Assignment 02

Sumanth Donthula

2023-01-31

1. Recitation Exercises

1.1.Chapter 4

Exercise 4:

(a) Since, the data is uniformly distributed the probability for each point will be same. So if we have 100 points we need 10 points, averagely 10/100 points with a fraction of 0.1 or 1/10th of total points.

(b) Now we have 2 features which are uniformly distributed with 10% observations used in prediction. So, we need (10/100)*(10/100)=100/10000=0.01. So, we need 1% of total observations or 1/100 the fraction of total observations.

(c) As we can see for 100 features we will be using (10/100)ˆ100 fraction of points.

(d) As we can see if we increase the number of predictors the fraction of points used for making prediction gradually decreasing exponentially. So, Knn can not make good predictions for larger values of p.

(e) If p=1, length=0.1ˆ1=0.1 p=2, length=0.1ˆ(1/2)=0.316 for p=100, length=0.1ˆ(1/100)=0.97 As we can see increasing the features is making the side length move towards 1 which implies concentration of points is more towards boundary of hypercube.

Exercise 6:

(a) P(Y)=$e^{(\text{Beta0}+\text{Beta1}X1+Beta2\text{X2})/1+e}$(Beta0+Beta1$X1$+$Beta2$X2)

P(Y)=$e^{(-6+0.05*40+1*3.5)/1+e}$(-6+0.05$*40+1*$3.5) =0.377

(b) 0.5=$e^{(-6+0.05*?+1*3.5)/1+e}$(-6+0.05$*?+1*$3.5)

Solving this we will get 50 hrs.

Exercise 7:

From Bayes theorem:

Pk(x)=pik$*f(x)/sigma(Pil$f(x))

After substituting the probabilities and f(X) values in the above eqn we will get a probability of 75.2%.

Exercise 9:

(a) odds=p(x)/(1-p(x)) 0.37=p(x)/(1-p(x)) solving this we get, p(x)=0.27

(b) odds=p(x)/(1-p(x)) odds=0.16/1-0.16 odd=0.19 so, she might default by 19%

1.2.Chapter 5:

Exercise 2:

(a) The probabilty of picking any observaton in the bootstrapping is 1/n. The probabilty that 1st bootstrap sample is not jth obesrvation is 1-1/n i.e, (n-1)/n

(b) It is same as (n-1)/n the selection of second sample does not depend as bootstrap model sample with replacement.

(c) Each time the observation not being bootstrapping sample is (1-1/n) so the probability of not chosing sample n times is (1-1/n)^n

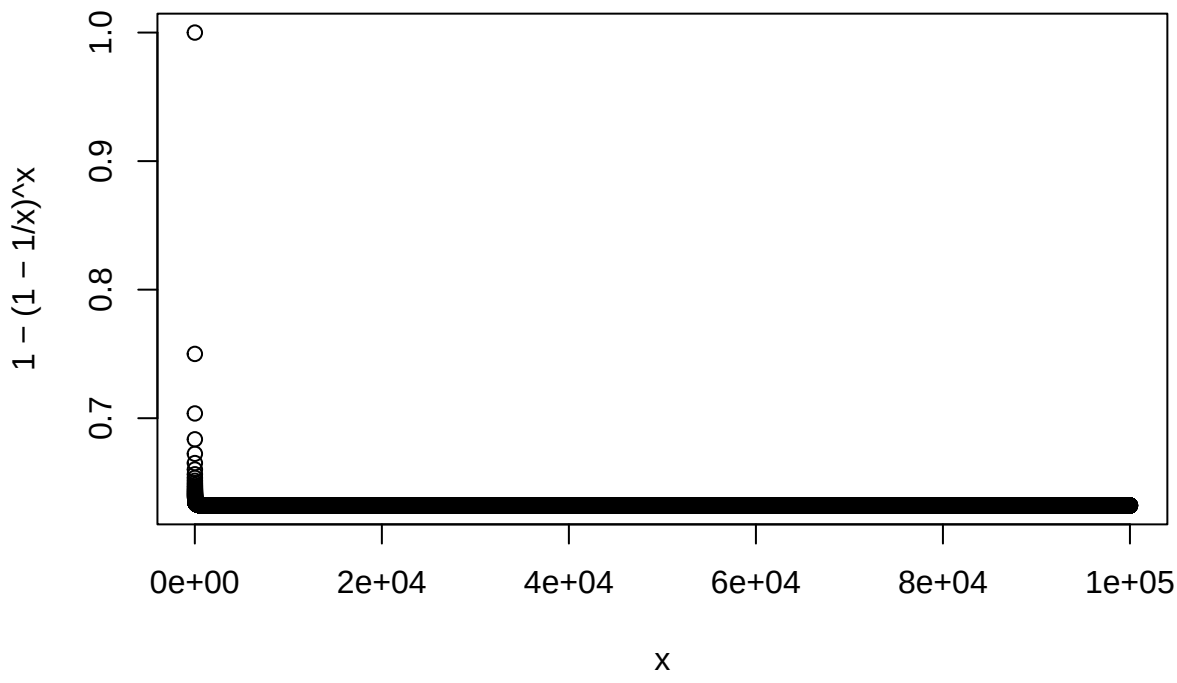
(d) The probability that jth observation is in bootstrap is 1 – (1-1/5)^5 = 0.672

