred,"tpr","fpr")
plot(perf)
abline(0,1)
```

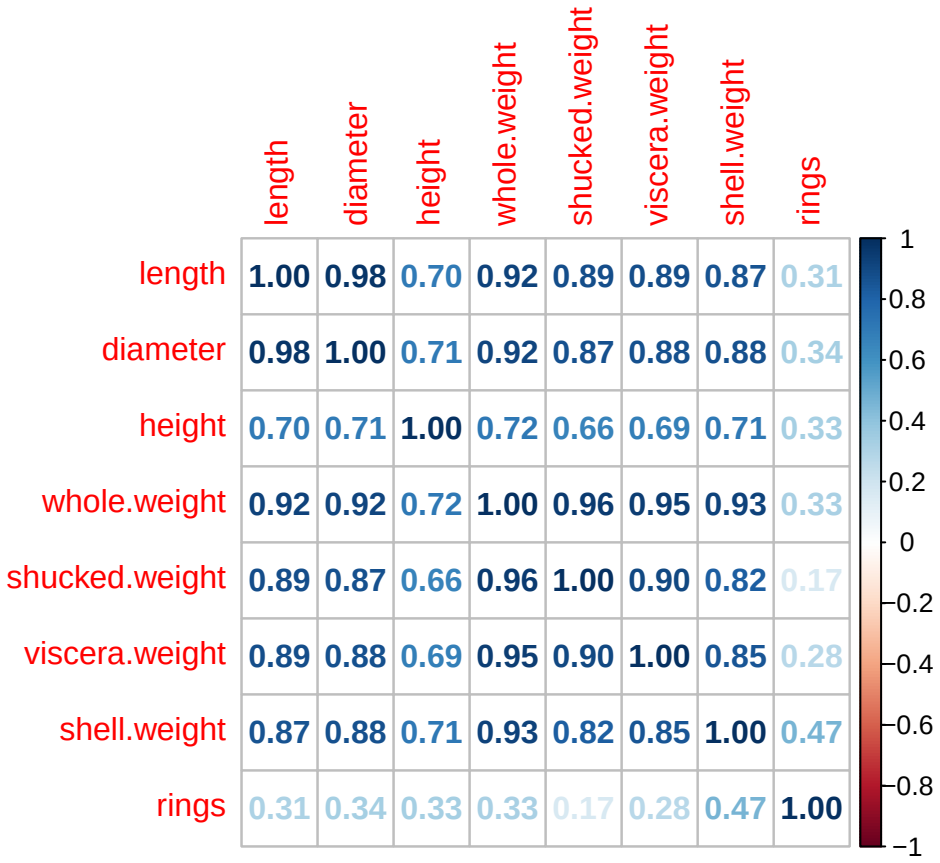


The accuracy is around 57% and the ROC curve shows that the classifier is above the 'coin-flip' line, so it isn't random.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(cor(df[,-1]),method = "number")
```



From this correlation plot we see that apart from rings, the other predictors were highly correlated, a lot of collinearity, thus the model was not extracting useful information from the predictors leading to poor classifier performance.

2.2 Problem 2

```
mushrooms_df<-read.table('agaricus-lepiota.data',header=F,sep = ',', col.names = c('Edibility','cap-shap

mushrooms_df<- subset(mushrooms_df,stalk.root != '?')
dim(mushrooms_df)
```

```
## [1] 5644   23
```

I removed ? values entirelt from dataset and we still have reasonabale amount of data left

```
library(e1071)
mushrooms_df$Edibility <- factor(mushrooms_df$Edibility)
# https://stackoverflow.com/questions/17200114/how-to-split-data-into-training-testing-sets-using-sampl

smp_size <- floor(0.80 * nrow(mushrooms_df))
```

```
train_ind <- sample(seq_len(nrow(mushrooms_df)), size = smp_size)

shroom_train <- mushrooms_df[train_ind, ]
shroom_test <- mushrooms_df[-train_ind, ]

nb.fit<-naiveBayes(Edibility ~ ., data =shroom_train)

shroom_train_pred<-predict(nb.fit, shroom_train[,2:length(shroom_train)])

shroom_test_pred<-predict(nb.fit, shroom_test[,2:length(shroom_test)])

cat("Accuracy of Training Model: ",mean(shroom_train_pred == shroom_train$Edibility)*100,"%\n")
```

```
## Accuracy of Training Model:  95.1052 %
```

```
cat("Accuracy of Testing Model: ",mean(shroom_test_pred == shroom_test$Edibility)*100,"%")
```

```
## Accuracy of Testing Model:  96.63419 %
```

```
table(shroom_test_pred,shroom_test$Edibility)
```

```
##
## shroom_test_pred   e   p
##                e 730  34
##                p   4 361
```

