Business Analytics 2nd edition Jaggia Solutions Manual

SKU: 9781264302802-SOLUTIONS

Chapter 12. Supervised Data Mining: k-Nearest Neighbors and Naïve Bayes

Solutions (Note: all calculations are made using unrounded values.)

12_1. <Analytic Solver>

a. The optimal vale of k is 2.

b. Cutoff Value = .50 (Using the test data set)
- Accuracy rate = 83.33%
- Specificity = 0.7778
- Sensitivity = 1
- Precision = 0.6

12_1. <R>

a. The optimal value of k is 9.

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.9130
- Specificity = 1
- Sensitivity = 0.8
- Precision = 1

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:3])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[3] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
```

```
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
```

12_2. <Analytic Solver>

a. The optimal value of k is 5.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 50%
   - Specificity = 0.3846
   - Sensitivity = 0.6667
   - Precision = 0.4286

c. The performance measures in b suggest that the performance of the KNN classifier is mediocre at best. The accuracy rate suggests that only half of the cases in the test set are classified correctly by the model. Given that there are 9 target class (1) cases out of the 22 test data cases, the naïve rule (classifying all cases into the predominant class) would have an accuracy rate of (22-9)/22 = 0.5909. Although sensitivity suggests that 66.67% of the target class cases are classified correctly, specificity shows that the great majority of the non-target class classes are misclassified. Both the lift chart and ROC curve suggest that the KNN classifier performs about the same as the baseline model. The AUC value is 0.5085, which is very close to the baseline model (AUC = 0.5).

12_2. <R>

a. The optimal value of k is 4.

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.5814
   - Specificity = 0.6190
   - Sensitivity = 0.5455
   - Precision = 0.6

c. The performance measures in b suggest that the performance of the KNN classifier is mediocre at best. The accuracy rate suggests that only 58.14% of the cases in the validation set are classified correctly by the model. Given that there are 22 target class (1) cases out of the 43 cases in the validation data set, the naïve rule (classifying all cases into the predominant class) would have an accuracy rate of 22/43 = 0.5116. The KNN

classifier does not offer much improvement over the naïve rule. Both sensitivity (0.5455) and specificity (0.6190) show that only a little more than half of the target and non-target cases are classified correctly. The lift chart and ROC curve suggest that the KNN classifier performs only slightly better than the baseline model. The AUC value is 0.5411, which is very close to the baseline model (AUC = 0.5).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:5])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[5] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
#c
validationSet$y <- as.numeric(as.character(validationSet$y))
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
gains_table <- gains(validationSet$y, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
```

```
plot.roc(roc_object)
auc(roc_object)
```

12_3. <Analytic Solver>

a. The optimal vale of k is 6.

b. The misclassification rate associated with k=6 is 4.1667% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 93.75%
   - Specificity = 0.9286
   - Sensitivity = 1
   - Precision = 0.6667

d. The AUC value is 0.9643.

e. The predicted response value of the first observation is 0.

12_3. <R>

a. The optimal value of k is 4.

b. The misclassification rate associated with k=4 is (1 - 0.88) = 0.12 (Using the cross-validation results).

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.9032
   - Specificity = 0.96
   - Sensitivity = 0.6667
   - Precision = 0.8

d. The AUC value is 0.9467.

e. The predicted response value of the first observation is 0.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
```

```
library(pROC)
myData1<- scale(myData[2:5])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[5] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#e
PreProcessing <- preProcess(myData[ , 2:5], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
```


12_4. <Analytic Solver>

a. The optimal vale of k is 9.

b. The misclassification rate associated with k=9 is 18.3333% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 67.5%
   - Specificity = 0.8438
   - Sensitivity = 0
   - Precision = 0

d. Cutoff Value = .20 (Using the validation data set)
   - Accuracy rate = 77.5%
   - Specificity = 0.9375
   - Sensitivity = 0.125

- Precision = 0.333

e. The performance measures in c suggest that the performance of the KNN classifier is not very good. Using 0.5 as the cutoff value, the model's accuracy rate suggests that only 67.5% of the cases in the test set are classified correctly by the model. Given that there are 40 target class (1) cases out of the 200 cases, the naïve rule (classifying all cases into the predominant class) would have an accuracy rate of 1 – 40/200 = 0.8. The sensitivity value is 0 meaning that none of the target class cases is classified correctly. Both the lift chart and ROC curve suggest that the KNN classifier does not offer much improvement over the baseline model (random classifier). The AUC value is 0.5313, which is very close to the baseline model (AUC = 0.5).

12_4. <R>

a. The optimal value of k is 6.

b. The misclassification rate associated with k=6 is (1 – 0.7837) = 0.2163 (Using the cross-validation results).

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.75
   - Specificity = 0.9375
   - Sensitivity = 0
   - Precision = 0/(0+4) = 0

d. Cutoff Value = .20 (Using the validation data set)
   - Accuracy rate = 0.6
   - Specificity = 0.625
   - Sensitivity = 0.5
   - Precision = 0.25

e. The performance measures in c suggest that the KNN classifier is slightly effective. Using 0.5 as the cutoff value, we find that the accuracy rate suggests that 75% of the cases in the validation set are classified correctly by the model. Given that there are 16 target class (1) cases out of the 80 cases, the naïve rule (classifying all cases into the predominant class) would have an accuracy rate of 1 – 16/80 = 0.8 but a sensitivity value of 0. The sensitivity value is 0 meaning that none of the target class cases, usually the cases that the analyst is most interested in, is classified correctly. However, changing the cutoff rate to 0.2 generates a sensitivity value of 0.5. Both the lift chart and ROC curve suggest that the KNN classifier does not offer much improvement over the baseline model (random classifier). The AUC value is 0.6406, which is close to the baseline model (AUC = 0.5).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:6])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[6] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
KNN_Class_prob
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.2, '1',
'0')),   validationSet$y, positive = '1')
#e
validationSet$y <- as.numeric(as.character(validationSet$y))
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
gains_table <- gains(validationSet$y, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
plot.roc(roc_object)
```

```
auc(roc_object)
```

12_5. <Analytic Solver>

a. The optimal vale of k is 9.

b. The misclassification rate associated with k=9 is 20.6667% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 82.5%
   - Specificity = 0.9192
   - Sensitivity = 0.7327
   - Precision = 0.9024

d. The performance measures in c suggest that the performance of the KNN classifier is better than that of the naïve rule (classifying all cases into the predominant class). Given that there are 101 target class (1) cases out of the 200 cases in the test data set, the naïve rule would have an accuracy rate of 0.505. In addition, sensitivity and specificity show that 73.27% of the target class cases and 91.92% of the non-target class cases are classified correctly. Both the lift chart and ROC curve lie well above the diagonal lines suggesting that the KNN classifier performs better than the baseline model (random classifier). The AUC value is 0.8611, which is close to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5). In conclusion, the KNN model is effective in classifying the data.

12_5. <R>

a. The optimal value of k is 7.

b. The misclassification rate associated with k=7 is (1-0.7853) = 0.2147 (Using the cross-validation results).

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.792
   - Specificity = 0.9246
   - Sensitivity = 0.6600
   - Precision = 0.8980

d. The performance measures in c suggest that the performance of the KNN classifier is better than that of the naïve rule (classifying all cases into the predominant class). Given that there are 200 target class (1) cases out of the 399 cases in the validation data, the naïve rule would have an accuracy rate of 0.5013. In addition, sensitivity and specificity show that 66% of the target class cases and 92.46% of the non-target class cases are

classified correctly. Both the lift chart and ROC curve lie well above the diagonal lines suggesting that the KNN classifier performs better than the baseline model (random classifier). The AUC value is 0.8305, which is close to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5). In conclusion, the KNN model is effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[4] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit

#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')

#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
```

```
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```

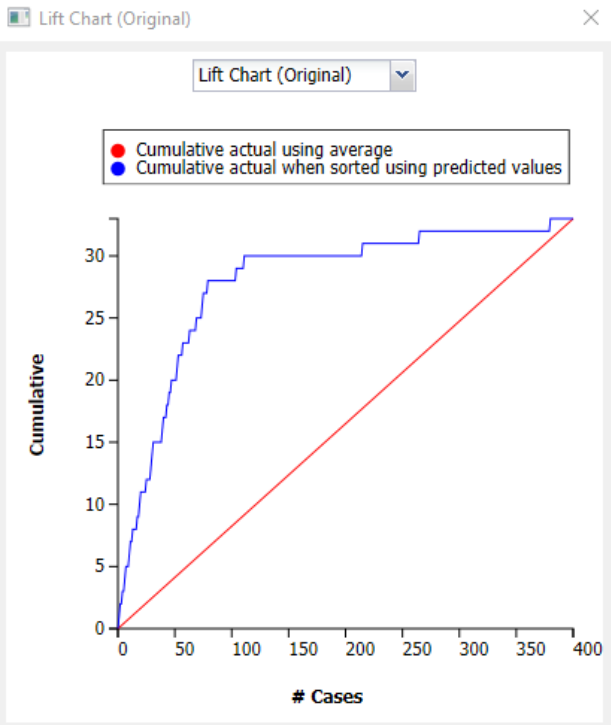
12_6. <Analytic Solver>

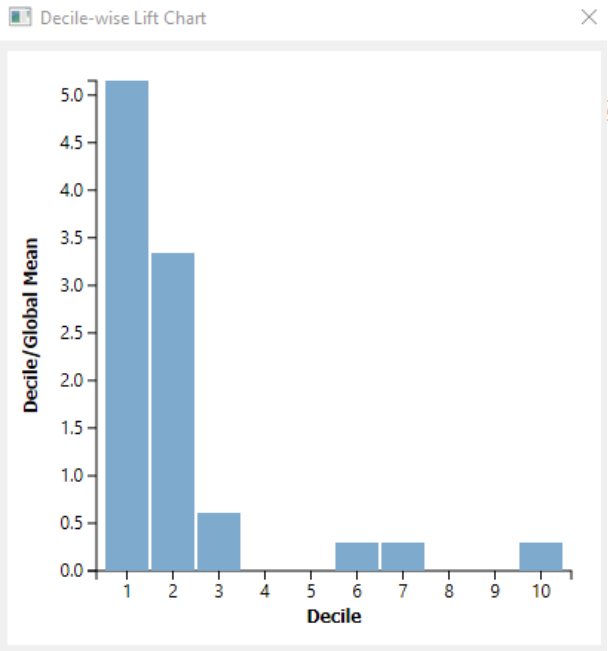
a. The optimal vale of k is 9.

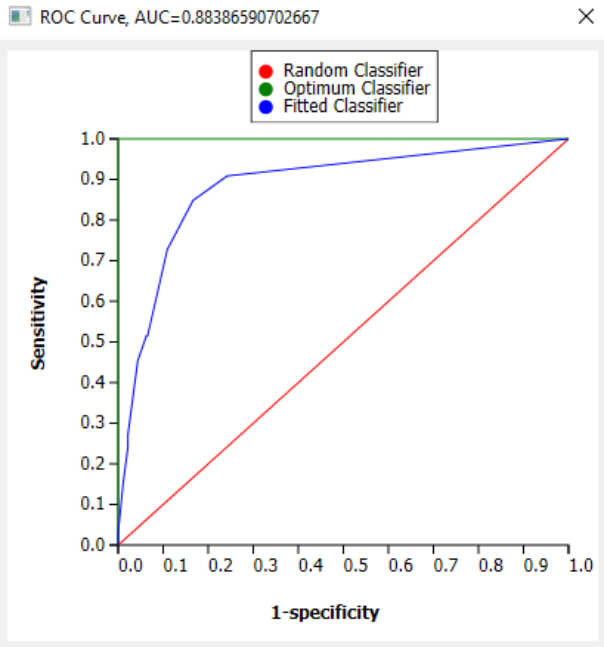
b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 91.5%
   - Specificity = 0.9564
   - Sensitivity = 0.4545
   - Precision = 0.4839
   - AUC = 0.8839

c. The specificity measure shows that using the cutoff value of 0.5, 95.64% of the non-target class cases are classified correctly, and the sensitivity measure shows that 45.45% of the target class cases are classified correctly. This suggests that the KNN classifier is better at classifying non-target class cases than classifying target class cases. However, it is important to understand that both the specificity and sensitivity are affected by the cutoff value. One can choose to increase or decrease the cutoff value to classify more cases into the non-target or target class thus yield different specificity, sensitivity and overall accuracy measures. The performance of the KNN classifier is better assessed using the cumulative lift chart and ROC curve, which are independent of the cutoff value.

d. The plots are below:







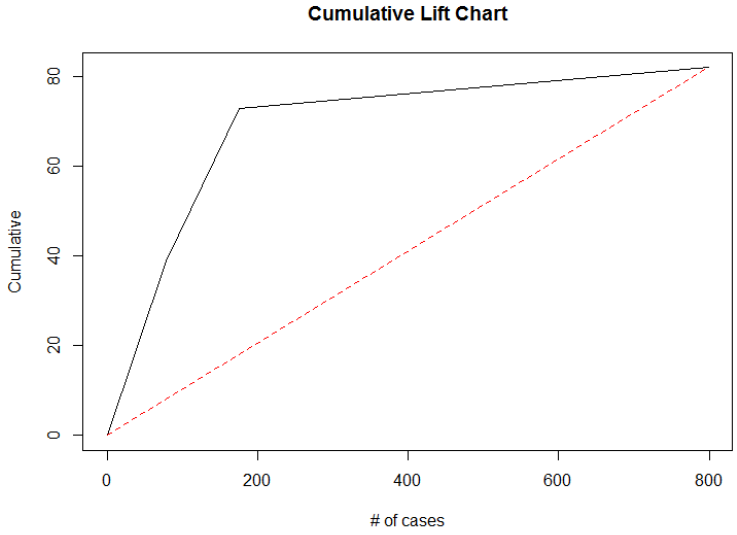
12_6. <R>

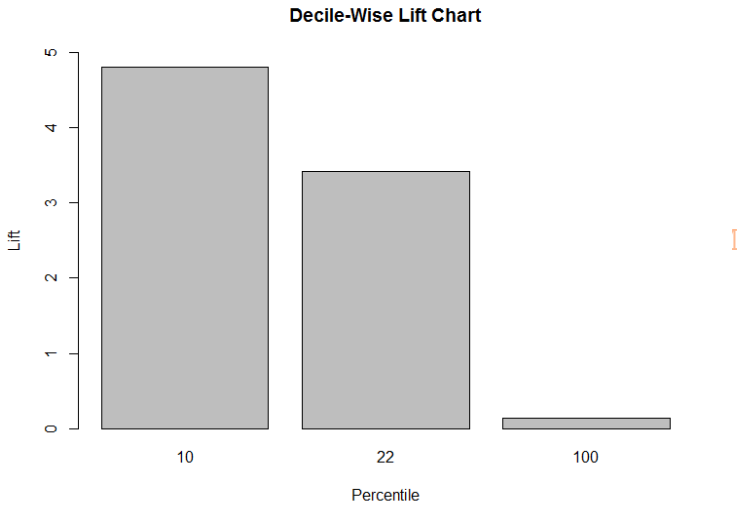
a. The optimal value of k is 10.

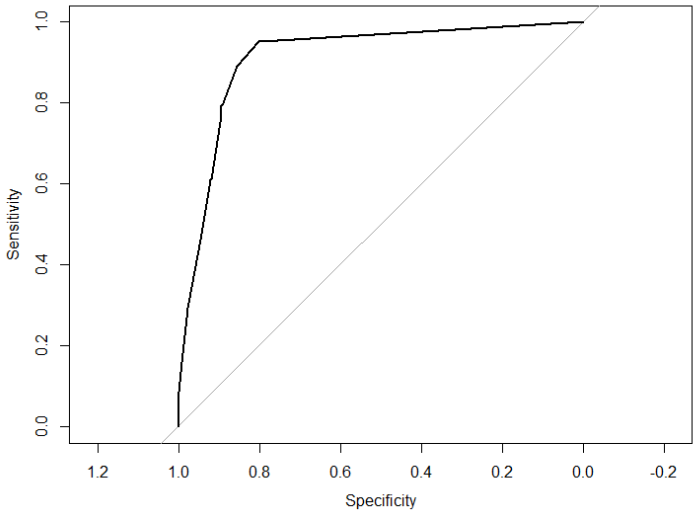
b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.9036
   - Specificity = 0.9623
   - Sensitivity = 0.3902
   - Precision = 0.5424
   - AUC = 0.9118

c. The specificity measure shows that using the cutoff value of 0.5, 96.23% of the non-target class cases are classified correctly, and the sensitivity measure shows that 39.02% of the target class cases are classified correctly. This suggests that the KNN classifier is better at classifying non-target class cases than classifying target class cases. However, it is important to understand that both the specificity and sensitivity are affected by the cutoff value. One can choose to increase or decrease the cutoff value to classify more cases into the non-target or target class thus yield different specificity, sensitivity and overall accuracy measures. The performance of the KNN classifier is better assessed using the cumulative lift chart and ROC curve, which are independent of the cutoff value.

d. The plots are below:







R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[4] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit

#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
auc(roc_object)
```

```
#c No R code is needed for part c.