(e) The probability that jth observation is in bootstrap is (1/n)^100 1 –(1-1/100)^100 = 0.634

(f) The probability that jth observation is in bootstrap is 1 –(1-1/10000)^10000 = 0.632

(g) We can see that plot reaches assymptote at 0.632.

```
x <- 1:100000
plot(x, 1 - (1 - 1/x)^x)
```



(h) As n goes to infinity the probability of bootstrap sample containing jth observation will become 1-1/e~0.63

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

```
## [1] 0.625
```

Exercise 3:

a) K fold is a method in which model the data set is split into test and train in the ratio n/k for test set and n(1-1/k) samples for training set. After building models with different hold out sets. Once the evaluation is done we will obtain mean of mean squared errors of the models.

b)

(i) validation test can be easily applied but it has some drawbacks.

1)The test error can be highly varibale, depending on which observations are included in the validation set.

2)Since only a few subsets of data is available in training set, fitting a model with less data results in poor performance so, validation error might over stimate test error.

(ii) LOOCV produces higher variance as the model will be trained on highly correlated data. Moreover it has high computation cost as it needs to do fitting for n samples.

2. Practicum problems

Problem 1:

Reading data and doing some pre processing

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


abalone_reduced=subset(abalone, abalone$V1[]!= "I")

unique(abalone_reduced$V1)
```

```
## [1] "M" "F"
```

```r
abalone_reduced$V1=ifelse(abalone_reduced$V1=="M",1,0)
```

splitting test and train datasets

```r
library(caret)
```

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

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

```
set.seed(0)
trainIndex = createDataPartition(abalone_reduced$V1, p = .8,
                                 list = FALSE)

abaloneTrain=abalone_reduced[trainIndex,]
abaloneTest=abalone_reduced[-trainIndex,]
```

Training the logistic model.

By observing P values of predictors V1, V2 and V3 are very less and significant.

```
logisticModel = glm(V1~., data = abaloneTrain, family = binomial(link = 'logit'))

summary(logisticModel)
```

```
##
## Call:
## glm(formula = V1 ~ ., family = binomial(link = "logit"), data = abaloneTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7992  -1.2093   0.9005   1.1172   1.4017
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.638301   0.513878   5.134 2.83e-07 ***
## V2          -2.041855   2.284242  -0.894  0.37138
## V3          -4.336290   2.686196  -1.614  0.10647
## V4          -2.770398   2.001166  -1.384  0.16624
## V5           0.324599   0.800777   0.405  0.68522
## V6           2.582607   0.972570   2.655  0.00792 **
## V7          -1.604534   1.405278  -1.142  0.25354
## V8          -0.025334   1.230339  -0.021  0.98357
## V9          -0.003675   0.017603  -0.209  0.83462
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3129.2  on 2267  degrees of freedom
## Residual deviance: 3081.8  on 2259  degrees of freedom
## AIC: 3099.8
##
## Number of Fisher Scoring iterations: 4
```

Based on the 95% confidence interval observations, V6 does not have 0 in its range, So we can reject the null hypothesis.

