Assignment 4.3

Suraj Bhattarai

2023-07-04

1. Read the titanic.csv data with base R function and save it as "data" and remove the name column and save again as data

```
setwd("D:/r_programs_stats/exam pracice/second_assement")
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
data <- read.csv("D:/r_programs_stats/statitistics/titanic.csv")
# remove the name column
#data <- subset(data, select = -c(Name))
data <- data[,-3]
str(data)
```

```
## 'data.frame':    887 obs. of  7 variables:
##  $ Survived               : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass                 : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex                    : chr  "male" "female" "female" "female" ...
##  $ Age                    : num  22 38 26 35 35 27 54 2 27 14 ...
##  $ Siblings.Spouses.Aboard: int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parents.Children.Aboard: int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Fare                   : num  7.25 71.28 7.92 53.1 8.05 ...
```

##2. Fit binary logistic regression model with "Survived" variable as dependent variable and rest of variables as independent variables using "data", get summary of the model, check VIF and interpret the results carefully

```
data$Survived <- as.factor(data$Survived)
# let us change it as factor variable
data$Pclass <- as.factor(data$Pclass)
str(data$Pclass)
```

```
##  Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
```

```
# let us do the same for the sex variable as well
data$Sex <- as.factor(data$Sex)
str(data$Sex)
```

```
##  Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
```

```r
model.full <- glm(Survived~., data= data, family = binomial)
summary(model.full)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7773  -0.5991  -0.3984   0.6131   2.4412
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              4.109777   0.463602   8.865  < 2e-16 ***
## Pclass2                 -1.161491   0.300960  -3.859 0.000114 ***
## Pclass3                 -2.350022   0.304666  -7.713 1.22e-14 ***
## Sexmale                 -2.756710   0.200642 -13.739  < 2e-16 ***
## Age                     -0.043410   0.007790  -5.573 2.51e-08 ***
## Siblings.Spouses.Aboard -0.401572   0.110795  -3.624 0.000290 ***
## Parents.Children.Aboard -0.106884   0.118767  -0.900 0.368151
## Fare                     0.002823   0.002468   1.144 0.252771
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1182.77  on 886  degrees of freedom
## Residual deviance:  780.93  on 879  degrees of freedom
## AIC: 796.93
##
## Number of Fisher Scoring iterations: 5
```

```r
library(car)
```

```
## Loading required package: carData
```

```r
vif(model.full)
```

```
##                             GVIF Df GVIF^(1/(2*Df))
## Pclass                  2.041787  2        1.195371
## Sex                     1.201233  1        1.096008
## Age                     1.477422  1        1.215492
## Siblings.Spouses.Aboard 1.290358  1        1.135939
## Parents.Children.Aboard 1.267656  1        1.125902
## Fare                    1.578965  1        1.256569
```

```r
# Here all the features except for`Parents.Children.Aboard` and Fare seem to string predictors for the 
```

3. Randomly split the data into 70% and 30% with replacement of samples as "train" and "test" data

```
set.seed(38)
# data partition
ind <- sample(2,nrow(data), replace = T, prob = c(0.7,0.3))
train <- data[ind==1,]
test <- data[ind==2,]
```