#d
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,5), main="Decile-Wise Lift Chart")
plot.roc(roc_object)
```

12_7. <Analytic Solver>

a. The optimal vale of k is 9.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 81.25%
   - Specificity = 0.7105
   - Sensitivity = 0.9048
   - Precision = 0.7755
   - AUC = 0.8571

c. The lift value of the leftmost bar is 1.9048.

d. The predicted response values of the first two observations are 1 and 1.

12_7. <R>

a. The optimal value of k is 3.

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7296
   - Specificity = 0.68
   - Sensitivity = 0.7738
   - Precision = 0.7303
   - AUC = 0.7815

c. The lift value of the leftmost bar is 1.48.

d. The predicted response values of the first two observations are 0 and 1.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:5])
myData1<- data.frame(myData1, myData$y)
colnames(myData1)[5] <- 'y'
myData1$y<- as.factor(myData1$y)
set.seed(1)
myIndex<- createDataPartition(myData1$y, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(y ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$y, positive = '1')
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$y, KNN_Class_prob[,2])
auc(roc_object)
#c
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, KNN_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#d
PreProcessing <- preProcess(myData[ , 2:5], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
```


12_8. <Analytic Solver>

a. The optimal value of k is 9.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 79.17%; this is the proportion of the cases that are correctly classified.
   - Specificity = 0.8889; this is the proportion of non-target class cases that the model is able to classify correctly.
   - Sensitivity = 0.7333; this is the proportion of ~~non-~~target class cases that the model is able to classify correctly.
   - Precision = 0.9167; this is the proportion of the predicted target class cases that belong to the target class.

c. The AUC value is 0.8037. The performance measurements and graphs suggest that the KNN classifier is effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (students who are admitted) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8037, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.6 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 79.17% with a sensitivity of 0.7333 and a specificity of 0.8889. This suggests that the model is able to classify 73.33% of the target classes and 88.89% of the non-target class cases correctly. Given that there are 15 target class cases among 24 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 15/24 = 0.625 but specificity of 0. The KNN model performs better than the naïve rule in terms of overall accuracy. In conclusion, the KNN model is effective in classifying the data.

d. The predicted admission outcome for the first new application is 1 (Admit).

12_8. <R>

a. The optimal value of k is 7.

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.8298; this is the proportion of the cases that are correctly classified.
   - Specificity = 0.8421; this is the proportion of non-target class cases that the model is able to classify correctly.
   - Sensitivity = 0.8214; this is the proportion of ~~non-~~target class cases that the model is able to classify correctly.

- Precision = 0.8846; this is the proportion of the predicted target class cases that belong to the target class.

c. The AUC value is 0.828. The performance measurements and graphs suggest that the KNN classifier is effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (students who are admitted) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.828, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 11 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.68 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.8298 with a sensitivity of 0.8214 and a specificity of 0.8421. This suggests that the model is able to classify 82.14% of the target classes and 84.21% of the non-target class cases correctly. Given that there are 28 target class cases among 47 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 28/47 = 0.5957 but specificity of 0. The KNN model performs better than the naïve rule in terms of overall accuracy. In conclusion, the KNN model is effective in classifying the data.

d. The predicted admission outcome for the first new application is 1 (Admit).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:3])
myData1<- data.frame(myData1, myData$Admit)
colnames(myData1)[3] <- 'Admit'
myData1$Admit<- as.factor(myData1$Admit)
set.seed(1)
myIndex<- createDataPartition(myData1$Admit, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
```

```
set.seed(1)
KNN_fit <- train(Admit ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Admit, positive = '1')
#c
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$Admit, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
validationSet$Admit <-
as.numeric(as.character(validationSet$Admit))
gains_table <- gains(validationSet$Admit, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Admit))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Admit))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Admit),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#d
PreProcessing <- preProcess(myData[ , 2:3], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
```


12_9. <Analytic Solver>

a. Misclassification rate$_{k=3}$ = 26.32% (Using the validation data set)
Misclassification rate$_{k=4}$ = 26.32% (Using the validation data set)
Misclassification rate$_{k=5}$ = 26.32% (Using the validation data set)
The optimal value of k is 3 as it involves the least number of neighbors.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 83.33%
   - Specificity = 0.8571
   - Sensitivity = 0.8
   - Precision = 0.8

c. The predicted admission outcome for the first new application is 1 (Pass).

12_9. <R>

a. Misclassification rate$_{k=3}$ = 1 – 0.7150 = 0.2850 (Using the cross-validation results)
Misclassification rate$_{k=4}$ = 1 – 0.6233 = 0.3767 (Using the cross-validation results)
Misclassification rate$_{k=5}$ = 1 – 0.6817 = 0.3183 (Using the cross-validation results)
The optimal value of k is 1.

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.875
- Specificity = 1
- Sensitivity = 0.7857
- Precision = 1

c. The predicted admission outcome for the first new application is 1 (Pass).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:3])
myData1<- data.frame(myData1, myData$Pass)
colnames(myData1)[3] <- 'Pass'
myData1$Pass<- as.factor(myData1$Pass)
set.seed(1)
myIndex<- createDataPartition(myData1$Pass, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Pass ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Pass, positive = '1')
#c
PreProcessing <- preProcess(myData[ , 2:3], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
```

```
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
```

12_10. <Analytic Solver>

a. The optimal value of k is 3.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 56.14%; this is the proportion of the cases that are correctly classified.
   - Specificity = 0.625; this is the proportion of non-target class cases that the model is able to classify correctly.
   - Sensitivity = 0.48; this is the proportion of ~~non~~ target class cases that the model is able to classify correctly.
   - Precision = 0.5; this is the proportion of the predicted target class cases that belong to the target class.

c. The AUC value is 0.6031.

   The performance measurements and graphs suggest that the KNN classifier is slightly effective in classifying the data. The lift curve of the KNN model lies slightly above the diagonal line. This suggests that the KNN classifier performs just a little better than the baseline model (random classifier) and is only able to identify a slightly larger percentage of target class (individuals who respond to a social media campaign) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6031, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs almost the same as the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.824 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 56.14% with a sensitivity of 0.48 and a specificity of 0.625. This suggests that the model is able to classify 48% of the target classes and 62.5% of the non-target class cases correctly. Given that there are 25 target class cases among 57 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (57-25)/57 = 0.5614 but sensitivity of 0. The KNN model has the same overall accuracy as the naïve rule has. In conclusion, the KNN model is only slightly effective in classifying the data.

d. The predicted outcome for the first new consumer is 0 (Did not respond).

e. Cutoff Value = .30 (Using the test data set)
   - Accuracy rate = 54.39%
   - Specificity = 0.5313

- Sensitivity = 0.56
- Precision = 0.4828

12_10. <R>

a. The optimal value of k is 6.

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.6814; this is the proportion of the cases that are correctly classified.
   - Specificity = 0.7681; this is the proportion of non-target class cases that the model is able to classify correctly.
   - Sensitivity = 0.5455; this is the proportion of ~~non-~~target class cases that the model is able to classify correctly.
   - Precision = 0.6; this is the proportion of the predicted target class cases that belong to the target class.

c. The AUC value is 0.7085. (Comment on the performance of the k-NN classifier.)

   The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (individuals who respond to a social media campaign) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7085, which about half way between the baseline model (AUC = 0.5) and the optimum model (AUC = 1), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 12 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.10 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6814 with a sensitivity of 0.5455 and a specificity of 0.7681. This suggests that the model is able to classify 54.55% of the target classes and 76.81% of the non-target class cases correctly. Given that there are 44 target class cases among 113 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (113-44)/113 = 0.6106 but sensitivity of 0. The KNN model has slightly better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

d. The predicted outcome for the first new consumer is 0 (Did not respond).

e. Cutoff Value = .30 (Using the validation data set)
   - Accuracy rate = 0.6106
   - Specificity = 0.4203

- Sensitivity = 0.9091
- Precision = 0.5

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:3])
myData1<- data.frame(myData1, myData$SocialMedia)
colnames(myData1)[3] <- 'SocialMedia'
myData1$SocialMedia<- as.factor(myData1$SocialMedia)
set.seed(1)
myIndex<- createDataPartition(myData1$SocialMedia, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(SocialMedia ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$SocialMedia, positive =
'1')
#c
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$SocialMedia, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
validationSet$SocialMedia <-
as.numeric(as.character(validationSet$SocialMedia))
gains_table <- gains(validationSet$SocialMedia,
KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$SocialMedia))~c(0
, gains_table$cume.obs), xlab = "# of cases", ylab =
"Cumulative", main="Cumulative Lift Chart", type="l")
```

```
lines(c(0, sum(validationSet$SocialMedia))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$SocialMedia),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
#d
PreProcessing <- preProcess(myData[ , 2:3], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
#e
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.3, '1',
'0')), as.factor(validationSet$SocialMedia), positive = '1')
```

12_11. <Analytic Solver>

a. The optimal value of k is 9.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 0.8
   - Specificity = 0.7647
   - Sensitivity = 0.8367
   - Precision = 0.7736

c. The AUC value is 0.8834.

d. The predicted outcome for the first new e-mail is 1 (Spam).

12_11. <R>

a. The optimal value of k is 3.

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.8492
   - Specificity = 0.8958
   - Sensitivity = 0.8058
   - Precision = 0.8925

c. The AUC value is 0.8781.

d. The predicted outcome for the first new e-mail is 1 (Spam).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Spam)
colnames(myData1)[4] <- 'Spam'
myData1$Spam<- as.factor(myData1$Spam)
set.seed(1)
myIndex<- createDataPartition(myData1$Spam, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Spam ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Spam, positive = '1')
#c
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$Spam, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
PreProcessing <- preProcess(myData[ , 2:4], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
```


12_12. <Analytic Solver>

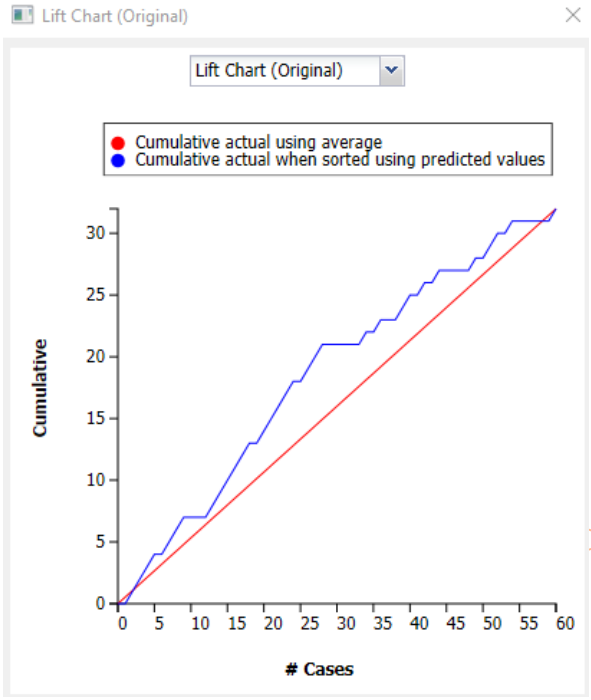
a. The optimal value of k is 3.

b. The misclassification rate associated with k=3 is 33.33% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 0.5833; this is the proportion of the cases that are correctly classified.

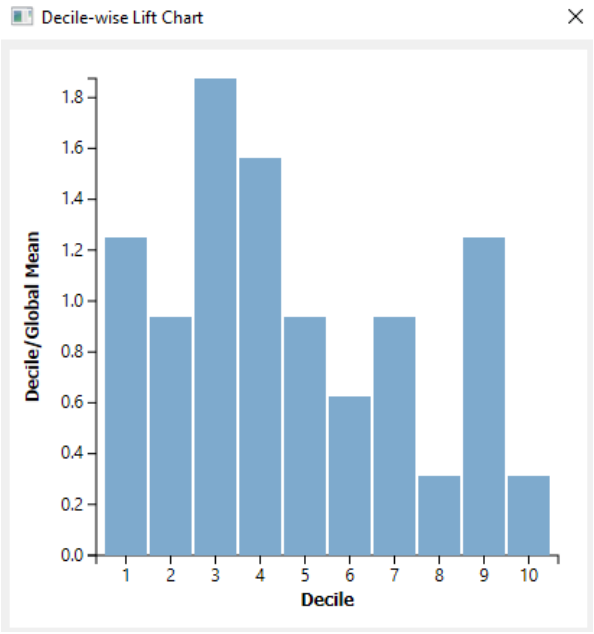
- Specificity = 0.2857; this is the proportion of non-target class cases that the model is able to classify correctly.
- Sensitivity = 0.8438; this is the proportion of ~~non-~~target class cases that the model is able to classify correctly.
- Precision = 0.5745; this is the proportion of the predicted target class cases that belong to the target class.

d.



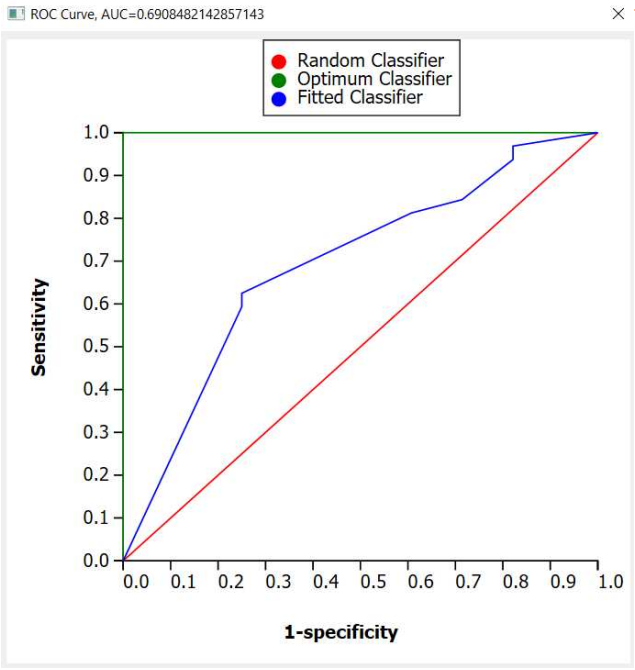
No, a part of the lift curve lies below the baseline.

e.



The lift value of the leftmost bar is 1.25.

f.



The AUC value is 0.6908.

g. The performance measurements and graphs suggest that the KNN classifier is slightly effective in classifying the data. The lift curve of the KNN model lies slightly above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a slightly larger percentage of target class (threats) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6908, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.25 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 58.33% with a sensitivity of 0.8438 and a specificity of 0.2857. This suggests that the model is able to classify 84.38% of the target classes and 28.57% of the non-target class cases correctly. Given that there are 32 target class cases among 60 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 32/60 = 0.5333 but a specificity of 0. The KNN model has slightly better overall accuracy than the naïve rule has. In conclusion, the KNN model is slightly effective in classifying the data.

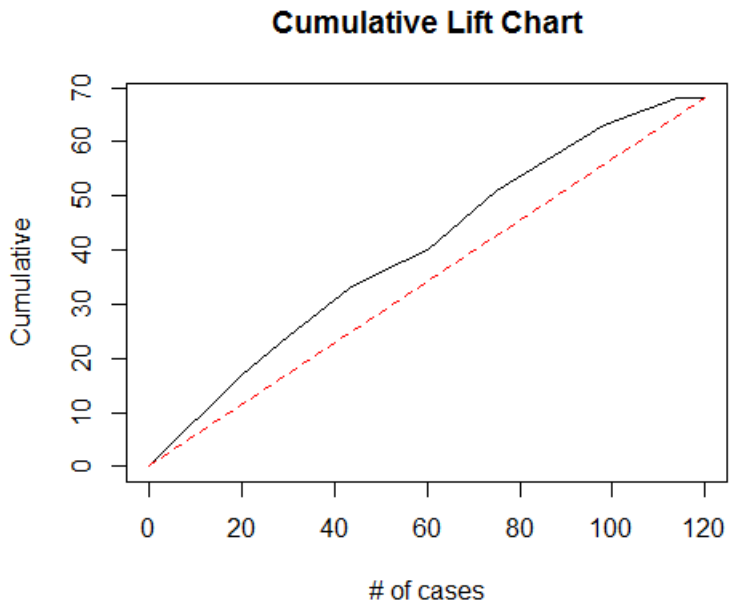
12_12. <R>

a. The optimal value of k is 9

b. The misclassification rate associated with k=9 is 1-0.6928 = 0.3072 (Using the cross-validation result).