34 False Positives on test set

2.3 Problem 3

```
library(caret)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
yacht_df<-read.table('yacht_hydrodynamics.data',header=F,sep='',col.names = c("Longitudinal position","
myIndex_yacht<-createDataPartition(yacht_df$Residuary,p=0.8,list=F)


yacht_train <- yacht_df[myIndex_yacht,]

yacht_test <- yacht_df[-myIndex_yacht,]

lm_yacht<-lm(Residuary~.,data=yacht_train)

summary(lm_yacht)
```

```
##
## Call:
## lm(formula = Residuary ~ ., data = yacht_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.438  -7.198  -1.722   5.435  32.336
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -21.70000   31.15349  -0.697    0.487
## Longitudinal.position       0.27279    0.36651   0.744    0.457
## Prismatic.coefficient      -7.68427   50.83795  -0.151    0.880
## Length.displacement.ratio  -0.81195   16.25678  -0.050    0.960
## Beam.draught.ratio          0.04933    6.41881   0.008    0.994
## X.Length.beam.ratio         2.02891   16.20266   0.125    0.900
## Froude.number             118.83942    5.60361  21.208   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.838 on 241 degrees of freedom
## Multiple R-squared:  0.6534, Adjusted R-squared:  0.6448
## F-statistic: 75.74 on 6 and 241 DF,  p-value: < 2.2e-16
```

```
yachttrain.predict <- predict(lm_yacht, newdata = yacht_train)
yachttest.predict <- predict(lm_yacht, newdata =yacht_test)
cat("training MSE" , mse(yacht_train$Residuary, yachttrain.predict), "\n")
```

```
## training MSE 75.90819
```

```
cat("training RMSE" , rmse(yacht_train$Residuary, yachttrain.predict), "\n")
```

```
## training RMSE 8.712531
```

```
cat("training adjusted R^2" , summary(lm_yacht)$r.squared, "\n")
```

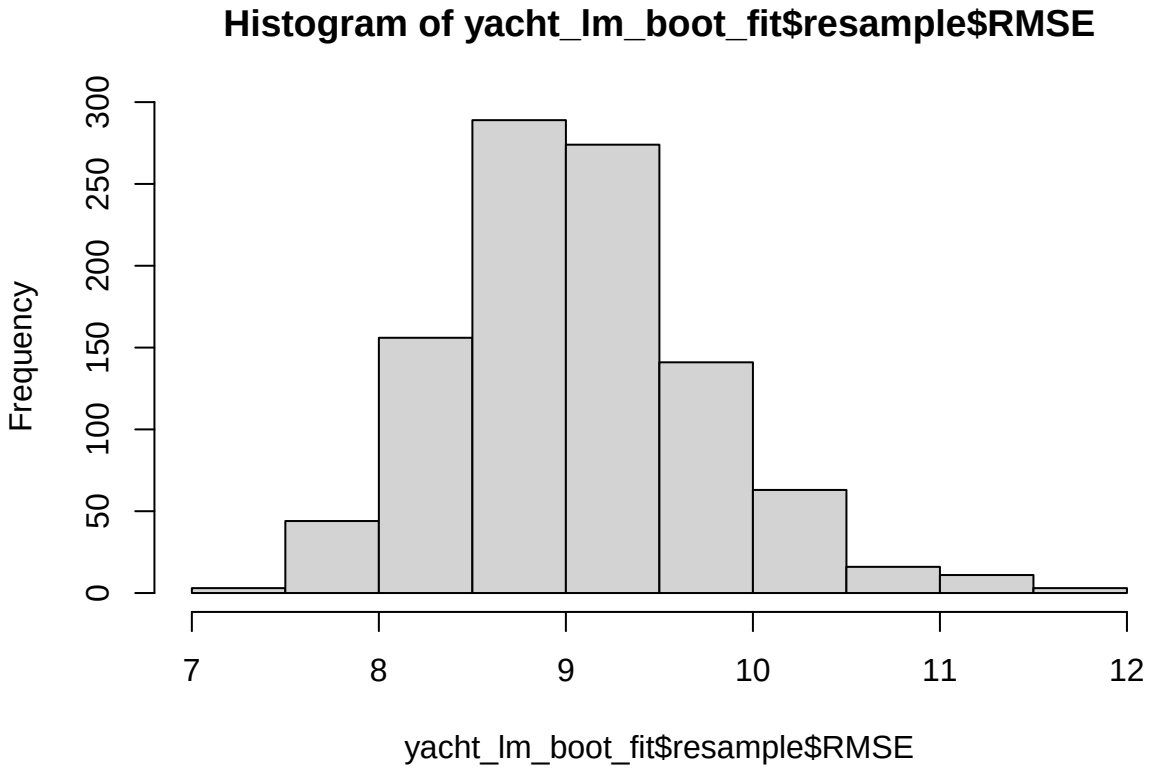
```
## training adjusted R^2 0.6534496
```