5 Fit binary logistic regression classifier, knn classifier, ann classifier, naive bayes classifier, svm classifier, decision tree classifier, decision tree bagging classifier, random forest classifier, tuned random forest classifier and random forest boosting classifier models using the "train" data

```
#  binary logistic regression classifier
model.full <- glm(Survived~., data= train, family = binomial)

# knn classifier
library('caret')
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
model.knn <- train(Survived~., data = train, method = "knn")

# ann classifie
library(nnet)
nn_model <- nnet(Survived~., data = train, size = 5, linear.output= TRUE)
```

```
## # weights:  46
## initial  value 520.254125
## iter  10 value 386.168454
## iter  20 value 350.476083
## iter  30 value 332.145866
## iter  40 value 313.812039
## iter  50 value 296.617007
## iter  60 value 276.376654
## iter  70 value 264.904892
## iter  80 value 263.491628
## iter  90 value 254.557869
## iter 100 value 248.835823
## final  value 248.835823
## stopped after 100 iterations
```

```
# naive bayes classifier
library(e1071)
model.nb <- naiveBayes(Survived~., data = train)
```

```r
#  svm classifier
library('e1071')
model.svm <- svm(formula = Survived~., data= train,
                 type= 'C-classification',
                 kernel = 'linear')

#decision tree classifier
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:car':
##
##     Predict

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```r
tree <- ctree(Survived~., data= train)

#decision tree bagging classifier
library(ipred)
MBTree <- bagging(Survived~., data = train, coob= T)

#random forest classifier
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

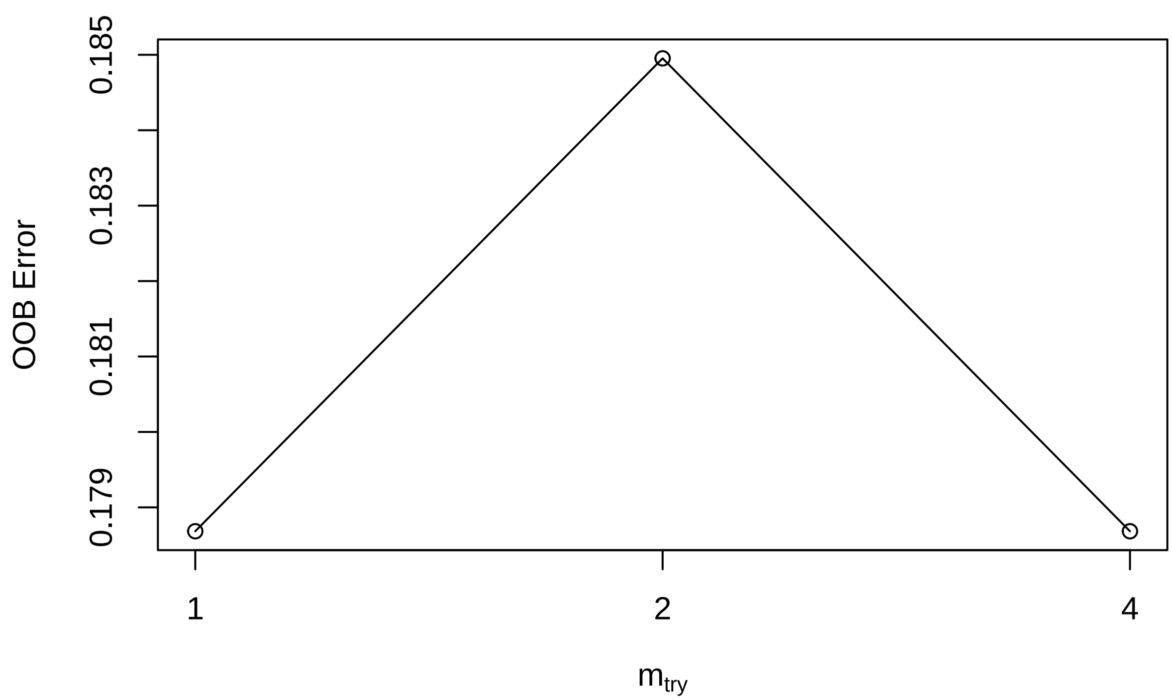
```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(38)
rfm <- randomForest(Survived~., data= train)


#tuned random forest classifier
t <- tuneRF(train[,-1],train[,1],
            stepFactor = 0.5,
            plot = T,
            ntreeTry = 300,
            trace = T,
            improve = 0.05)
```

```
## mtry = 2  OOB error = 18.5%
## Searching left ...
## mtry = 4     OOB error = 17.87%
## 0.03389831 0.05
## Searching right ...
## mtry = 1     OOB error = 17.87%
## 0.03389831 0.05
```



```r
# improve "rfm" model
rfm1 <- randomForest(Survived~., data= train, ntree= 300, mtry = 4,
                     improtance = T, proximity= T)

#random forest boosting classifier models
library(caret)
mod.gbm <- train(Survived~., data= train, method = "gbm",verbose = F)
```

##5. Get confusion matrix and accuracy/misclassification error for all the classifier models and interpret them carefully

```r
#  binary logistic regression classifier
predict.test <- predict(model.full,test, type= "response")
predicted.test <- factor(ifelse(predict.test>0.5,1,0))
reference.test <- factor(test$Survived)
(Bcm <- table(predicted.test, reference.test))
```

```
##                 reference.test
## predicted.test   0   1
##              0 141  30
##              1  22  56
```

```r
(Baccuracy <- sum(diag(Bcm))/sum(Bcm))
```

```
## [1] 0.7911647
```

```r
(Berror <- 1 -Baccuracy)
```

```
## [1] 0.2088353
```

```r
# knn classifier
kpredict.test <- predict(model.knn, test)
(kcm <- table(kpredict.test, test$Survived))
```

```
##
## kpredict.test   0   1
##             0 136  40
##             1  27  46
```