```
confint(logisticModel)
```

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

```
##                   2.5 %     97.5 %
## (Intercept)  1.64472352 3.66064280
## V2          -6.52448536 2.43615314
## V3          -9.61513021 0.92340461
## V4          -7.29365369 0.61202450
## V5          -1.24777189 1.90272550
## V6           0.68020558 4.50076197
## V7          -4.36598105 1.14884506
## V8          -2.44571608 2.38887736
## V9          -0.03820398 0.03085203
```

Building Confidence Interval

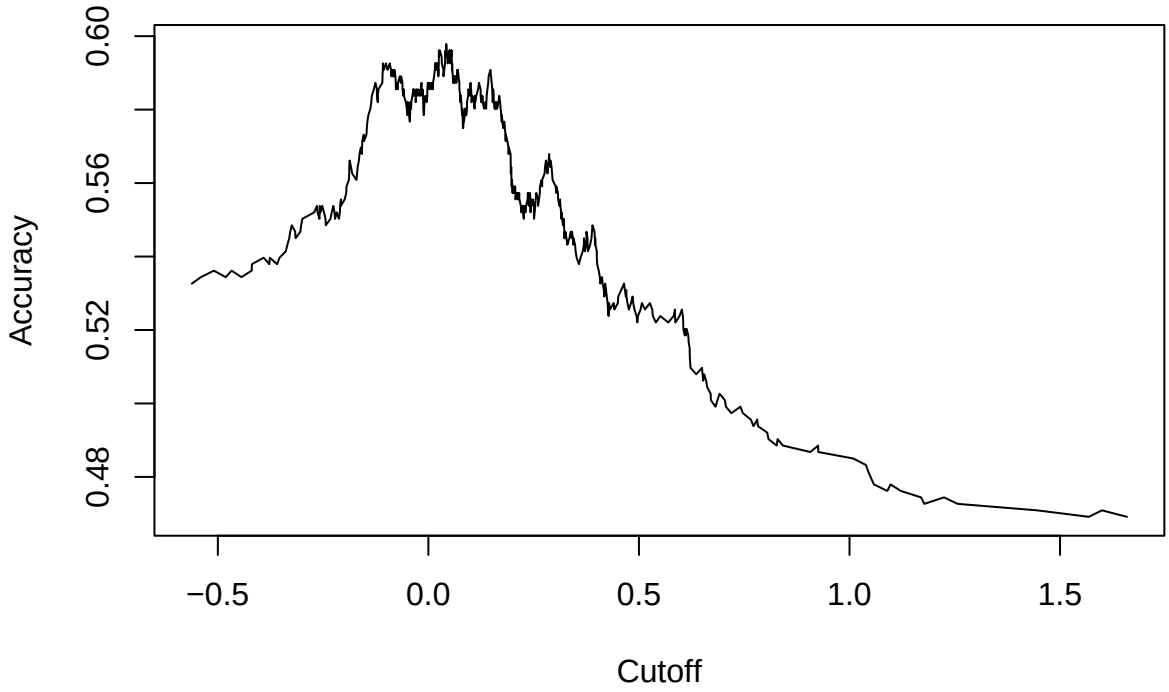
```
#testDataFrame=data.frame(abaloneTest$V2,abaloneTest$V3,abaloneTest$V4,abaloneTest$V5,abaloneTest$V6,ab