The R^2 is .6652, the adjusted R^2 is .6569. The RMSE is 8.938

```
train.control <- trainControl(method = "boot", number = 1000)


yacht_lm_boot_fit <- train(Residuary~., data = yacht_train, method = "lm", trControl = train.control)

hist(yacht_lm_boot_fit$resample$RMSE)
```




```
cat("bootstrap mean RMSE" , mean(yacht_lm_boot_fit$resample$RMSE), "\n")


## bootstrap mean RMSE 9.070142

cat("bootstrap mean R2" , mean(yacht_lm_boot_fit$resample$Rsquared), "\n")


## bootstrap mean R2 0.6334167
```

The RMSE for bootstrap is higher than regular fit, and R^2 is lower for bootstrap

```
rss <- function(label, data){
    return (sum((label - data)^2))
}
tss <- function(label){
    return (sum((label - mean(label))^2))
}
```

```
r2 <- function(label, data){
    return(1-(rss(label, data)/tss(label)))
}


cat("test MSE" , mse(yacht_test$Residuary, yachttest.predict), "\n")
```

```
## test MSE 90.9442
```

```
cat("test RMSE" , rmse(yacht_test$Residuary, yachttest.predict), "\n")
```

```
## test RMSE 9.536467
```

```
cat("test R^2" , r2(yacht_test$Residuary, yachttest.predict), "\n")
```

```
## test R^2 0.6631414
```

```
yacht_pred_boot = predict(yacht_lm_boot_fit,yacht_test)



cat("test MSE" , mse(yacht_test$Residuary, yacht_pred_boot), "\n")
```

```
## test MSE 90.9442
```

```
cat("test RMSE" , rmse(yacht_test$Residuary, yacht_pred_boot), "\n")
```

```
## test RMSE 9.536467
```

```
cat("test R^2" , r2(yacht_test$Residuary, yacht_pred_boot), "\n")
```

```
## test R^2 0.6631414
```

bootstrap and orginal model test values are the same for the metrics

2.4 Problem 4

```
german_df<-read.table('german.data-numeric',header=F)
german_df$V25 <-factor(german_df$V25)
german_trainIndex <- createDataPartition(german_df$V25 , p = 0.8, list = FALSE)

germantrain <- german_df[german_trainIndex,]

germantest <- german_df[-german_trainIndex,]

germanfit <- glm(V25~., data=germantrain, family=binomial(link = "logit"))
```

```
german_fitted <- ifelse(germanfit$fitted.values > 0.5,2,1)
german_fitted <- factor(german_fitted)

germanconfusion<- confusionMatrix(german_fitted, germantrain$V25,mode = 'everything')
print(germanconfusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 502 111
##          2  58 129
##
##                Accuracy : 0.7888
##                  95% CI : (0.7588, 0.8166)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 9.675e-09
##
##                   Kappa : 0.4632
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.8964
##             Specificity : 0.5375
##          Pos Pred Value : 0.8189
##          Neg Pred Value : 0.6898
##               Precision : 0.8189
##                  Recall : 0.8964
##                      F1 : 0.8559
##              Prevalence : 0.7000
##          Detection Rate : 0.6275
##    Detection Prevalence : 0.7662
##       Balanced Accuracy : 0.7170
##
##        'Positive' Class : 1
##
```