```r
(kaccuracy <- sum(diag(kcm))/sum(kcm))
```

```
## [1] 0.7309237
```

```r
(kerror <- 1 -Baccuracy)
```

```
## [1] 0.2088353
```

```r
# ann classifier
Apredict.test <- predict(nn_model, test)
Apredicted.test <- factor(ifelse(predict.test>0.5,1,0))
Areference.test <- factor(test$Survived)
(Acm <- table(Apredicted.test, Areference.test))
```

```
##                 Areference.test
## Apredicted.test   0   1
##               0 141  30
##               1  22  56
```

```r
(Aaccuracy <- sum(diag(Acm))/sum(Acm))
```

```
## [1] 0.7911647
```

```r
(Aerror <- 1 -Aaccuracy)
```

```
## [1] 0.2088353
```

```r
# naive bayes classifier
Npredict.test <- predict(model.nb, test)
(Ncm <- table(Npredict.test, test$Survived))
```

```
##
## Npredict.test   0   1
##             0 140  27
##             1  23  59
```

```r
(Naccuracy <- sum(diag(Ncm))/sum(Ncm))
```

```
## [1] 0.7991968
```

```r
(Nerror <- 1 -Naccuracy)
```

```
## [1] 0.2008032
```

```r
# svm classifier
Spredict.test <- predict(model.svm, test)
(Scm <- table(Spredict.test, test$Survived))
```

```
##
## Spredict.test   0   1
##             0 139  29
##             1  24  57
```

```r
(Saccuracy <- sum(diag(Scm))/sum(Scm))
```

```
## [1] 0.7871486
```

```r
(Serror <- 1 -Saccuracy)
```

```
## [1] 0.2128514
```

```r
# Decision tree
Dpredict.test <- predict(tree, test)
(Dcm <- table(Dpredict.test, test$Survived))
```

```
##
## Dpredict.test   0   1
##             0 138  25
##             1  25  61
```

```r
(Daccuracy <- sum(diag(Dcm))/sum(Dcm))
```

```
## [1] 0.7991968
```

```r
(Derror <- 1 -Daccuracy)
```

```
## [1] 0.2008032
```

```r
# decision tree bagging classifier
Mpredict.test <- predict(MBTree, test)
(Mcm <- table(Mpredict.test, test$Survived))
```

```
##
## Mpredict.test   0   1
##             0 135  26
##             1  28  60
```

```r
(Maccuracy <- sum(diag(Mcm))/sum(Mcm))
```

```
## [1] 0.7831325
```

```r
(Merror <- 1 -Maccuracy)
```

```
## [1] 0.2168675
```

```r
# random forest classifier
Rpredict.test <- predict(rfm, test)
(Rcm <- table(Rpredict.test, test$Survived))
```

```
##
## Rpredict.test   0   1
##             0 150  25
##             1  13  61
```

```r
(Raccuracy <- sum(diag(Rcm))/sum(Rcm))
```

```
## [1] 0.8473896
```

```r
(Rerror <- 1 - Raccuracy)
```

```
## [1] 0.1526104
```

```r
# tuned random forest classifier
Rtpredict.test <- predict(rfm1, test)
(Rtcm <- table(Rtpredict.test, test$Survived))
```

```
##
## Rtpredict.test   0   1
##              0 144  25
##              1  19  61
```

```r
(Rtaccuracy <- sum(diag(Rtcm))/sum(Rtcm))
```

```
## [1] 0.8232932
```

```r
(Rterror <- 1 - Rtaccuracy)
```

```
## [1] 0.1767068
```

```r
#random forest boosting classifier
Fpredict.test <- predict(mod.gbm, test)
(Fcm <- table(Fpredict.test, test$Survived))
```

```
##
## Fpredict.test   0   1
##             0 147  26
##             1  16  60
```

```r
(Faccuracy <- sum(diag(Fcm))/sum(Fcm))
```

```
## [1] 0.8313253
```

```r
(Ferror <- 1 - Faccuracy)
```

```
## [1] 0.1686747
```

##6. Get confusion matrix and accuracy/misclassification error for all the predicted models and interpret them carefully

```r
# binary logistic regression classifier
predict.test <- predict(model.full,test, type= "response")
predicted.test <- factor(ifelse(predict.test>0.5,1,0))
reference.test <- factor(test$Survived)
(Bcm <- table(predicted.test, reference.test))
```

```
##               reference.test
## predicted.test   0   1
##              0 141  30
##              1  22  56
```