#colnames(testDataFrame)=c('V2','V3','V4','V5','V6','V7','V8','V9')

predprob = predict(logisticModel,abaloneTest ,V1="response")


pred=ifelse(predprob>0.5,1,0)



library(caret)

cf=confusionMatrix(as.factor(pred),as.factor(abaloneTest$V1))
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245 249
##          1  20  53
##
##                Accuracy : 0.5256
##                  95% CI : (0.4835, 0.5673)
##     No Information Rate : 0.5326
##     P-Value [Acc > NIR] : 0.6478
##
##                   Kappa : 0.095
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9245
##             Specificity : 0.1755
##          Pos Pred Value : 0.4960
##          Neg Pred Value : 0.7260
##              Prevalence : 0.4674
##          Detection Rate : 0.4321
##    Detection Prevalence : 0.8713
##       Balanced Accuracy : 0.5500
##
```

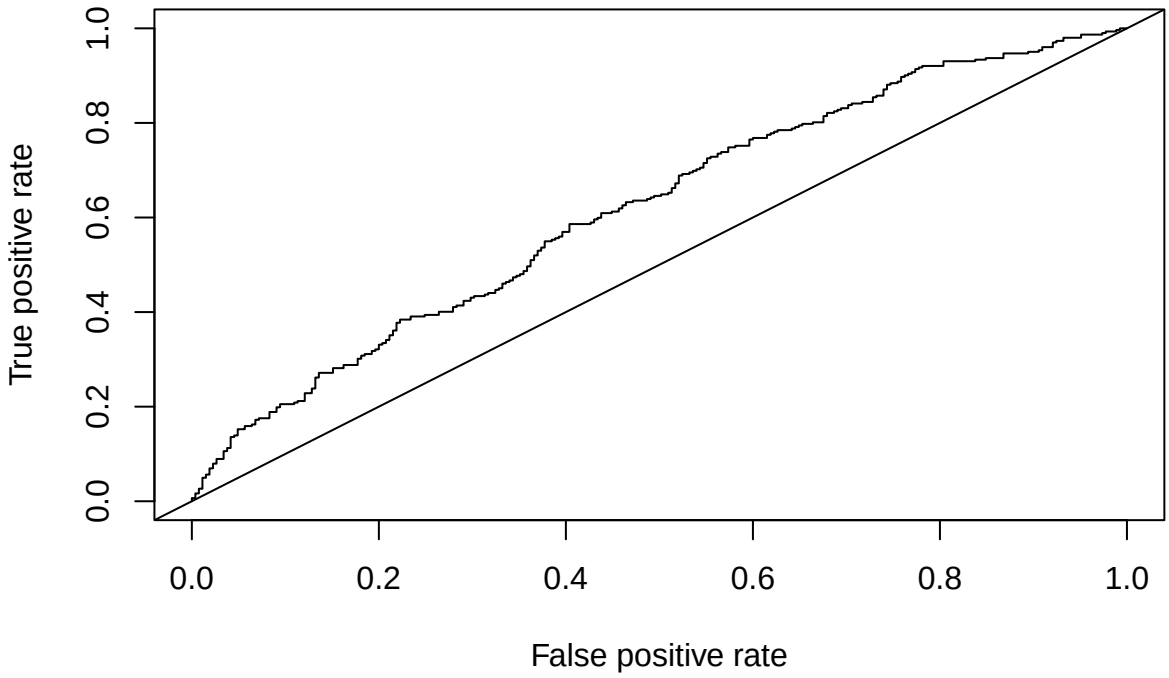
```
##         'Positive' Class : 0
##
```

Plotting ROC curve. From the curve we can observe that the accuracy for cut off value of 0.5 is less compared to other thresholds.

```
library(ROCR)

predic = prediction(predprob,abaloneTest$V1)
eval = performance(predic, "acc")
plot(eval)
```