Precision train : 0.8126
Recall train : 0.8982
F1 train : 0.8533

```
german_pred <- predict(germanfit, germantest, type = "response")

german_pred_fit <- ifelse(german_pred > 0.5,2,1)
german_pred_fit <- factor(german_pred_fit)

germanconfusiontrain <- confusionMatrix(german_pred_fit, germantest$V25,mode='everything')
print(germanconfusiontrain)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   1   2
##          1 125  36
##          2  15  24
##
##                Accuracy : 0.745
##                  95% CI : (0.6787, 0.8039)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.093436
##
##                   Kappa : 0.3254
##
##  Mcnemar's Test P-Value : 0.005101
##
##             Sensitivity : 0.8929
##             Specificity : 0.4000
##          Pos Pred Value : 0.7764
##          Neg Pred Value : 0.6154
##               Precision : 0.7764
##                  Recall : 0.8929
##                      F1 : 0.8306
##              Prevalence : 0.7000
##          Detection Rate : 0.6250
##    Detection Prevalence : 0.8050
##       Balanced Accuracy : 0.6464
##
##        'Positive' Class : 1
##
```

Precision test : 0.8025
Recall test : 0.9286
F1 test : 0.8609

```
train.controlCV <-  trainControl(method = "cv", number = 10)

germanCV <- train(V25~., data = germantrain, method = "glm", family = "binomial", trControl = train.con


germanfitCV <- ifelse(germanCV$finalModel$fitted.values > 0.5,2,1)
germanfitCV <- factor(germanfitCV)

confusion_matrix_cv <-confusionMatrix(germanfitCV, germantrain$V25, mode ='everything')
print(confusion_matrix_cv)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 502 111
##          2  58 129
##
##                Accuracy : 0.7888
##                  95% CI : (0.7588, 0.8166)
##     No Information Rate : 0.7
```

```
##     P-Value [Acc > NIR] : 9.675e-09
##
##                   Kappa : 0.4632
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.8964
##             Specificity : 0.5375
##          Pos Pred Value : 0.8189
##          Neg Pred Value : 0.6898
##               Precision : 0.8189
##                  Recall : 0.8964
##                      F1 : 0.8559
##              Prevalence : 0.7000
##          Detection Rate : 0.6275
##    Detection Prevalence : 0.7662
##       Balanced Accuracy : 0.7170
##
##        'Positive' Class : 1
##
```

Precision CV train : 0.8126
Recall CV train : 0.8982
F1 CV train : 0.8533

```
german_cv_pred <- predict(germanCV$finalModel, newdata = germantest, type = "response")


germancvpred_fit <- ifelse(german_cv_pred > 0.5,2,1)
germancvpred_fit <- factor(germancvpred_fit)

confusion_matrix_cv_test = confusionMatrix(germancvpred_fit, germantest$V25,mode = 'everything')
print(confusion_matrix_cv_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 125  36
##          2  15  24
##
##                Accuracy : 0.745
##                  95% CI : (0.6787, 0.8039)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.093436
##
##                   Kappa : 0.3254
##
##  Mcnemar's Test P-Value : 0.005101
##
##             Sensitivity : 0.8929
##             Specificity : 0.4000
##          Pos Pred Value : 0.7764
```

```
##          Neg Pred Value : 0.6154
##               Precision : 0.7764
##                  Recall : 0.8929
##                      F1 : 0.8306
##              Prevalence : 0.7000
##          Detection Rate : 0.6250
##    Detection Prevalence : 0.8050
##       Balanced Accuracy : 0.6464
##
##        'Positive' Class : 1
##
```

Precision CV test : 0.8025
Recall CV test : 0.9286
F1 CV test : 0.8609

the metrics for orginal model and for CV model are the same