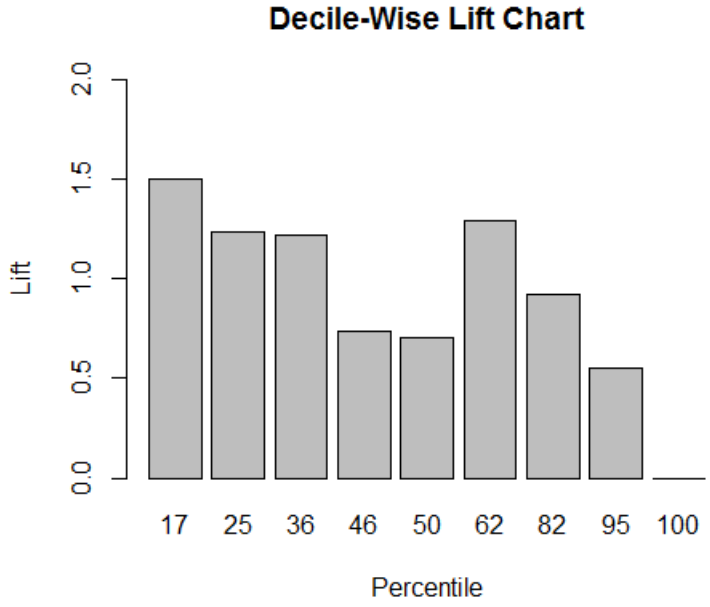
c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.6167
   - Specificity = 0.6154
   - Sensitivity = 0.6176
   - Precision = 0.6774

d.



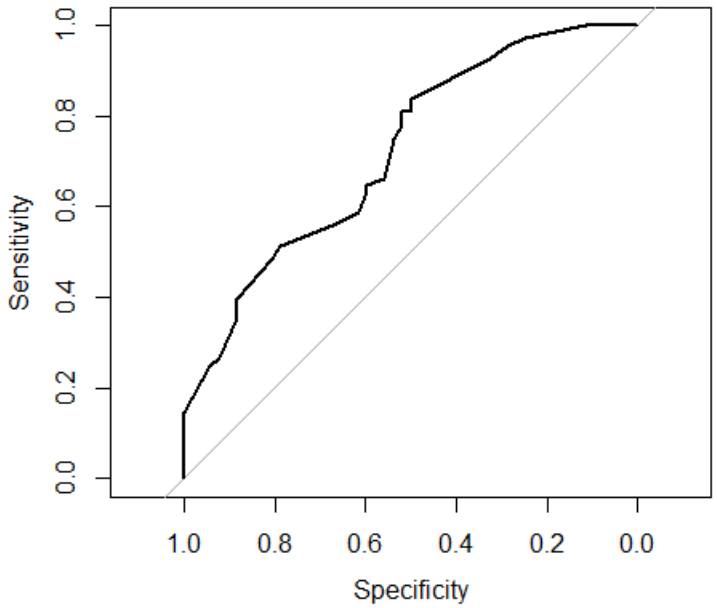
Yes, the lift curve lies above the baseline.

e.



The lift value of the leftmost bar is 1.5.

f.



The AUC value is 0.7216.

g. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (threat) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7216, which is about half way between the baseline model (AUC = 0.5) and the optimum model (AUC = 1), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 17 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.5 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6167 with a sensitivity of 0.6176 and a specificity of 0.6154. This suggests that the model is able to classify 61.76% of the target classes and 61.54% of the non-target class cases correctly. Given that there are 68 target class cases among 120 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 68/120 = 0.5667 but a specificity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
```

```
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Threat)
colnames(myData1)[4] <- 'Threat'
myData1$Threat<- as.factor(myData1$Threat)
set.seed(1)
myIndex<- createDataPartition(myData1$Threat, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Threat ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Threat, positive = '1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$Threat <-
as.numeric(as.character(validationSet$Threat))
gains_table <- gains(validationSet$Threat, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Threat))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Threat))~c(0,
dim(validationSet)[1]), col="red", lty=2)
#e
barplot(gains_table$mean.resp/mean(validationSet$Threat),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#f
roc_object <- roc(validationSet$Threat, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#g No R code is needed for this question.
```

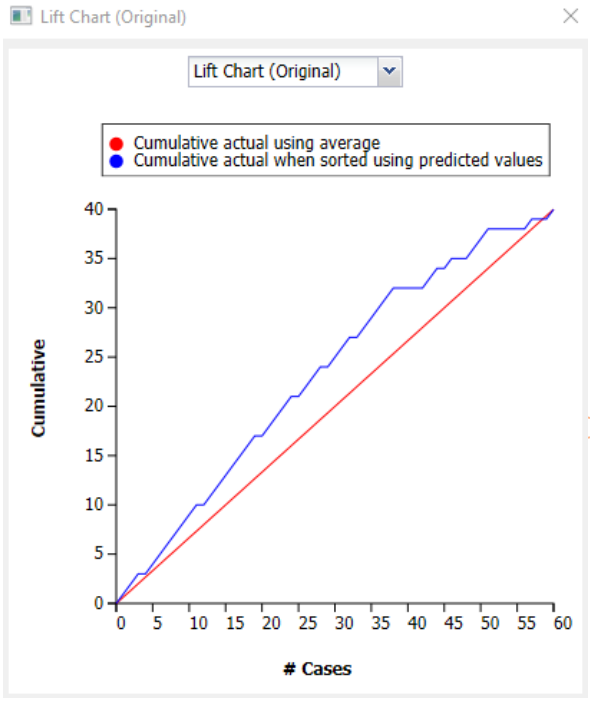

12_13. <Analytic Solver>

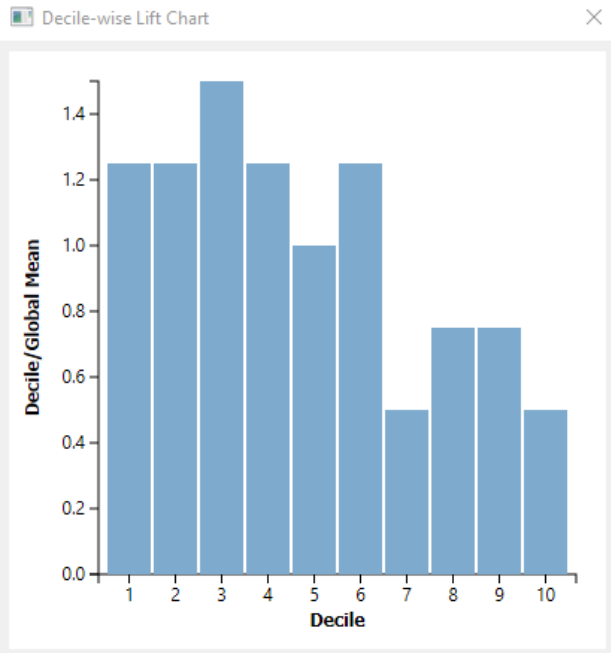
a. The optimal k is 8. The predicted outcome for the first new employee is 0 (Will not be promoted into a management role after 10 years with the company).

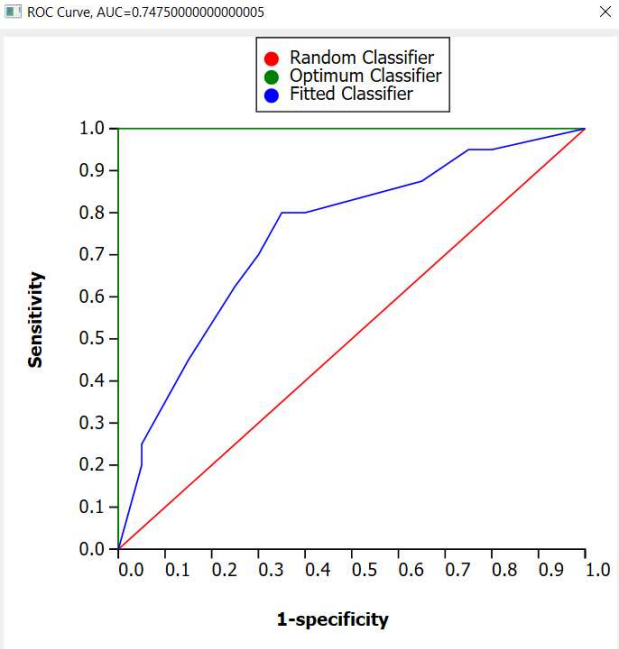
b. The misclassification rate associated with k = 8 is 18.8889%.

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 0.75
   - Specificity = 0.65
   - Sensitivity = 0.8
   - Precision = 0.8205

d.







e. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (employees who are promoted into management positions) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to

the target class. The AUC value is 0.7475, which is about half way between the baseline model (AUC = 0.5) and the optimum model (AUC = 1), suggesting that the model performs moderately better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.25 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 75% with a sensitivity of 0.80 and a specificity of 0.65. This suggests that the model is able to classify 80% of the target classes and 65% of the non-target class cases correctly. Given that there are 40 target class cases among 60 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 40/60 = 0.6667 but a specificity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

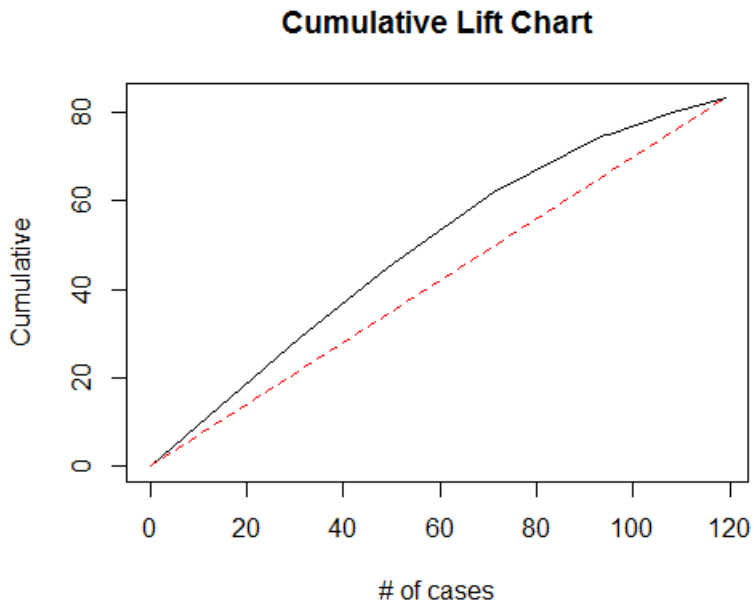
12_13. <R>

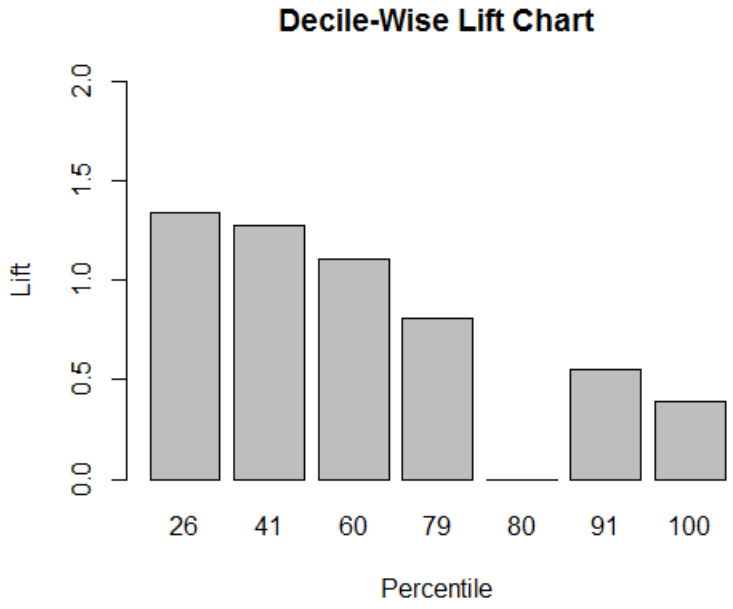
a. The optimal value of k is 7. The predicted outcome for the first new employee is 0 (Will not be promoted to a management position after 10 years with the company).

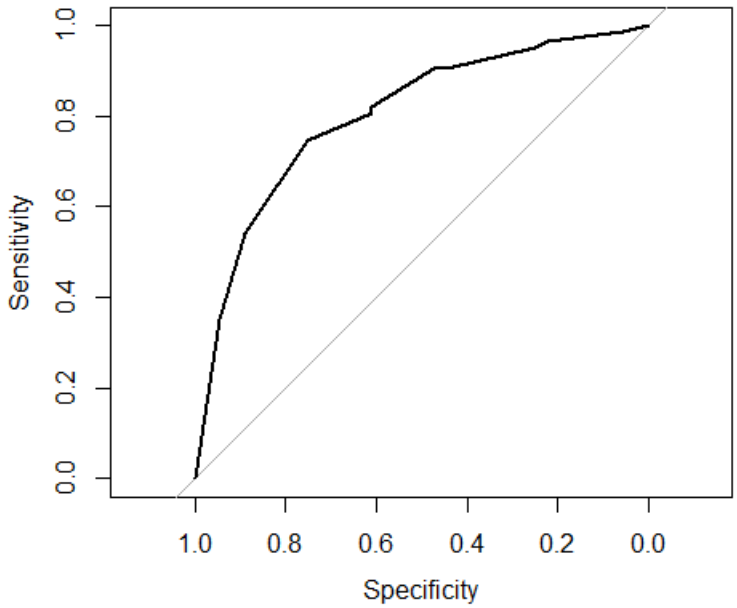
b. The misclassification rate associated with k=7 is 1-0.7754 = 0.2246 (Using the cross-validation result).

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7563
   - Specificity = 0.6111
   - Sensitivity = 0.8193
   - Precision = 68/(68+14) = 0.8293

d.







AUC = 0.8015

e. The performance measurements and graphs suggest that the KNN classifier is effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (employees who are promoted into management positions) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8015, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 26 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.34 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7563 with a sensitivity of 0.8193 and a specificity of 0.6111. This suggests that the model is able to classify 81.93% of the target classes and 61.11% of the non-target class cases correctly. Given that there are 83 target class cases among 119 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 83/119 = 0.6975 but a specificity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Promoted)
colnames(myData1)[4] <- 'Promoted'
myData1$Promoted<- as.factor(myData1$Promoted)
set.seed(1)
myIndex<- createDataPartition(myData1$Promoted, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Promoted ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
PreProcessing <- preProcess(myData[ , 2:4], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Promoted, positive =
'1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$Promoted <-
as.numeric(as.character(validationSet$Promoted))
gains_table <- gains(validationSet$Promoted, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Promoted))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Promoted))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Promoted),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Promoted, KNN_Class_prob[,2])
```

```
plot.roc(roc_object)
auc(roc_object)
#e No R code is needed for this question.
```

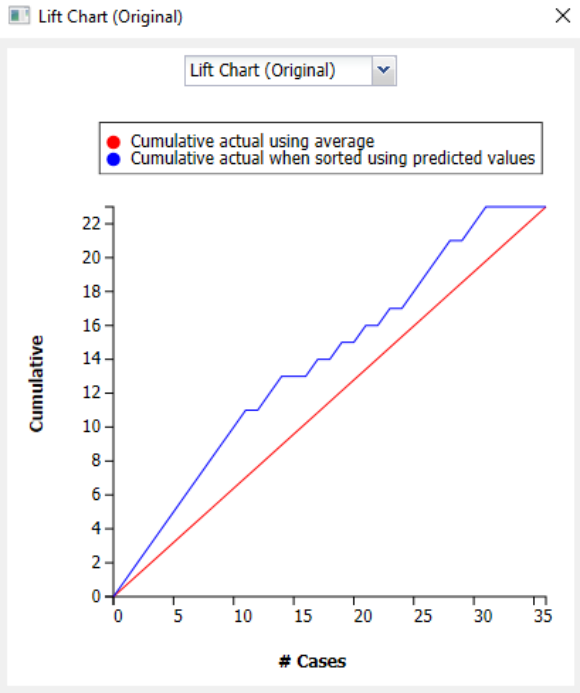
12_14. <Analytic Solver>

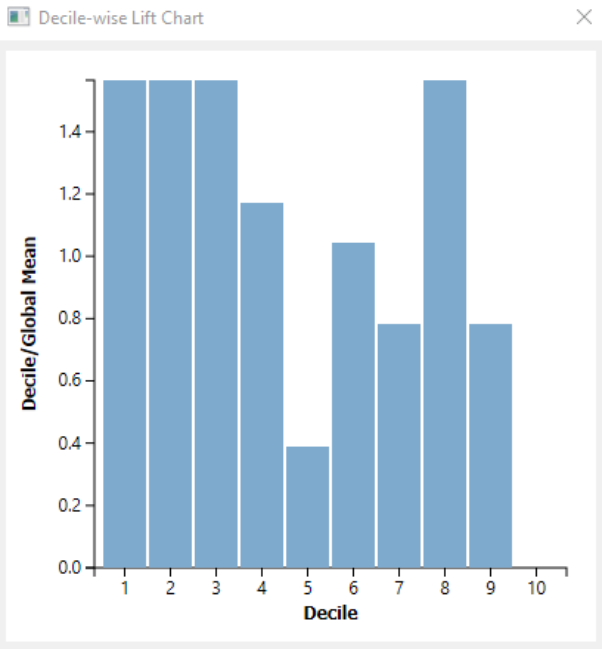
a. The optimal k is 9.