```
perfROC = performance(predic, measure = "tpr", x.measure = "fpr")
plot(perfROC)
abline(0,1)
```



From the Correlation plot we can observe there is high correlation in predictors. Having high correlation results in poor preformance of the model.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

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

Problem 2:

Loading the Dataset and filling ? in stalk root columns with na and dropping columns with na.

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

names(mushroom) <- c("class","cap.shape","cap.surface","cap.color","bruises","odor","gill.attachment","g
"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","sp
"habitat")

summary(mushroom)
```

```
##     class           cap.shape         cap.surface        cap.color
##  Length:8124        Length:8124        Length:8124        Length:8124
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##    bruises            odor           gill.attachment    gill.spacing
##  Length:8124        Length:8124        Length:8124        Length:8124
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   gill.size          gill.color         stalk.shape        stalk.root
##  Length:8124        Length:8124        Length:8124        Length:8124
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
```

```
##  Length:8124              Length:8124              Length:8124
##  Class :character         Class :character         Class :character
##  Mode  :character         Mode  :character         Mode  :character
##  stalk.color.below.ring  veil.type          veil.color
##  Length:8124            Length:8124        Length:8124
##  Class :character       Class :character   Class :character
##  Mode  :character       Mode  :character   Mode  :character
##  ring.number         ring.type         spore.print.color   population
##  Length:8124        Length:8124        Length:8124        Length:8124
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##    habitat
##  Length:8124
##  Class :character
##  Mode  :character
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mushroom <- mutate(mushroom,stalk.root=ifelse(stalk.root=='?',NA,stalk.root))
table(is.na(mushroom))
```

```
##
##  FALSE   TRUE
## 184372   2480
```

```
library(tidyr)
colSums(is.na(mushroom))
```

```
##                     class                 cap.shape               cap.surface
##                         0                         0                         0
##                cap.color                   bruises                     odor
##                         0                         0                         0
##          gill.attachment             gill.spacing                gill.size
##                         0                         0                         0
##                gill.color               stalk.shape               stalk.root
##                         0                         0                      2480
## stalk.surface.above.ring stalk.surface.below.ring   stalk.color.above.ring
##                         0                         0                         0
##   stalk.color.below.ring                 veil.type               veil.color
##                         0                         0                         0
```

```
##              ring.number                ring.type        spore.print.color
##                        0                        0                        0
##               population                  habitat
##                        0                        0
```

```
mushroom=drop_na(mushroom)
colSums(is.na(mushroom))
```

```
##                    class                cap.shape              cap.surface
##                        0                        0                        0
##                cap.color                  bruises                     odor
##                        0                        0                        0
##          gill.attachment             gill.spacing                gill.size
##                        0                        0                        0
##                gill.color              stalk.shape                stalk.root
##                        0                        0                        0
## stalk.surface.above.ring stalk.surface.below.ring   stalk.color.above.ring
##                        0                        0                        0
##   stalk.color.below.ring                veil.type               veil.color
##                        0                        0                        0
##              ring.number                ring.type        spore.print.color
##                        0                        0                        0
##               population                  habitat
##                        0                        0
```

Splitting Train and Test Data.

```
library(caret)
set.seed(0)
trainIndex = sample(1:nrow(mushroom),size=0.8*nrow(mushroom))

mushroomTrain=mushroom[trainIndex,]
mushroomTest=mushroom[-trainIndex,]
```

Training Classifier.

From the Confusion Matrix we can see the train accuracy as 95.55 %.The false Positive value for test set is 189.

```
library(e1071)

Classifier <- naiveBayes(class ~ ., data = mushroomTrain)

y_pred_train <- predict(Classifier, newdata = mushroomTrain[,-1])

cf=confusionMatrix(table(y_pred_train,mushroomTrain$class))
cf
```

```
## Confusion Matrix and Statistics
##
##
## y_pred_train    e    p
##            e 2771  189
```

```
##            p   12 1543
##
##                Accuracy : 0.9555
##                  95% CI : (0.9491, 0.9613)
##     No Information Rate : 0.6164
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.904
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9957
##             Specificity : 0.8909
##          Pos Pred Value : 0.9361
##          Neg Pred Value : 0.9923
##              Prevalence : 0.6164
##          Detection Rate : 0.6137
##    Detection Prevalence : 0.6556
##       Balanced Accuracy : 0.9433
##
##        'Positive' Class : e
##
```

```
y_pred_train=data.frame(y_pred_train)
```