```r
confusionMatrix(Bcm)
```

```
## Confusion Matrix and Statistics
##
##               reference.test
## predicted.test   0   1
##              0 141  30
##              1  22  56
##
##                Accuracy : 0.7912
##                  95% CI : (0.7353, 0.8399)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 1.721e-06
##
##                   Kappa : 0.5278
##
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.8650
##             Specificity : 0.6512
##          Pos Pred Value : 0.8246
##          Neg Pred Value : 0.7179
##              Prevalence : 0.6546
##          Detection Rate : 0.5663
##    Detection Prevalence : 0.6867
##       Balanced Accuracy : 0.7581
##
##        'Positive' Class : 0
##
```

```r
# knn classifier
kpredict.test <- predict(model.knn, test)
(kcm <- table(kpredict.test, test$Survived))
```

```
##
## kpredict.test   0   1
##             0 136  39
##             1  27  47
```

```r
confusionMatrix(kcm)
```

```
## Confusion Matrix and Statistics
##
##
## kpredict.test   0   1
##             0 136  39
##             1  27  47
##
##                Accuracy : 0.7349
##                  95% CI : (0.6755, 0.7887)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 0.00407
##
##                   Kappa : 0.3938
##
##  Mcnemar's Test P-Value : 0.17573
##
##             Sensitivity : 0.8344
##             Specificity : 0.5465
##          Pos Pred Value : 0.7771
##          Neg Pred Value : 0.6351
##              Prevalence : 0.6546
##          Detection Rate : 0.5462
##    Detection Prevalence : 0.7028
##       Balanced Accuracy : 0.6904
##
##        'Positive' Class : 0
##
```

```r
# ann classifier
Apredict.test <- predict(nn_model, test)
Apredicted.test <- factor(ifelse(predict.test>0.5,1,0))
Areference.test <- factor(test$Survived)
Acm <- table(Apredicted.test, Areference.test)
confusionMatrix(kcm)
```


```
## Confusion Matrix and Statistics
##
##
## kpredict.test   0   1
##             0 136  39
##             1  27  47
##
##                Accuracy : 0.7349
##                  95% CI : (0.6755, 0.7887)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 0.00407
##
##                   Kappa : 0.3938
##
##  Mcnemar's Test P-Value : 0.17573
##
```

```
##             Sensitivity : 0.8344
##             Specificity : 0.5465
##          Pos Pred Value : 0.7771
##          Neg Pred Value : 0.6351
##              Prevalence : 0.6546
##          Detection Rate : 0.5462
##    Detection Prevalence : 0.7028
##       Balanced Accuracy : 0.6904
##
##        'Positive' Class : 0
##
```

```
# naive bayes classifier
Npredict.test <- predict(model.nb, test)
Ncm <- table(Npredict.test, test$Survived)
confusionMatrix(Ncm)
```

```
## Confusion Matrix and Statistics
##
##
## Npredict.test   0   1
##             0 140  27
##             1  23  59
##
##                Accuracy : 0.7992
##                  95% CI : (0.744, 0.8471)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 3.983e-07
##
##                   Kappa : 0.551
##
##  Mcnemar's Test P-Value : 0.6714
##
##             Sensitivity : 0.8589
##             Specificity : 0.6860
##          Pos Pred Value : 0.8383
##          Neg Pred Value : 0.7195
##              Prevalence : 0.6546
##          Detection Rate : 0.5622
##    Detection Prevalence : 0.6707
##       Balanced Accuracy : 0.7725
##
##        'Positive' Class : 0
##
```

```
# svm classifier
Spredict.test <- predict(model.svm, test)
Scm <- table(Spredict.test, test$Survived)
confusionMatrix(Scm)
```

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

```
## Spredict.test   0   1
##             0 139  29
##             1  24  57
##
##                Accuracy : 0.7871
##                  95% CI : (0.731, 0.8363)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 3.452e-06
##
##                   Kappa : 0.5227
##
##  Mcnemar's Test P-Value : 0.5827
##
##             Sensitivity : 0.8528
##             Specificity : 0.6628
##          Pos Pred Value : 0.8274
##          Neg Pred Value : 0.7037
##              Prevalence : 0.6546
##          Detection Rate : 0.5582
##    Detection Prevalence : 0.6747
##       Balanced Accuracy : 0.7578
##
##        'Positive' Class : 0
##
```

```r
# Decision tree
Dpredict.test <- predict(tree, test)
Dcm <- table(Dpredict.test, test$Survived)
confusionMatrix(Dcm)
```