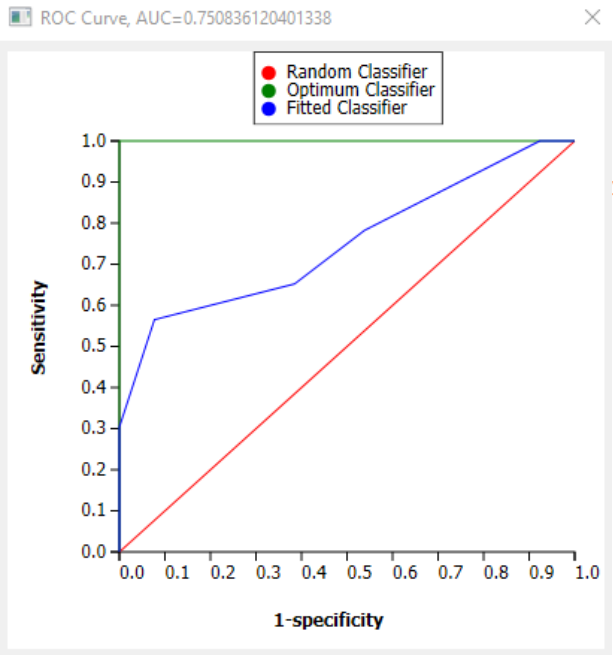
b. The misclassification rate associated with k = 9 is 27.7778%. (Using the validation data set)

c. Cutoff Value = .50 (Using the test data set)
- Accuracy rate = 63.89%
- Specificity = 0.6154
- Sensitivity = 0.6522
- Precision = 0.75

d.







e. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (patients with heart diseases) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7508, which is about half way between the baseline model (AUC = 0.5) and the optimum model (AUC = 1), suggesting that the model performs moderately better than

the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.57 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 63.89% with a sensitivity of 0.6522 and a specificity of 0.6154. This suggests that the model is able to classify 65.22% of the target classes and 61.54% of the non-target class cases correctly. Given that there are 23 target class cases among 36 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 23/36 = 0.6389 but a specificity of 0. The KNN model has slightly better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

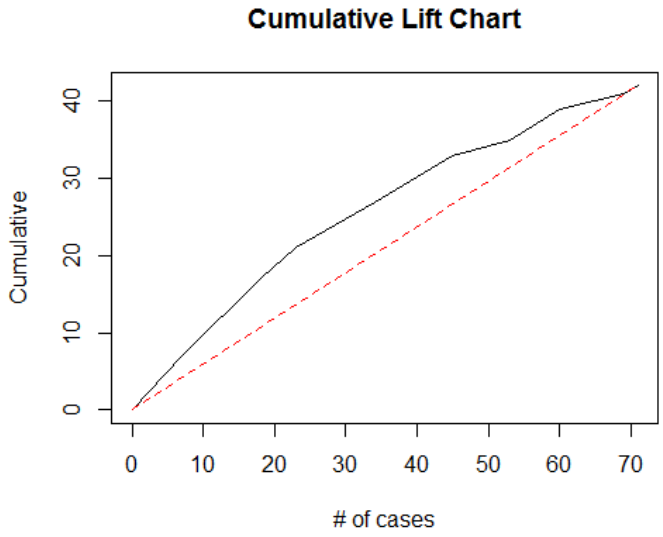
12_14. <R>

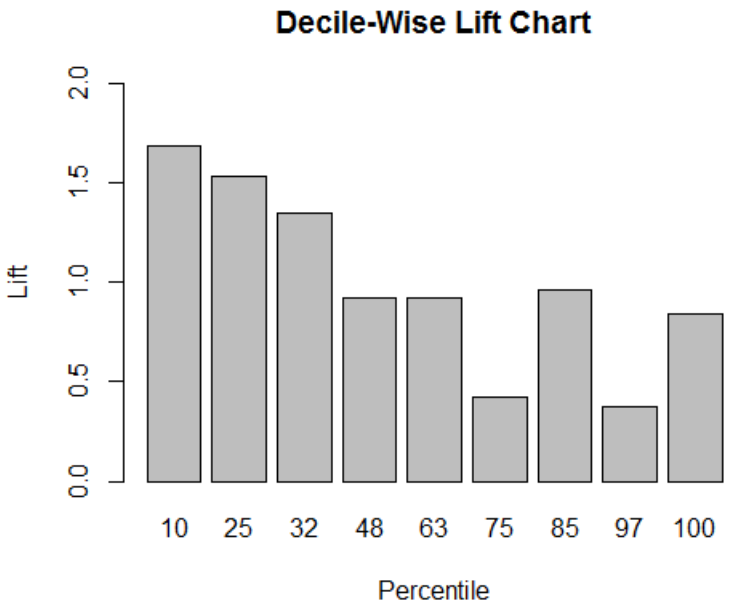
a. The optimal value of k is 9.

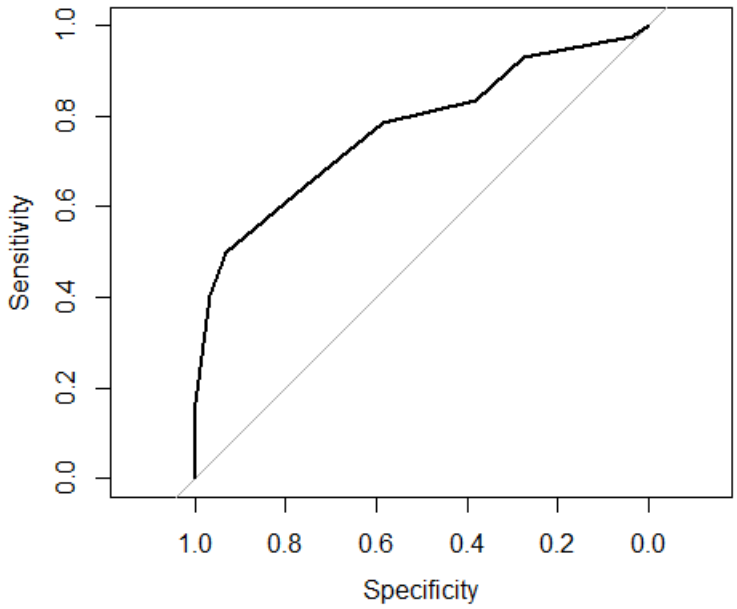
b. The misclassification rate associated with k=9 is 1-0.7541 = 0.2459 (Using the cross-validation result).

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7042
   - Specificity = 0.5862
   - Sensitivity = 0.7857
   - Precision = 0.7333

d.







AUC = 0.7697

e. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (patients who have heart diseases) cases by looking at a small percentage of the

validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7697, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.69 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7042 with a sensitivity of 0.7857 and a specificity of 0.5862. This suggests that the model is able to classify 78.57% of the target classes and 58.62% of the non-target class cases correctly. Given that there are 42 target class cases among 71 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 42/71 = 0.5915 but a specificity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:5])
myData1<- data.frame(myData1, myData$Disease)
colnames(myData1)[5] <- 'Disease'
myData1$Disease<- as.factor(myData1$Disease)
set.seed(1)
myIndex<- createDataPartition(myData1$Disease, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Disease ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Disease, positive = '1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
```

```
validationSet$Disease <-
as.numeric(as.character(validationSet$Disease))
gains_table <- gains(validationSet$Disease, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Disease))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Disease))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Disease),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Disease, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#e No R code is needed for this question.
```

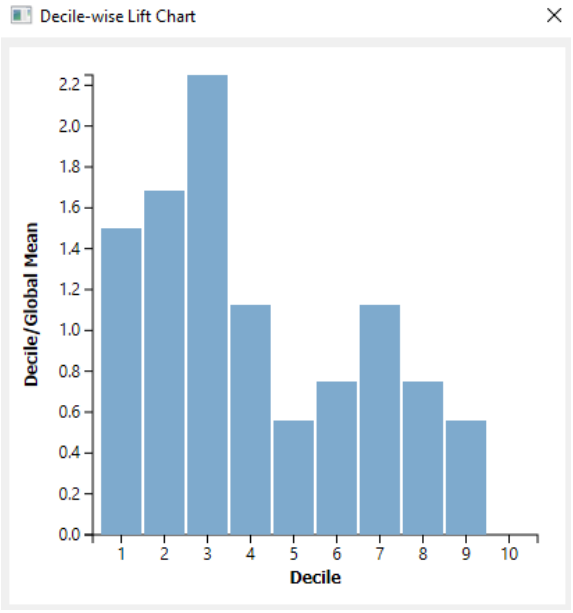
12_15. <Analytic Solver>

a. The optimal vale of k is 3. The predicted outcome for the first new consumer is 1 (Purchase).

b. The misclassification rate associated with k=3 is 31.48% (Using the validation data set).

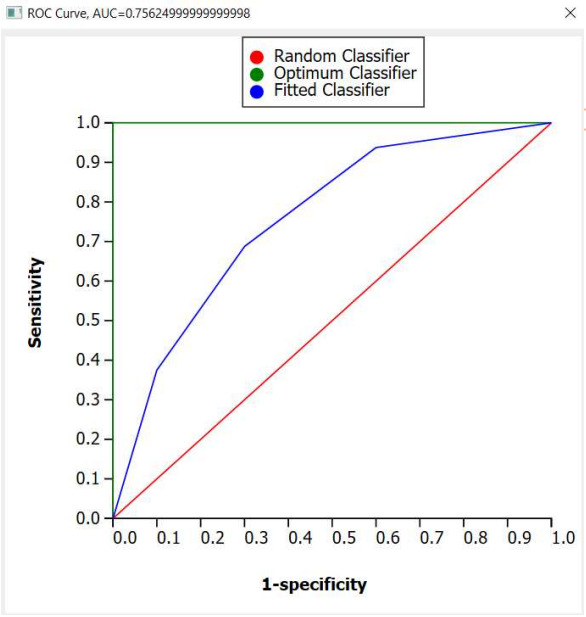
c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 69.44%
   - Specificity = 0.7
   - Sensitivity = 0.6875
   - Precision = 0.6471

d.



The lift value of the leftmost bar is 1.5.

e.



The AUC value is 0.7562.

f. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (customers who purchase) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is

0.7562, which is about half way between the baseline model (AUC = 0.5) and the optimum model (AUC = 1), suggesting that the model performs moderately better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.5 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 69.44% with a sensitivity of 0.6875 and a specificity of 0.7. This suggests that the model is able to classify 68.75% of the target classes and 70% of the non-target class cases correctly. Given that there are 16 target class cases among 36 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (36-16)/36 = 0.5556 but a sensitivity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

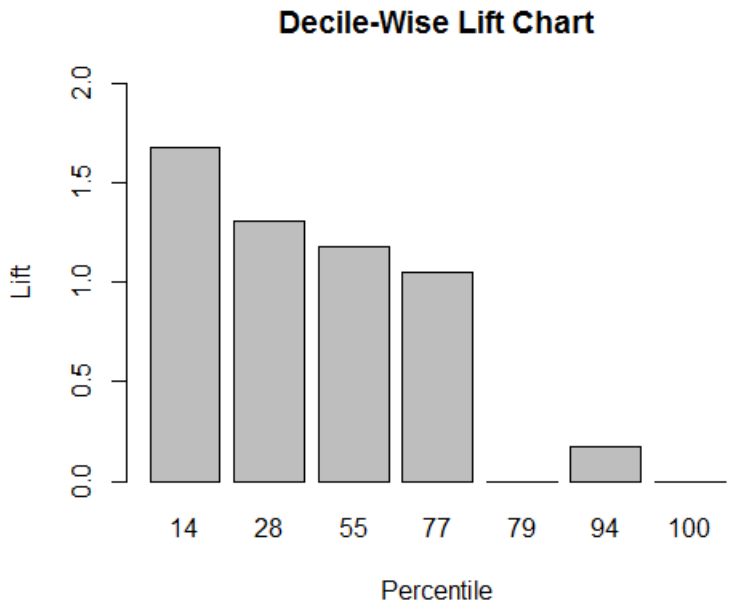
12_15. <R>

a. The optimal vale of k is 9. The predicted outcome for the first new consumer is 1 (Purchase).

b. The misclassification rate associated with k=9 is 1-0.5677 = 0.4323 (Using the cross-validation results).

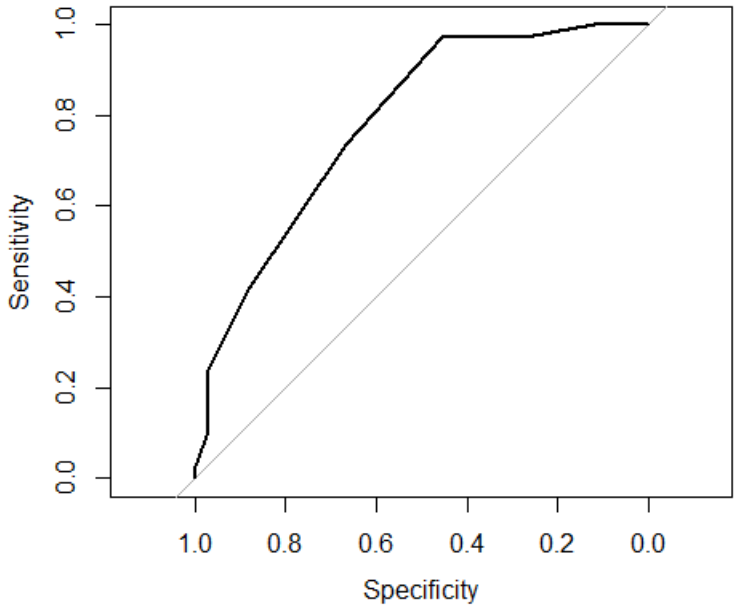
c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7042
   - Specificity = 0.6667
   - Sensitivity = 0.7368
   - Precision = 0.7179

d.



The lift value of the leftmost bar is 1.68.

e.



The AUC value is 0.7839.

f. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class

(customers who purchase) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7839, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 14 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.68 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7042 with a sensitivity of 0.7368 and a specificity of 0.6667. This suggests that the model is able to classify 73.68% of the target classes and 66.67% of the non-target class cases correctly. Given that there are 38 target class cases among 71 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 38/71 = 0.5352 but a specificity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Purchase)
colnames(myData1)[4] <- 'Purchase'
myData1$Purchase<- as.factor(myData1$Purchase)
set.seed(1)
myIndex<- createDataPartition(myData1$Purchase, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Purchase ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
PreProcessing <- preProcess(myData[ , 2:4], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
#c
```

```
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Purchase, positive =
'1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$Purchase <-
as.numeric(as.character(validationSet$Purchase))
gains_table <- gains(validationSet$Purchase, KNN_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$Purchase),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#e
roc_object <- roc(validationSet$Purchase, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#f No R code is needed for this question.
```

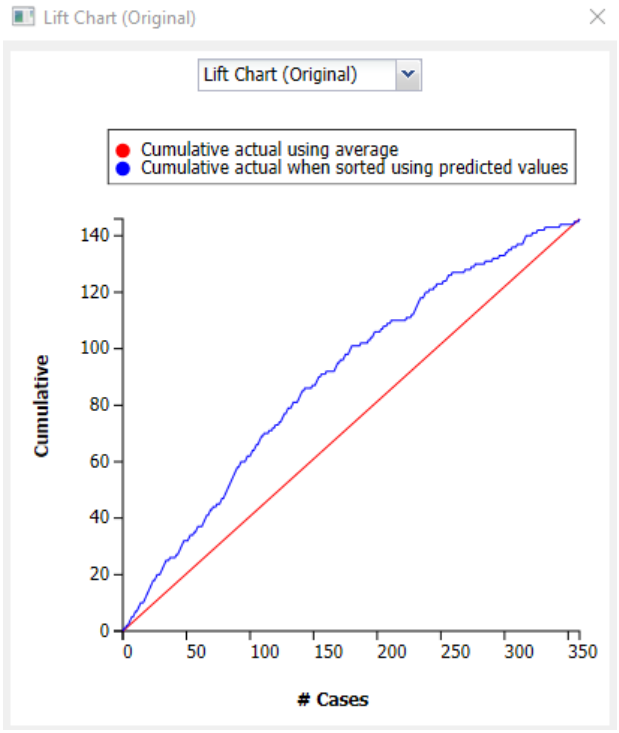
12_16. <Analytic Solver>

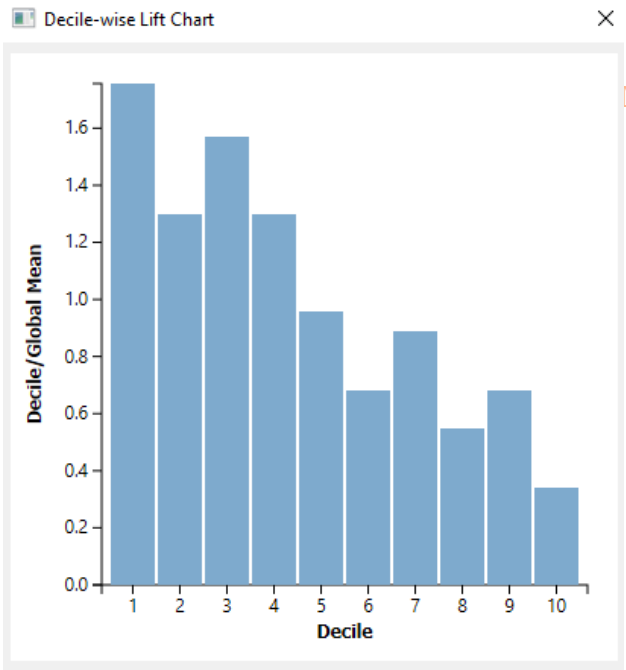
a. The optimal vale of k is 7. The predicted outcome for the first new consumer is N (Will not install solar panels).