From the Confusion Matrix we can see the test accuracy as 95.66 %. The false Positive value for test set is 45.

```
y_pred_test <- predict(Classifier, newdata = mushroomTest[,-1])


cf=confusionMatrix(table(y_pred_test,mushroomTest$class))
cf
```

```
## Confusion Matrix and Statistics
##
##
## y_pred_test   e   p
##           e 701  45
##           p   4 379
##
##                Accuracy : 0.9566
##                  95% CI : (0.943, 0.9677)
##     No Information Rate : 0.6244
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9056
##
##  Mcnemar's Test P-Value : 1.102e-08
##
##             Sensitivity : 0.9943
##             Specificity : 0.8939
##          Pos Pred Value : 0.9397
```

```
##          Neg Pred Value : 0.9896
##              Prevalence : 0.6244
##          Detection Rate : 0.6209
##    Detection Prevalence : 0.6608
##       Balanced Accuracy : 0.9441
##
##        'Positive' Class : e
##
```

Problem 3:

Loading data set and splitting data

```
yacht_hydrodynamics = read.table(url("https://archive.ics.uci.edu/ml/machine-learning-databases/00243/ya

names(yacht_hydrodynamics) <- c("Longitudinal","Prismatic_coefficient", "Length_displacement_ratio","Bea


library(caret)


set.seed(0)
trainIndex = createDataPartition(yacht_hydrodynamics$Residuary_resistance, p = .8,
                                 list = FALSE)

yachtTrain=yacht_hydrodynamics[trainIndex,]
yachtTest=yacht_hydrodynamics[-trainIndex,]
```

Building the model. From the summary R2 is 0.6573 and the RMSE is calculated as 8.693.

```
Model=lm(Residuary_resistance~.,data=yachtTrain)

summary(Model)
```

```
##
## Call:
## lm(formula = Residuary_resistance ~ ., data = yachtTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.587  -7.281  -1.769   6.007  31.921
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -11.9684    30.0432  -0.398    0.691
## Longitudinal                0.2281     0.3649   0.625    0.532
## Prismatic_coefficient     -26.3190    48.9174  -0.538    0.591
## Length_displacement_ratio  -2.7788    15.5065  -0.179    0.858
## Beam_draught_ratio          1.2189     6.0686   0.201    0.841
## Length_beam_ratio           3.6274    15.5090   0.234    0.815
## Froude_number             120.3319     5.6080  21.457   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 8.819 on 241 degrees of freedom
## Multiple R-squared:  0.6573, Adjusted R-squared:  0.6487
## F-statistic: 77.02 on 6 and 241 DF,  p-value: < 2.2e-16
```

```
trainDataFrame=data.frame(yachtTrain$Longitudinal,yachtTrain$Prismatic_coefficient,yachtTrain$Length_dis


colnames(trainDataFrame)=c('Longitudinal','Prismatic_coefficient','Length_displacement_ratio','Beam_drau


predicted=predict(Model,trainDataFrame)

library(Metrics)
```

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

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

```
rmse(yachtTrain$Residuary_resistance, predicted)
```

```
## [1] 8.693721
```

Performing Bootstrapping

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

model_boot <- train(Residuary_resistance~.,data=yachtTrain, method = "lm",
               trControl = train.control)

print(model_boot)
```

```
## Linear Regression
##
## 248 samples
##   6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (1000 reps)
## Summary of sample sizes: 248, 248, 248, 248, 248, 248, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   9.08499  0.6367714  7.250893
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Calculating RMSE and R squared values for bootstrap models and RMSE plot.