```
## Confusion Matrix and Statistics
##
##
## Dpredict.test   0   1
##             0 138  25
##             1  25  61
##
##                Accuracy : 0.7992
##                  95% CI : (0.744, 0.8471)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 3.983e-07
##
##                   Kappa : 0.5559
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8466
##             Specificity : 0.7093
##          Pos Pred Value : 0.8466
##          Neg Pred Value : 0.7093
##              Prevalence : 0.6546
##          Detection Rate : 0.5542
##    Detection Prevalence : 0.6546
##       Balanced Accuracy : 0.7780
```

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

```r
# decision tree bagging classifier
Mpredict.test <- predict(MBTree, test)
Mcm <- table(Mpredict.test, test$Survived)
confusionMatrix(Mcm)
```

```
## Confusion Matrix and Statistics
##
##
## Mpredict.test   0   1
##             0 135  26
##             1  28  60
##
##                Accuracy : 0.7831
##                  95% CI : (0.7267, 0.8327)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 6.768e-06
##
##                   Kappa : 0.523
##
##  Mcnemar's Test P-Value : 0.8918
##
##             Sensitivity : 0.8282
##             Specificity : 0.6977
##          Pos Pred Value : 0.8385
##          Neg Pred Value : 0.6818
##              Prevalence : 0.6546
##          Detection Rate : 0.5422
##    Detection Prevalence : 0.6466
##       Balanced Accuracy : 0.7629
##
##        'Positive' Class : 0
##
```

```r
# random forest classifier
Rpredict.test <- predict(rfm, test)
Rcm <- table(Rpredict.test, test$Survived)
confusionMatrix(Rcm)
```

```
## Confusion Matrix and Statistics
##
##
## Rpredict.test   0   1
##             0 150  25
##             1  13  61
##
##                Accuracy : 0.8474
##                  95% CI : (0.7966, 0.8897)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 7.08e-12
```

```
##
##                   Kappa : 0.651
##
##  Mcnemar's Test P-Value : 0.07435
##
##             Sensitivity : 0.9202
##             Specificity : 0.7093
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8243
##              Prevalence : 0.6546
##          Detection Rate : 0.6024
##    Detection Prevalence : 0.7028
##       Balanced Accuracy : 0.8148
##
##        'Positive' Class : 0
##
```

```r
# tuned random forest classifier
Rtpredict.test <- predict(rfm1, test)
(Rtcm <- table(Rtpredict.test, test$Survived))
```

```
##
## Rtpredict.test   0   1
##              0 144  25
##              1  19  61
```

```r
confusionMatrix(Rtcm)
```

```
## Confusion Matrix and Statistics
##
##
## Rtpredict.test   0   1
##              0 144  25
##              1  19  61
##
##                Accuracy : 0.8233
##                  95% CI : (0.7701, 0.8686)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 2.727e-09
##
##                   Kappa : 0.6027
##
##  Mcnemar's Test P-Value : 0.451
##
##             Sensitivity : 0.8834
##             Specificity : 0.7093
##          Pos Pred Value : 0.8521
##          Neg Pred Value : 0.7625
##              Prevalence : 0.6546
##          Detection Rate : 0.5783
##    Detection Prevalence : 0.6787
##       Balanced Accuracy : 0.7964
##
```

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

```
#random forest boosting classifier
Fpredict.test <- predict(mod.gbm, test)
Fcm <- table(Fpredict.test, test$Survived)
confusionMatrix(Fcm)
```

```
## Confusion Matrix and Statistics
##
##
## Fpredict.test   0   1
##             0 147  26
##             1  16  60
##
##                Accuracy : 0.8313
##                  95% CI : (0.7789, 0.8756)
##     No Information Rate : 0.6546
##     P-Value [Acc > NIR] : 4.198e-10
##
##                   Kappa : 0.6164
##
##  Mcnemar's Test P-Value : 0.1649
##
##             Sensitivity : 0.9018
##             Specificity : 0.6977
##          Pos Pred Value : 0.8497
##          Neg Pred Value : 0.7895
##              Prevalence : 0.6546
##          Detection Rate : 0.5904
##    Detection Prevalence : 0.6948
##       Balanced Accuracy : 0.7998
##
##        'Positive' Class : 0
##
```

7. Compare accuracy and misclassification error of predicted models based on "test" data to decide the "best"model

ans:Comparing all the model, I found that random forest boosting classifier have higher accuracy and less misclassification error. so, i prefer to recommend randomforest classifier for this data from the rest of all models.

##8. Write a reflection on your own word focusing on "what did I learn from this assignment?" #ans: few things i learned from this assigment are # - to check vif for Multicollinearity (ie: ommit attributes whose vif grater than 10) # - to use different classificattion model # - to find out accuracy and misclassification error #- used of different model and perfomance measure of model

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.