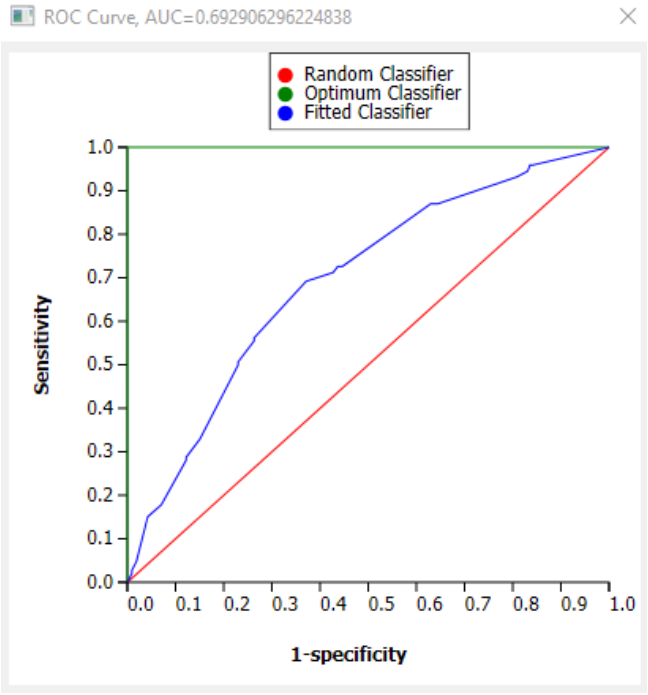
b. The misclassification rate associated with k=7 is 36.5492% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 66.30%
   - Specificity = 0.7371
   - Sensitivity = 0.5548
   - Precision = 0.5912

d.







e. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (customers who install solar panels) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6929, which is slightly closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.76 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 66.30% with a sensitivity of 0.5548 and a specificity of 0.7371. This suggests that the model is able to classify 55.48% of the target classes and 73.71% of the non-target class cases correctly. Given that there are 146 target class cases among 359 test cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (359-146)/359 = 0.5933 but a sensitivity of 0. The KNN model has better overall accuracy than the naïve rule has. In conclusion, the KNN model is moderately effective in classifying the data.

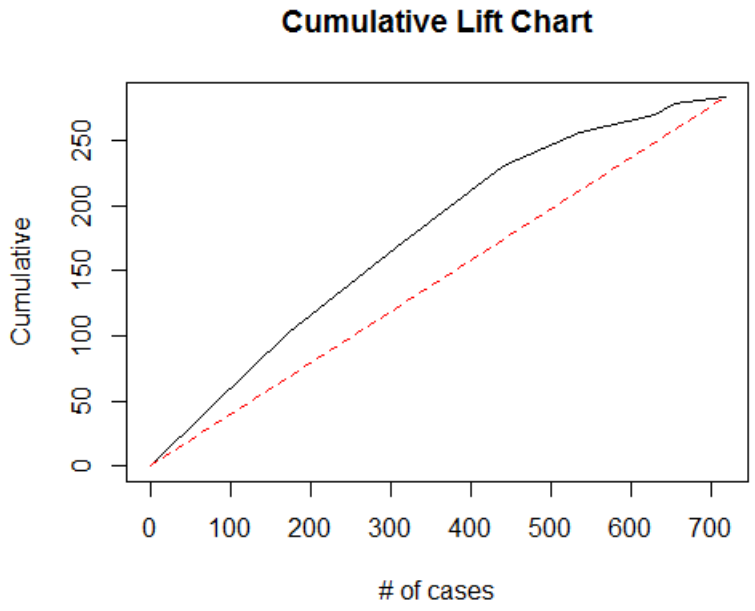
12_16. <R>

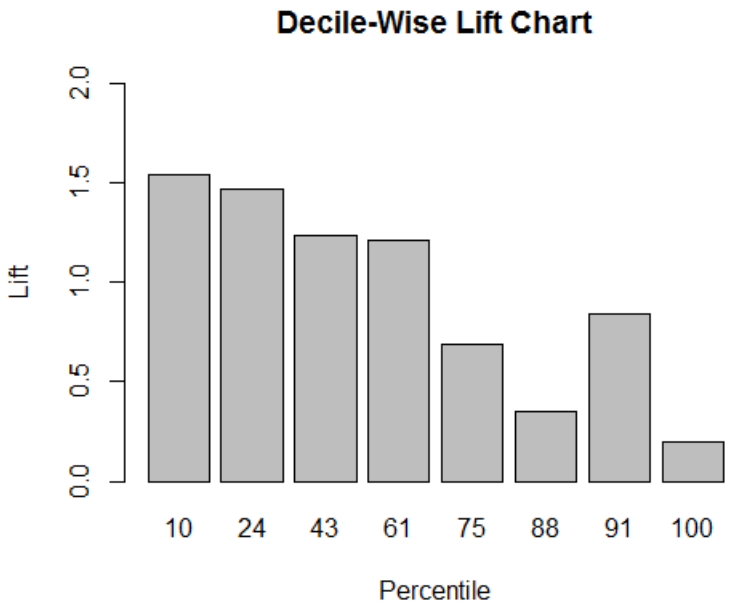
a. The optimal vale of k is 8. The predicted outcome for the first new consumer is N (Will not install solar panels).

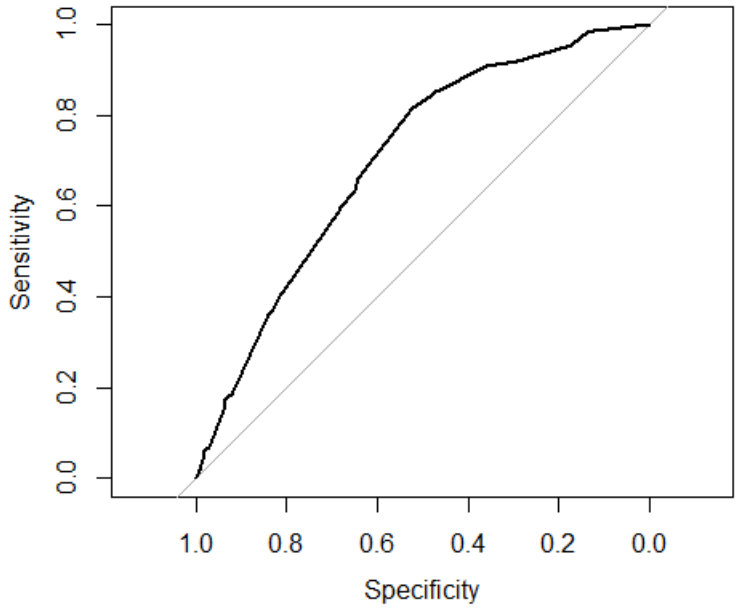
b. The misclassification rate associated with k=8 is 1 – 0.6569182 = 0.3431818 (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
- Accuracy rate = 0.6569
- Specificity = 0.7535
- Sensitivity = 0.5088
- Precision = 0.5737

d.







AUC = 0.7022

e. The performance measurements and graphs suggest that the KNN classifier is moderately effective in classifying the data. The lift curve of the KNN model lies above the diagonal line. This suggests that the KNN classifier performs better than the baseline model (random classifier) and is able to identify a moderately larger percentage of target class (customers who install solar panels) cases by looking at a small percentage of the

validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7022, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs moderately better than baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 1.54 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6569 with a sensitivity of 0.5088 and a specificity of 0.7535. This suggests that the model is able to classify 50.88% of the target classes and 75.35% of the non-target class cases correctly. Given that there are 283 target class cases among the 717 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (717-283)/717 = 0.6053 but with a sensitivity of 0. The KNN model shows moderately better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. In conclusion, the KNN model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData1<- scale(myData[1:2])
myData1<- data.frame(myData1, myData$Install)
colnames(myData1)[3] <- 'Install'
myData1$Install<- as.factor(myData1$Install)
set.seed(1)
myIndex<- createDataPartition(myData1$Install, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Install ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
PreProcessing <- preProcess(myData[ , 1:2], method = c("center",
"scale"))
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1)
KNN_Score
#c
```

```
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Install, positive = 'Y')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$Install <- ifelse(validationSet$Install == "Y", 1,
0)
gains_table <- gains(validationSet$Install, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Install))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Install))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Install),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Install, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#e No R code is needed for this question.
```

12_17. <Analytic Solver>

a. The optimal vale of k is 5.

b. The misclassification rate associated with k=5 is 42.3611% (Using the validation data set).

c. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 63.54%
   - Specificity = 0.8333
   - Sensitivity = 0.3056
   - Precision = 0.5238

d. The AUC is 0.5956.

e. The performance measurements and graphs suggest that the KNN classifier is not very effective in predicting whether the machine will break down during the next production period. Both the lift chart and ROC curve show that the KNN classifier only performs slightly better than the base model (random classifier). The overall accuracy is 63.54%. Given that there are 36 target class cases among the 96 test data cases, the naïve rule would have an accuracy rate of (96-36)/96 = 62.5%. While the KNN classifier is able to correctly classify the great majority of the non-target class cases, it is only able to classify 30.56% of the target class cases correctly. In conclusion, KNN is not a very effective way to classify in this case.

f. Cutoff Value = .30 (Using the test data set)
- Accuracy rate = 61.46%
- Specificity = 0.6833
- Sensitivity = 0.5
- Precision = 0.4865

12_17. <R>

a. The optimal vale of k is 5.

b. The misclassification rate associated with k=5 is 1 – 0.5746 = 0.4254 (Using the cross-validation results).

c. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.5602
- Specificity = 0.7107
- Sensitivity = 0.3
- Precision = 0.375

d. The AUC is 0.5031.

e. The performance measurements and graphs suggest that the KNN classifier is not a very effective in predicting whether the machine will break down during the next production period. Both the lift chart and ROC curve show that the KNN classifier performs about the same as the base model (random classifier). The overall accuracy is only 0.5602. Given that there are 70 target class cases among the 191 test data cases, the naïve rule would have an accuracy rate of (191-70)/191 = 63.35%. While the KNN classifier is able to correctly classify the great majority of the non-target class cases, it is only able to classify 30% of the target class cases correctly. In conclusion, KNN is not a very effective way to classify in this case.

f. Cutoff Value = .30 (Using the validation data set)
- Accuracy rate = 0.466
- Specificity = 0.3554
- Sensitivity = 0.6571
- Precision = 0.3710

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
```

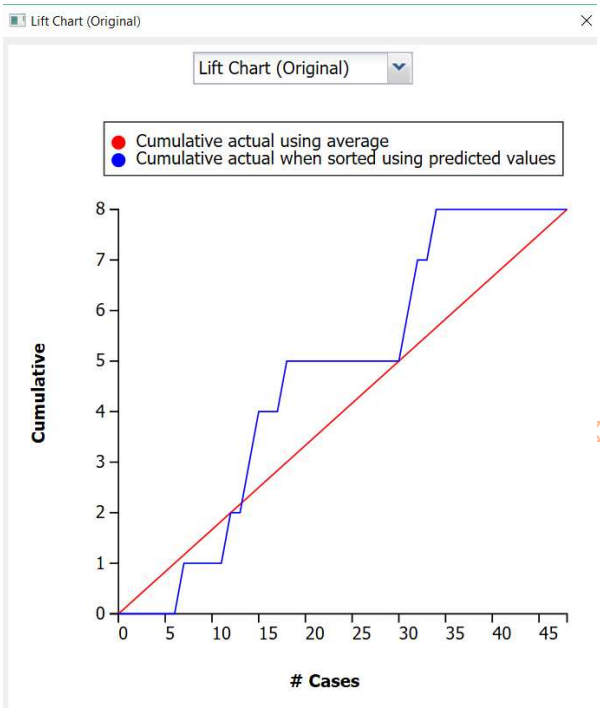
```
library(gains)
library(pROC)
myData1<- scale(myData[1:2])
myData1<- data.frame(myData1, myData$Breakdown)
colnames(myData1)[3] <- 'Breakdown'
myData1$Breakdown<- as.factor(myData1$Breakdown)
set.seed(1)
myIndex<- createDataPartition(myData1$Breakdown, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Breakdown ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#c
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Breakdown, positive =
'1')
#d
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$Breakdown, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#e
validationSet$Breakdown <-
as.numeric(as.character(validationSet$Breakdown))
gains_table <- gains(validationSet$Breakdown, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Breakdown))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Breakdown))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Breakdown),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#f
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.3, '1',
'0')), as.factor(validationSet$Breakdown), positive = '1')
```

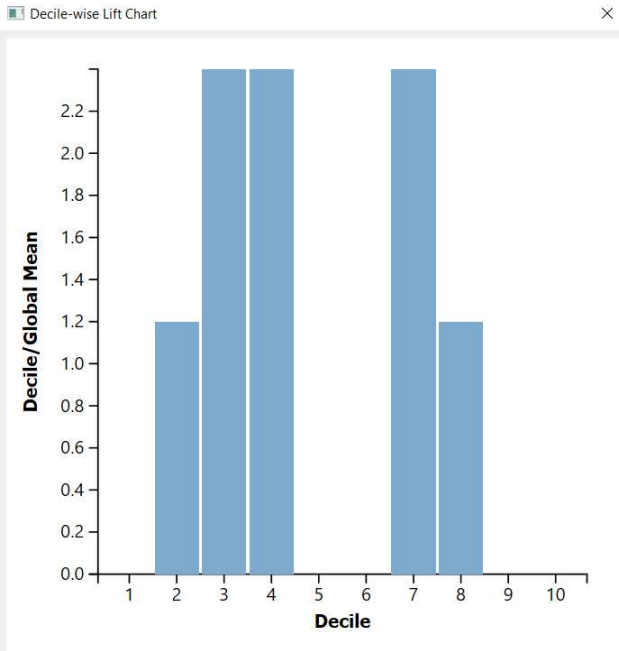

12_18. <Analytic Solver>

a. The optimal value of k is 8.

b. Cutoff Value = .50 (Using the test data set)
   - Accuracy rate = 72.92%
   - Specificity = 0.85
   - Sensitivity = 0.125
   - Precision = 0.1429

c. See the charts below. The lift curve does not lie entirely above the baseline. The leftmost bar has a lift of 0. It suggests that none of the top 10 percent of the test cases with the highest predicted probabilities of belonging to the target class actually belong to the target class.

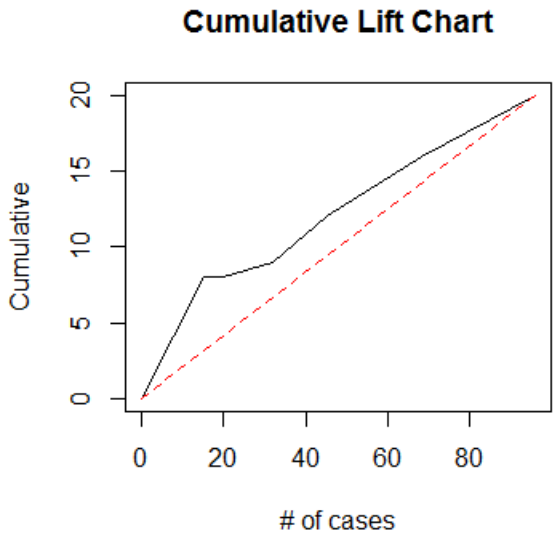


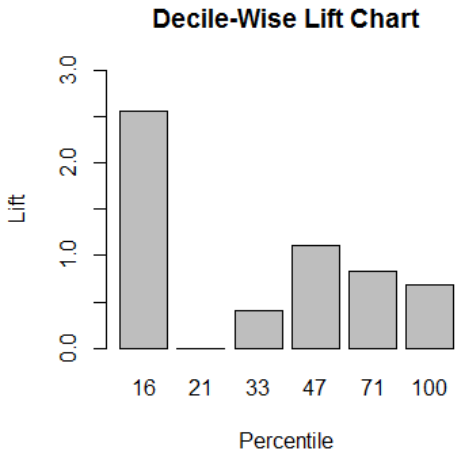


d. The AUC is 0.6328.