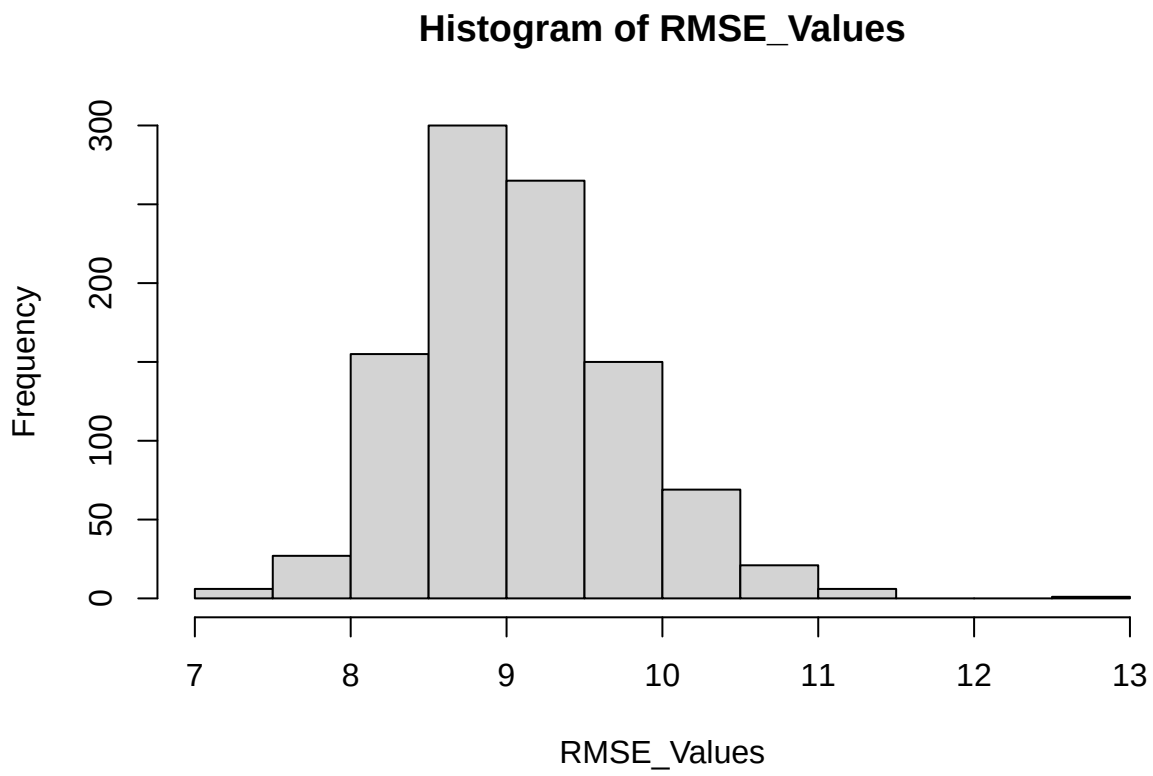
```
RMSE_Values=model_boot$resample["RMSE"]$RMSE
Squared_values=model_boot$resample["Rsquared"]$Rsquared


mean(RMSE_Values)
```

```
## [1] 9.08499
```

```
mean(Squared_values)
```

```
## [1] 0.6367714
```

```
hist(RMSE_Values)
```



From the above values we can see that for test set, the values are almost same for both basic and bootstrap models.

Problem 4:

Loading Data set and splitting data to test and train sets.

```
germanCredit = read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/ger

germanCredit$V25<-factor(germanCredit$V25)
```

```
library(caret)
set.seed(0)
trainIndex = createDataPartition(germanCredit$V25, p =.8,
                                 list = FALSE)

creditTrain=germanCredit[trainIndex,]
creditTest=germanCredit[-trainIndex,]
```

Building logistic model

```
Model=glm(V25~ ., data = creditTrain, family = binomial(link = 'logit'))
```

Building confusion matrix for test and train sets for basic model.

From the confusion matrix we can see Precisionm, Recall and F1 for test and train set.

Test Values:

Precision : 0.7457
Recall : 0.9214
F1 : 0.8243

Train Values:

Recall : 0.9429
F1 : 0.8462
Prevalence : 0.7000

```
predprob = predict(Model,creditTrain)

pred=ifelse(predprob>0.5,2,1)



library(caret)

print("Confusion Matrix for Train set of Basic Model")
```

```
## [1] "Confusion Matrix for Train set of Basic Model"
```

```
cf=confusionMatrix(as.factor(pred),as.factor(creditTrain$V25), mode = "everything",positive="1")
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 528 160
##          2  32  80
##
##                Accuracy : 0.76
##                  95% CI : (0.7289, 0.7892)
##     No Information Rate : 0.7
```

```
##     P-Value [Acc > NIR] : 9.295e-05
##
##                   Kappa : 0.3258
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9429
##             Specificity : 0.3333
##          Pos Pred Value : 0.7674
##          Neg Pred Value : 0.7143
##               Precision : 0.7674
##                  Recall : 0.9429
##                      F1 : 0.8462
##              Prevalence : 0.7000
##          Detection Rate : 0.6600
##    Detection Prevalence : 0.8600
##       Balanced Accuracy : 0.6381
##
##        'Positive' Class : 1
##
```

```
predprob = predict(Model,creditTest)

pred=ifelse(predprob>0.5,2,1)



library(caret)

print("Confusion Matrix for Test set of Basic Model")
```

```
## [1] "Confusion Matrix for Test set of Basic Model"
```

```
cf=confusionMatrix(as.factor(pred),as.factor(creditTest$V25), mode = "everything",positive="1")
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 129  44
##          2  11  16
##
##                Accuracy : 0.725
##                  95% CI : (0.6576, 0.7856)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.2455
##
##                   Kappa : 0.2232
##
##  Mcnemar's Test P-Value : 1.597e-05
##
##             Sensitivity : 0.9214
```

```
##             Specificity : 0.2667
##          Pos Pred Value : 0.7457
##          Neg Pred Value : 0.5926
##               Precision : 0.7457
##                  Recall : 0.9214
##                      F1 : 0.8243
##              Prevalence : 0.7000
##          Detection Rate : 0.6450
##    Detection Prevalence : 0.8650
##       Balanced Accuracy : 0.5940
##
##        'Positive' Class : 1
##
```

Building confusion matrix for test and train sets for cv models.

From the confusion matrix we can see Precision, Recall and F1 for test and train sets.

Test Values:

Precision : 0.7457
Recall : 0.9214
F1 : 0.8243

Train Values:

Precision : 0.7674
Recall : 0.9429
F1 : 0.8462

```
ctrl<-trainControl(method="cv",number=10)
CrossVadlid_Modl<-train(V25~.,data=creditTrain,trControl=ctrl,method="glm")
CrossVadlid_Modl<-CrossVadlid_Modl$finalModel
pred_train<-predict(CrossVadlid_Modl,creditTrain)

pred=ifelse(pred_train>0.5,2,1)

library(caret)
print("Confusion Matrix for Train set of cv Model")
```

```
## [1] "Confusion Matrix for Train set of cv Model"
```

```
cf=confusionMatrix(as.factor(pred),as.factor(creditTrain$V25), mode = "everything",positive="1")
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 528 160
##          2  32  80
##
##                Accuracy : 0.76
##                  95% CI : (0.7289, 0.7892)
##     No Information Rate : 0.7
```

```
##     P-Value [Acc > NIR] : 9.295e-05
##
##                   Kappa : 0.3258
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9429
##             Specificity : 0.3333
##          Pos Pred Value : 0.7674
##          Neg Pred Value : 0.7143
##               Precision : 0.7674
##                  Recall : 0.9429
##                      F1 : 0.8462
##              Prevalence : 0.7000
##          Detection Rate : 0.6600
##    Detection Prevalence : 0.8600
##       Balanced Accuracy : 0.6381
##
##        'Positive' Class : 1
##
```

```
predprob = predict(CrossVadlid_Modl,creditTest)

pred=ifelse(predprob>0.5,2,1)



library(caret)

print("Confusion Matrix for Test set of CV Model")
```

```
## [1] "Confusion Matrix for Test set of CV Model"
```

```
cf=confusionMatrix(as.factor(pred),as.factor(creditTest$V25), mode = "everything",positive="1")
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 129  44
##          2  11  16
##
##                Accuracy : 0.725
##                  95% CI : (0.6576, 0.7856)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.2455
##
##                   Kappa : 0.2232
##
##  Mcnemar's Test P-Value : 1.597e-05
##
##             Sensitivity : 0.9214
```

```
##             Specificity : 0.2667
##          Pos Pred Value : 0.7457
##          Neg Pred Value : 0.5926
##               Precision : 0.7457
##                  Recall : 0.9214
##                      F1 : 0.8243
##              Prevalence : 0.7000
##          Detection Rate : 0.6450
##    Detection Prevalence : 0.8650
##       Balanced Accuracy : 0.5940
##
##        'Positive' Class : 1
##
```

After looking at values of F1, Precision and recall we can see that these are almost same for both basic model and cv model.