12_18. <R>

a. The optimal k is 10.
b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 79.17%
   - Specificity = 0.8947
   - Sensitivity = 0.4
   - Precision = 0.5
c. See the charts below. The entire lift curve lies above the baseline. The leftmost bar has a lift of 2.56. It suggests that selecting the top 16 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the KNN classifier would identify 2.56 times as many target class cases as if the cases are randomly selected.





d. AUC = 0.6378

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a
library(caret)
library(gains)
library(pROC)
myData <- myData[, -1]
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Complication)
```

```
colnames(myData1)[4] <- 'Complication'
myData1$Complication<- as.factor(myData1$Complication)
set.seed(1)
myIndex<- createDataPartition(myData1$Complication, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Complication ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
#b
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Complication, positive =
'1')
#c
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
validationSet$Complication <-
as.numeric(as.character(validationSet$Complication))
gains_table <- gains(validationSet$Complication,
KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Complication))~c(
0, gains_table$cume.obs), xlab = "# of cases", ylab =
"Cumulative", main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Complication))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Complication),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
#d
roc_object <- roc(validationSet$Complication, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


12_19. <Analytic Solver>

a. The optimal value of k is 4.
b. The accuracy rates for k=3, k=4, and k=5 are 58.89%, 72.12%, and 67.78%, respectively.
c. Cutoff Value = .65 (Using the test data set)
   - Accuracy rate = 60%
   - Specificity = 0.6957
   - Sensitivity = 0.5405
   - Precision = 0.7407

The sensitivity values for cutoff values 0.35, 0.5, and 0.65 are 0.8919, 0.8378, and 0.5405, respectively. Therefore, the cutoff value of 0.35 does the best job identifying customers who end up subscribing the product.

d. The average probability is 0.52.
e. The AUC is 0.6821.

12_19. <R>

a. The optimal k is 8.
b. The accuracy rates for k=3, k=4, and k=5 are 0.6871, 0.6908, and 0.7128, respectively.
c. Cutoff Value = .65 (Using the validation data set)
   - Accuracy rate = 67.23%
   - Specificity = 0.5952
   - Sensitivity = 0.7143
   - Precision = 0.7639

   The sensitivity values for cutoff values 0.35, 0.5, and 0.65 are 0.9740, 0.8442, and 0.7143, respectively. Therefore, the cutoff value of 0.35 does the best job identifying customers who end up subscribing the product.
d. The average probability is 0.55.
e. AUC = 0.6909

R Code:

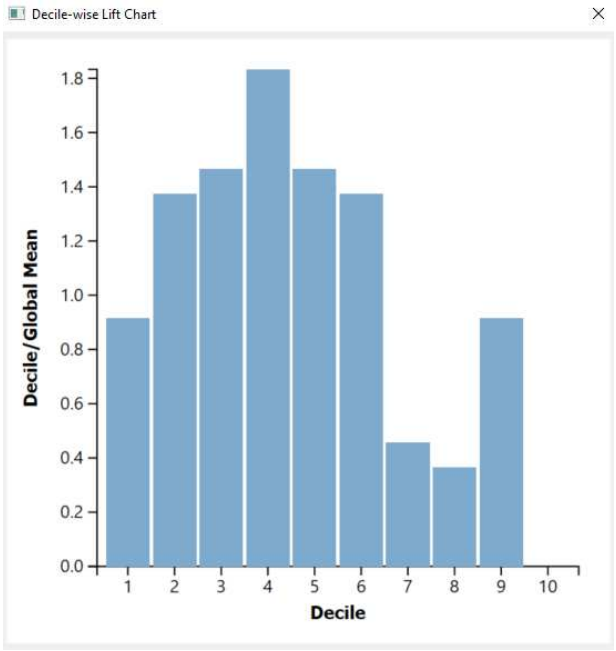
```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages(c("caret", "gains", "pROC"))
#a and b
library(caret)
library(gains)
library(pROC)
myData$Sex <- ifelse(myData$Sex == 'Female', 1, 0)
myData1<- scale(myData[2:4])
myData1<- data.frame(myData1, myData$Subscribe)
colnames(myData1)[4] <- 'Subscribe'
myData1$Subscribe<- as.factor(myData1$Subscribe)
set.seed(1)
myIndex<- createDataPartition(myData1$Subscribe, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Subscribe ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
```

```
KNN_fit
#c
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.65, '1',
'0')), as.factor(validationSet$Subscribe), positive = '1')
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.35, '1',
'0')), as.factor(validationSet$Subscribe), positive = '1')
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.5, '1',
'0')), as.factor(validationSet$Subscribe), positive = '1')
#d
PreProcessing <- preProcess(myData[ , 2:4], method = c("center",
"scale"))
myScoreData$Sex <- ifelse(myScoreData$Sex == "Female", 1, 0)
myScoreData1 <- predict(PreProcessing, myScoreData)
KNN_Score <- predict(KNN_fit, newdata=myScoreData1, type =
"prob")
KNN_Score
Avg_Prob <- mean(KNN_Score[,2])
Avg_Prob
#e
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$Subscribe, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


12_20. <Analytic Solver>

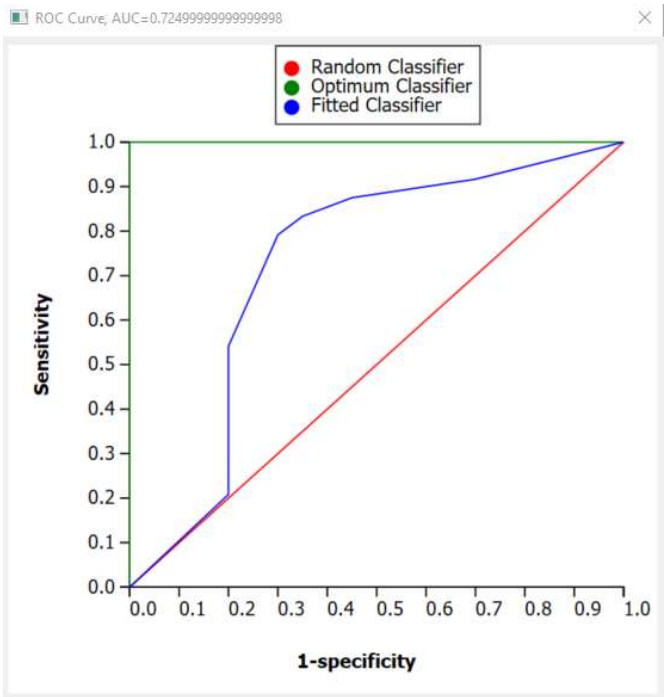
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 75%
   - Specificity = 0.7
   - Sensitivity = 0.7917
   - Precision = 0.76

b.



The lift value of the leftmost bar is 0.9167. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 0.9167 times as many target class cases as if the cases are randomly selected.

c.



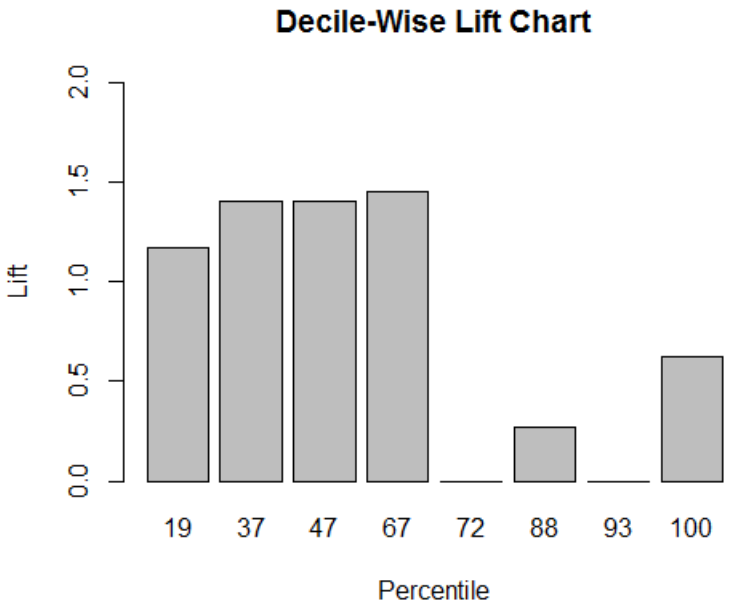
The AUC value is 0.725.

d. Using 0.5 as the cutoff value, the scoring results for the first three new observations are: Yes, No, Yes.

12_20. <R>

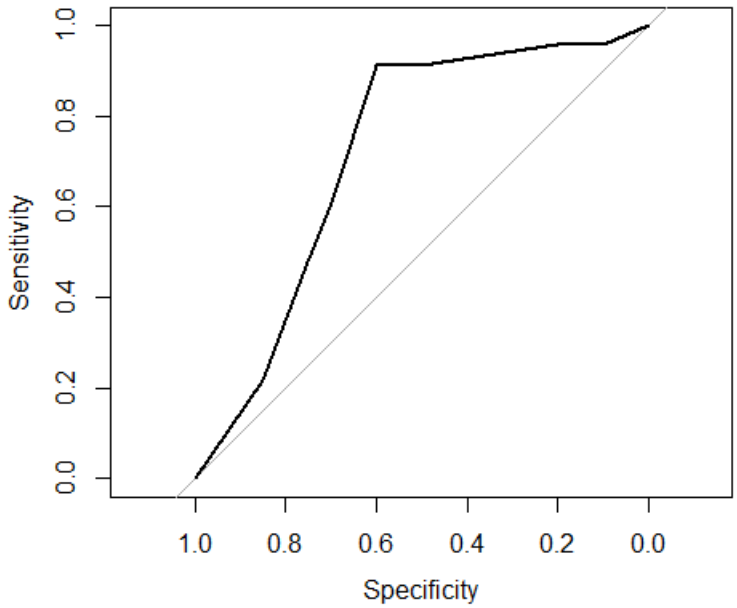
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7674
   - Specificity = 0.6
   - Sensitivity = 0.913
   - Precision = 0.7241

b.



The lift value of the leftmost bar is 1.17. This implies that by selecting the top 19% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.17 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.7196.

d. Using 0.5 as the cutoff value, the scoring results for the first three new observations are: Yes, No, Yes.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
```

```
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = 'Yes')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- ifelse(validationSet$y == "Yes", 1, 0)
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
```


12_21. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 40%
   - Specificity = 0.15
   - Sensitivity = 0.9
   - Precision = 0.3462

b. AUC = 0.6425

c. Using 0.5 as the cutoff value, the scoring results for the five new observations are: Y, N, Y, Y, Y.

12_21. <R>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.6333
   - Specificity = 0.6250
   - Sensitivity = 0.6429
   - Precision = 0.6

b. AUC = 0.5826

c. Using 0.5 as the cutoff value, the scoring results for the five new observations are: Y, N, Y, N, Y.

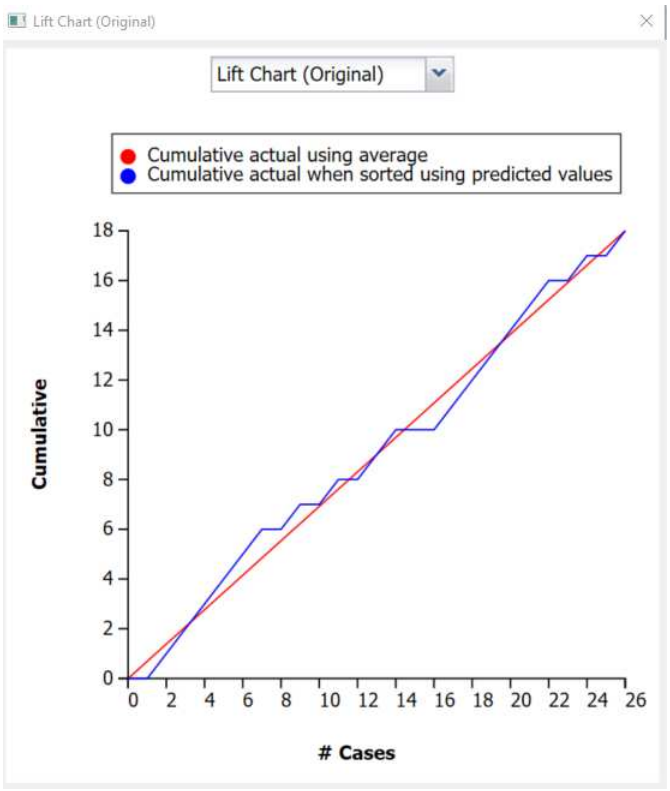
R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = 'Y')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
auc(roc_object)
#c
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
```


12_22. <Analytic Solver>

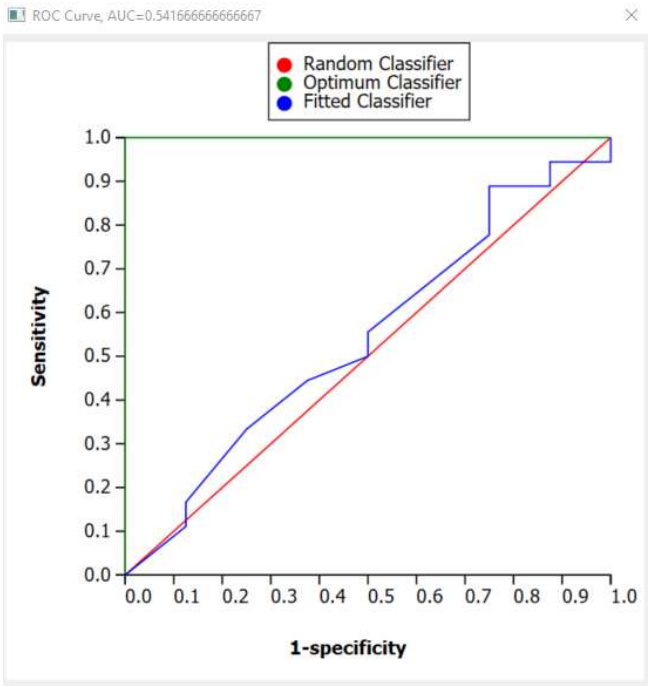
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 69.23%
   - Specificity = 0.25
   - Sensitivity = 0.8889
   - Precision = 0.7273

b.



No, the lift curve does not lie above the baseline entirely.

c.



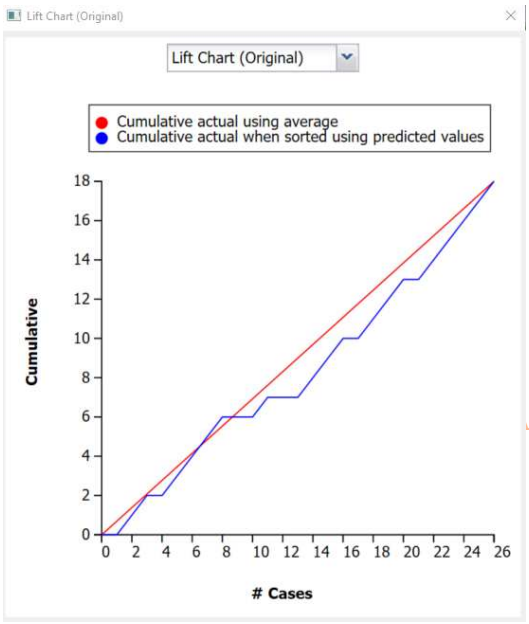
The AUC value is 0.5417.

d. Using 0.5 as the cutoff value, the scoring results for the five new observations are: 1, 0, 0, 1, 1.

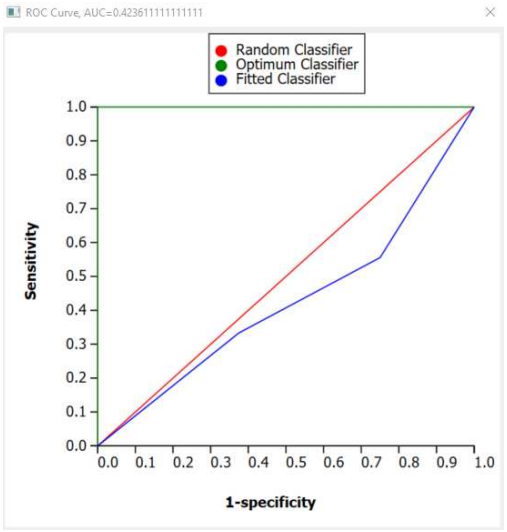
e. Using only x1 and x2 as the predictors, the model generates the following performance measures:

Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 46.15%
- Specificity = 0.25
- Sensitivity = 0.5556
- Precision = 0.625



No, the lift curve does not lie above the baseline entirely.



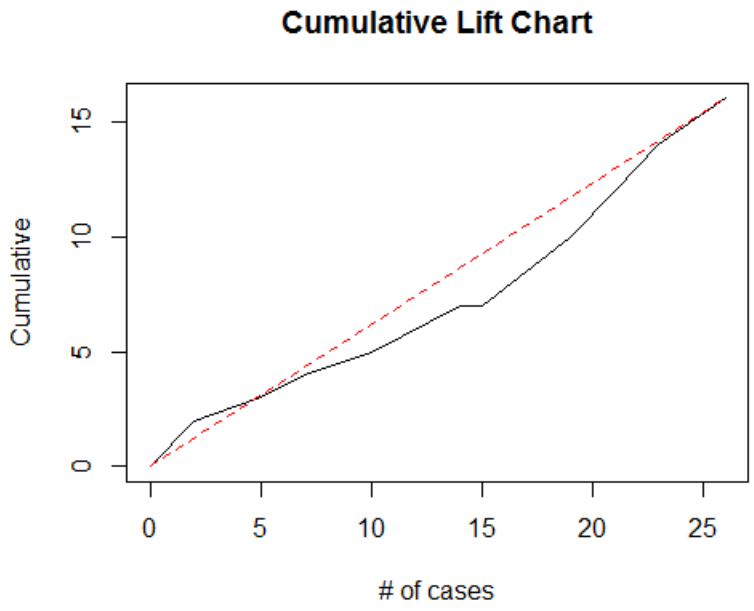
The AUC value is 0.4236.

All performance measures and graphs suggest that the naïve Bayes model that uses all four predictors performs slightly better than the model with only x_1 and x_2.

12_22. <R>

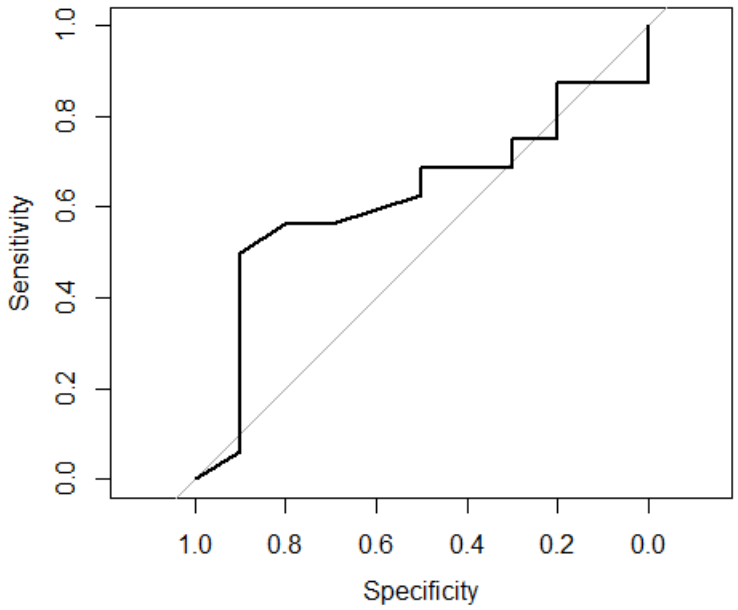
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.5
   - Specificity = 0.1
   - Sensitivity = 0.75
   - Precision = 0.5714

b.



No, the lift curve does not lie above the baseline entirely.

c.



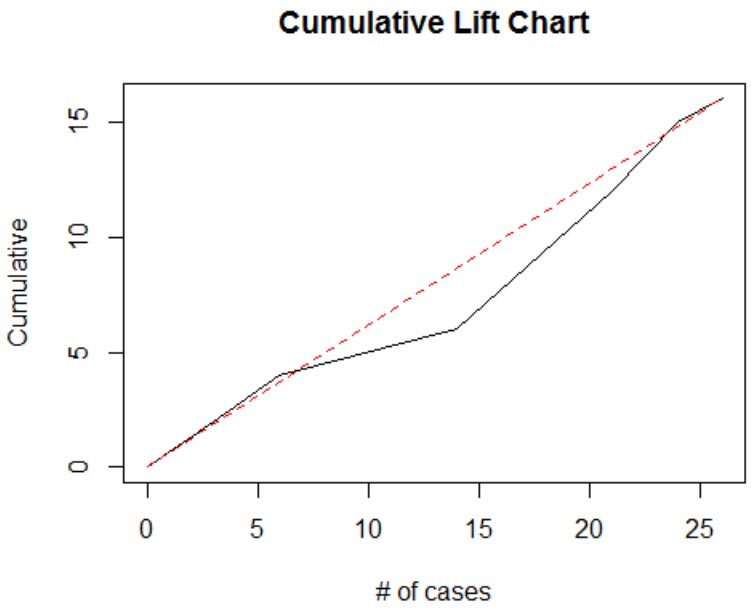
The AUC value is 0.6188.

d. Using 0.5 as the cutoff value, the scoring results for the five new observations are: 1, 1, 1, 0, 1.

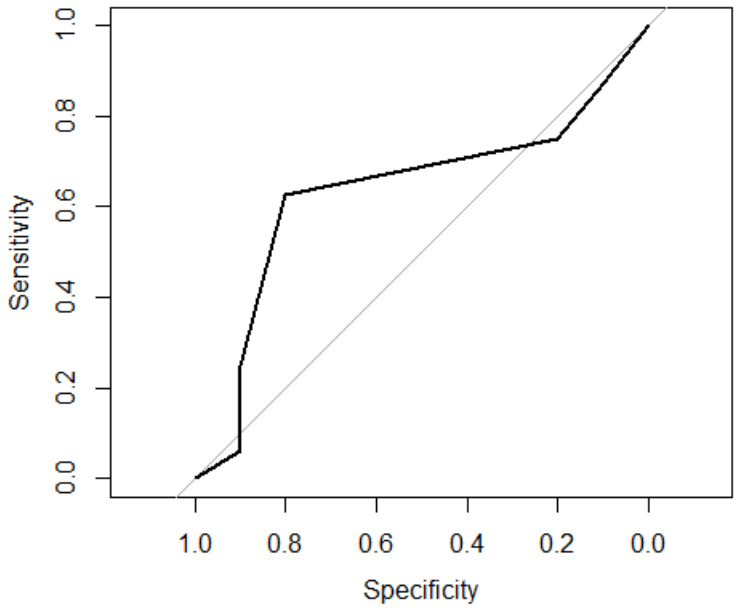
e. Using only x1 and x2 as the predictors, the model generates the following performance measures:

Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 0.6154
- Specificity = 0.1
- Sensitivity = 0.9375
- Precision = 0.625



No, the lift curve does not lie above the baseline entirely.



The AUC value is 0.6344.

All performance measures suggest that the naïve Bayes model that uses only x_1 and x_2 performs slightly better than the model that uses all four predictors.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
```

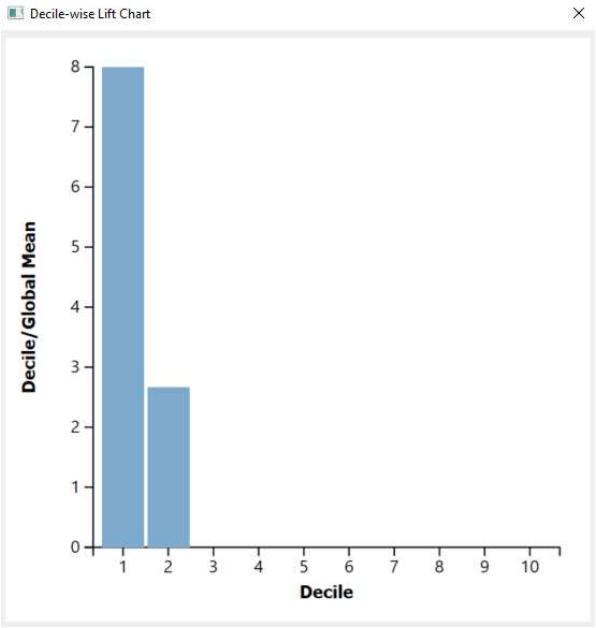
```
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
#c
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
#e Re-Read the data
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~ x1 + x2, data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
```

```
auc(roc_object)
```

12_23 <Analytic Solver>

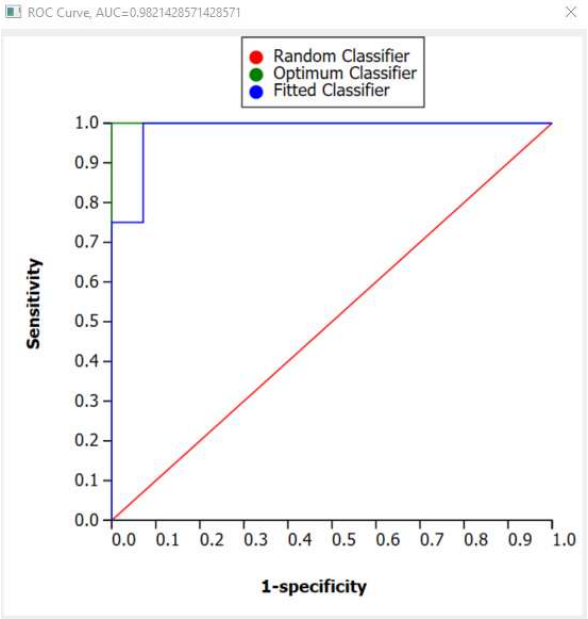
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 90.63%
   - Specificity = 0.8929
   - Sensitivity = 1
   - Precision = 0.5714

b.



The lift value of the leftmost bar is 8. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 8 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.9821.

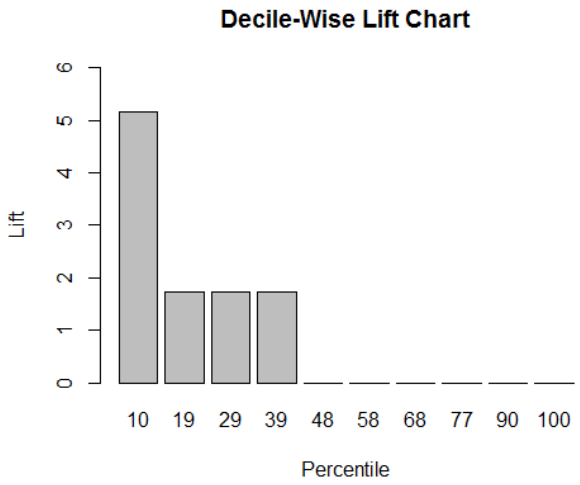
d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.9821, which is very close to the optimum model (AUC = 1), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 8 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.9063 with a sensitivity of 1 and a specificity of 0.8929. These accuracy measures are all very high. Given that there are 4 target class cases among the 32 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (32-4)/32 = 0.875 but with a sensitivity of 0. The naïve Bayes model shows better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. In conclusion, the naïve Bayes model is effective in classifying the data.

12_23. <R>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.9032
   - Specificity = 0.96

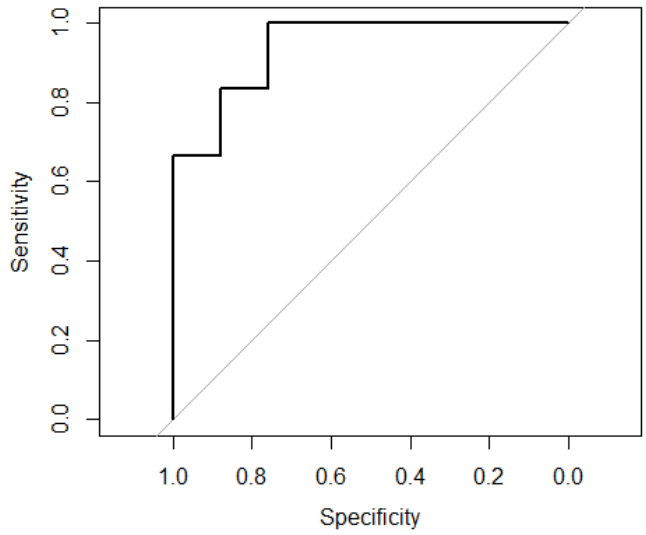
- Sensitivity = 0.6667
- Precision = 0.8

b.



The lift value of the leftmost bar is 5.17. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 5.17 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.9467.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.9467, which is very close to the optimum model (AUC = 1), suggesting that the model performs much better than the baseline model (AUC = 0.5) in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 5.17 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.9032 with a sensitivity of 0.6667 and a specificity of 0.96. These accuracy measures are all quite high. Given that there are 6 target class cases among the 31 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (31-6)/31 = 0.8065 but with a sensitivity of 0. The naïve Bayes model shows better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. In conclusion, the naïve Bayes model is effective in classifying the data.

R Code:

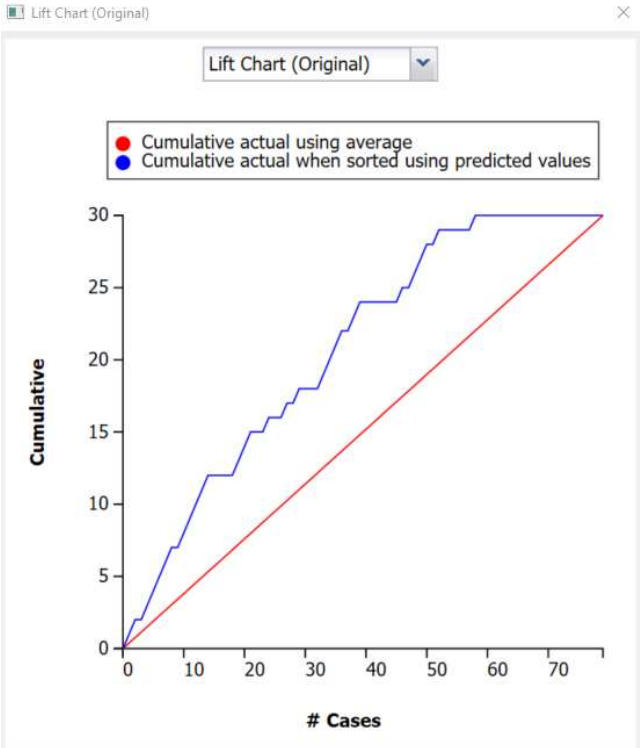
```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
```

```
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,6), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d No R code is needed for this question.
```

12_24. <Analytic Solver>

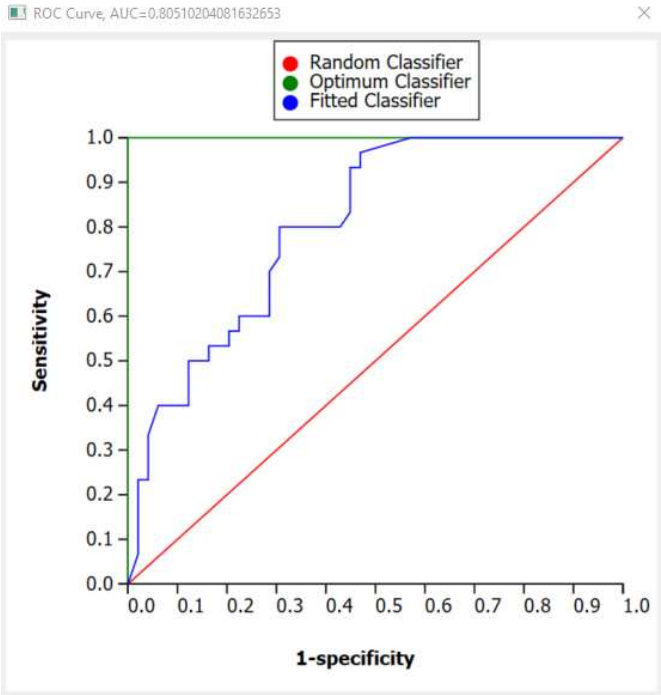
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 67.09%
   - Specificity = 0.7143
   - Sensitivity = 0.6
   - Precision = 0.5625

b.



Yes, the entire lift curve lies above the baseline.

c.



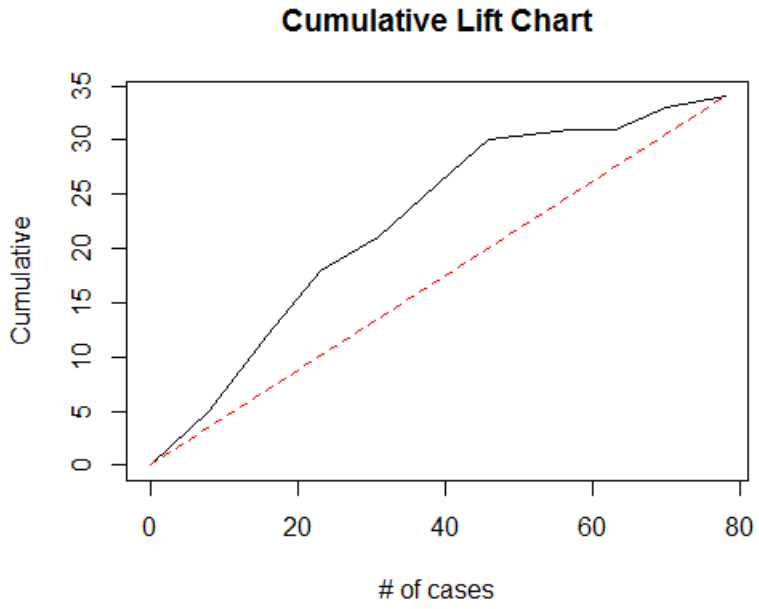
The AUC value is 0.8051.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8051, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.26 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 67.09% with a sensitivity of 0.6 and a specificity of 0.7143. Given that there are 30 target class cases among the 79 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (79-30)/79 = 0.6203 but with a sensitivity of 0. The naïve Bayes model shows better classification accuracy than the naïve rule. In conclusion, the naïve Bayes model is effective in classifying the data.

12_24. <R>

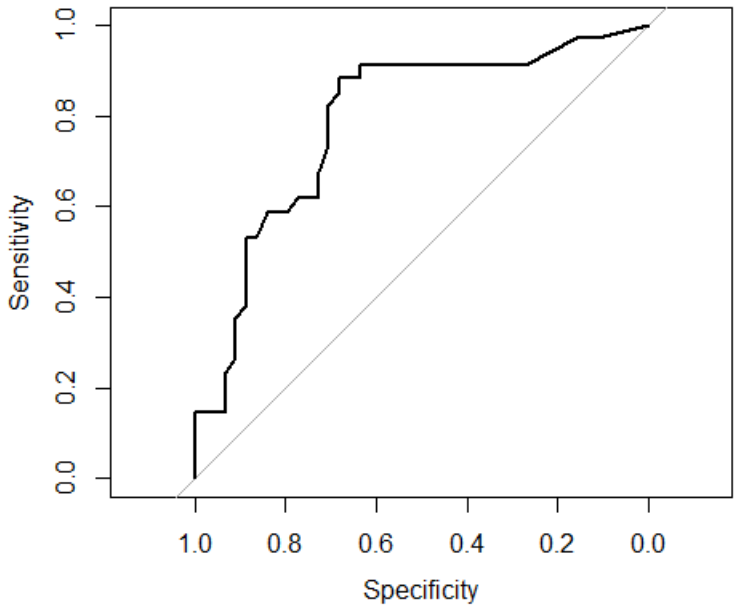
a. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.7051
- Specificity = 0.7273
- Sensitivity = 0.6765
- Precision = 0.6571

b.



Yes, the entire lift curve lies above the baseline.

c.



The AUC value is 0.7871.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7871, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7051 with a sensitivity of 0.6765 and a specificity of 0.7273. Given that there are 34 target class cases among the 78 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (78-34)/78 = 0.5641 but with a sensitivity of 0. The naïve Bayes model shows a moderately better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

R Code:

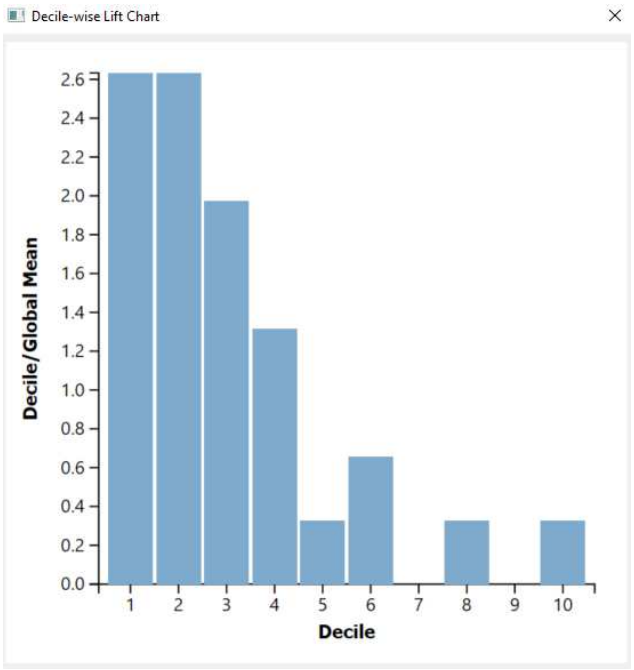
```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
```

```
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
#c
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d No R code is needed for this question.
```


12_25. <Analytic Solver>

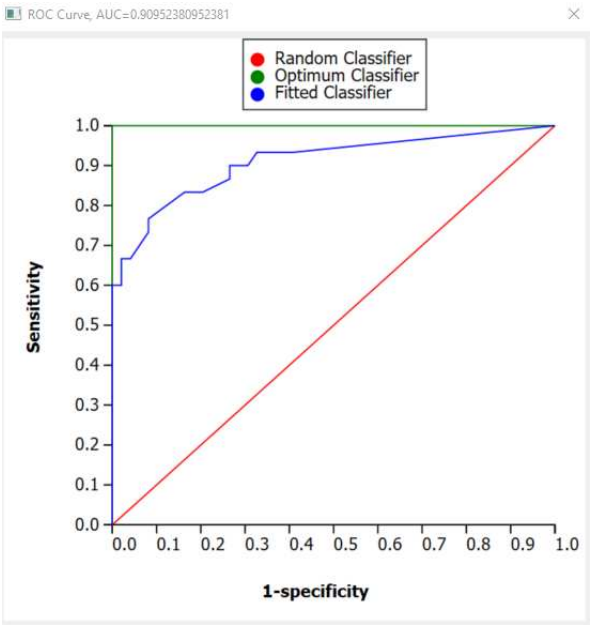
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 86.08%
   - Specificity = 0.9184
   - Sensitivity = 0.7667

b.



The lift value of the leftmost bar is 2.63.

c.



The AUC value is 0.9095.

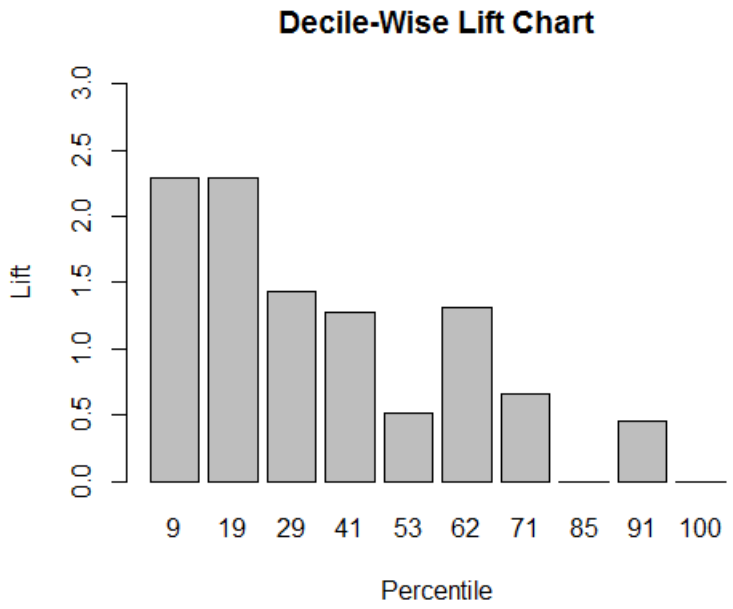
d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the

diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.9095, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.63 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 86.08% with a sensitivity of 0.7667 and a specificity of 0.9184. Given that there are 30 target class cases among the 79 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (79-30)/79 = 0.6203 but with a sensitivity of 0. The naïve Bayes model shows a much better overall classification accuracy than the naïve rule. In addition, large percentages of both target and non-target class cases are correctly classified. In conclusion, the naïve Bayes model is effective in classifying the data.

12_25. <R>

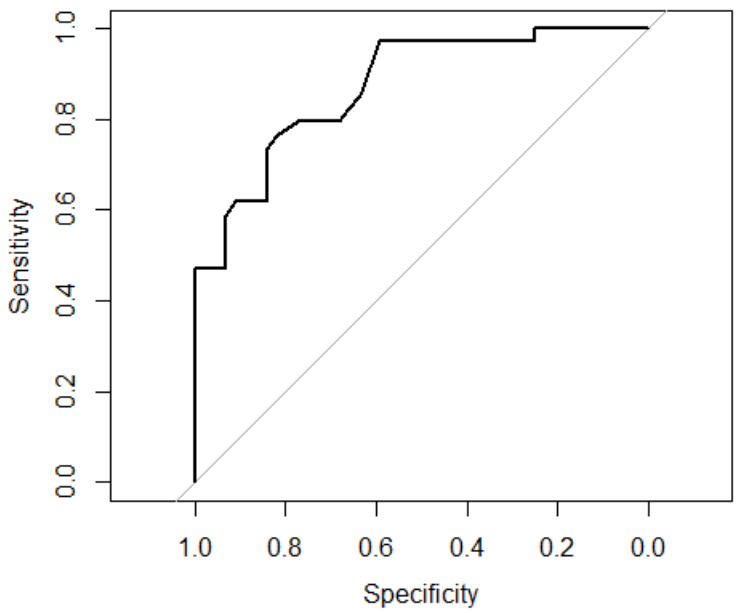
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7949
   - Specificity = 0.8182
   - Sensitivity = 0.7647
   - Precision = 0.7647

b.



The lift value of the leftmost bar is 2.29.

c.



The AUC value is 0.8723.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the

baseline model (random classifier) and is able to identify a larger percentage of target class (1) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8723, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.39 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7949 with a sensitivity of 0.7647 and a specificity of 0.8182. Given that there are 34 target class cases among the 78 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (78-34)/78 = 0.5641 but with a sensitivity of 0. The naïve Bayes model shows a much better overall classification accuracy than the naïve rule. In addition, large percentages of both target and non-target class cases are correctly classified. In conclusion, the naïve Bayes model is effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
```

```
gains_table <- gains(validationSet$y, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$y))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$y))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$y),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d No R code is needed for this question.
```

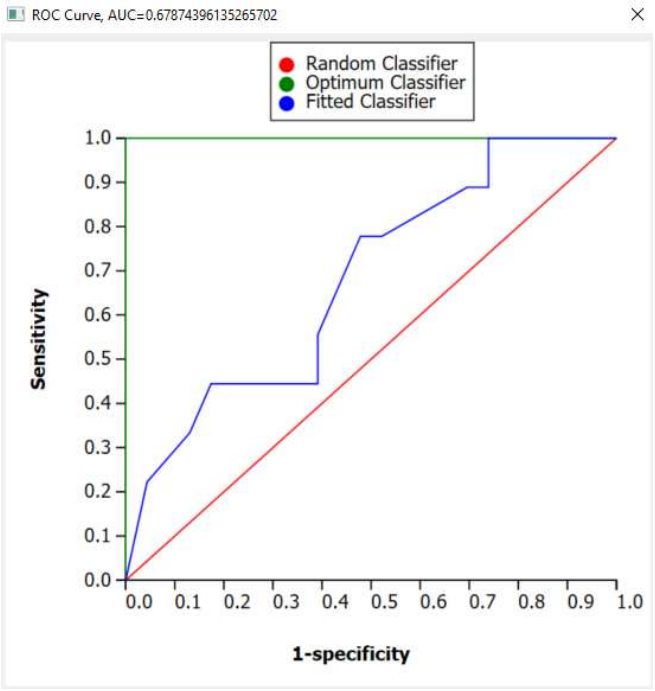
12_26. <Analytic Solver>

a.

	X1	X2	X3
Observation 1	1	2	1
Observation 2	1	2	1

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 68.75%
- Specificity = 0.7826
- Sensitivity = 0.4444
- Precision = 0.4444

c.



The AUC value is 0.6787

d. Cutoff Value = .20 (Using the validation data set)
   - Accuracy rate = 37.5%
   - Specificity = 0.1304
   - Sensitivity = 1
   - Precision = 0.3103

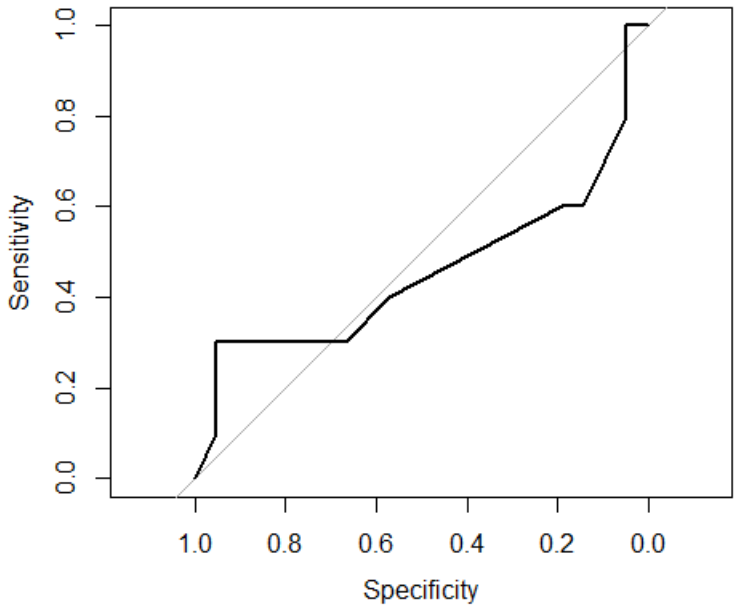
12_26. <R>

a.

	X1	X2	X3
Observation 1	1	2	1
Observation 2	1	2	1

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.6774
   - Specificity = 0.8571
   - Sensitivity = 0.3
   - Precision = 0.5

c.



The AUC value is 0.4548.

d. Cutoff Value = .20 (Using the validation data set)
   - Accuracy rate = 0.2903
   - Specificity = 0.1429
   - Sensitivity = 0.6
   - Precision = 0.25

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
myData$x1 <- cut(myData$x1, breaks = c(0, 6, 14, 30),labels =
c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
myData$x2 <- cut(myData$x2, breaks = c(0, 10, 20, 61),labels =
c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
```

```
myData$x3 <- cut(myData$x3, breaks = c(0, 3, 5, 10),labels =
c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
confusionMatrix(as.factor(ifelse(nb_Class_prob[,2]>0.2, '1',
'0')), as.factor(validationSet$y), positive = '1')
```

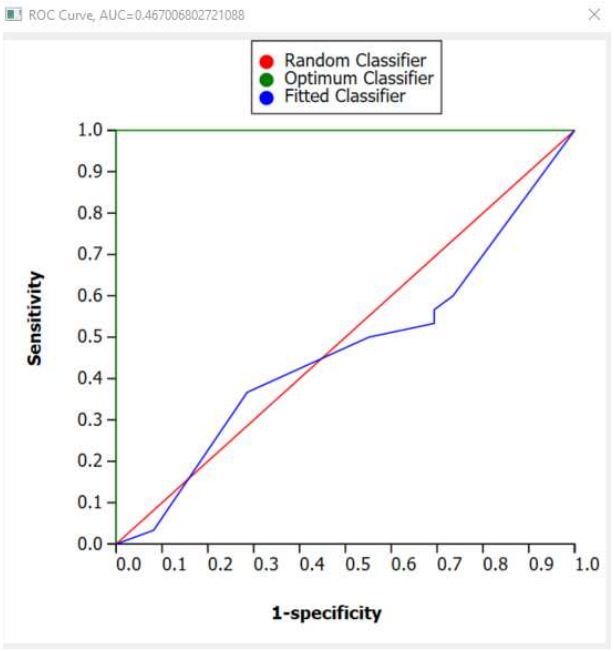

12_27. <Analytic Solver>

a.

	X1	X2
Observation 1	1	1
Observation 2	2	2

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 40.51%
- Specificity = 0.3061
- Sensitivity = 0.5667
- Precision = 0.3333

c.



The AUC value is 0.4670.

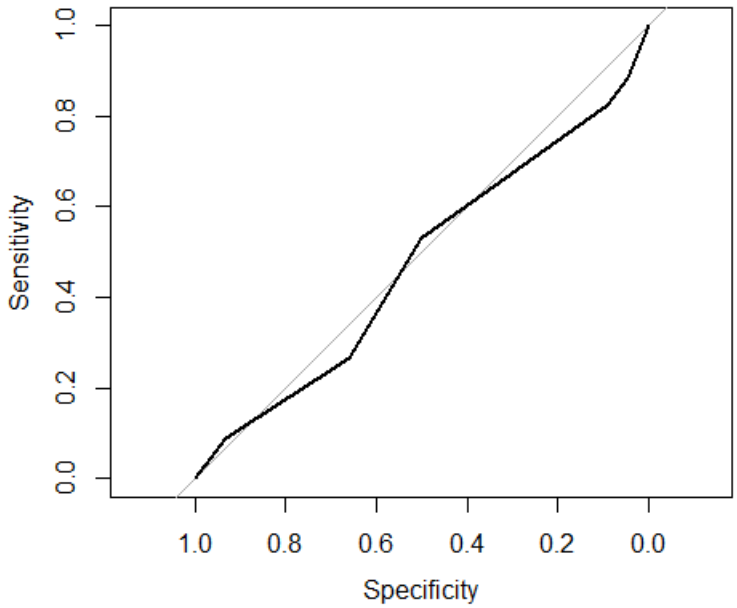
12_27. <R>

a.

	X1	X2
Observation 1	1	1
Observation 2	2	2

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.5641
- Specificity = 1
- Sensitivity = 0
- Precision = NaN

c.



The AUC value is 0.4726.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
myData$x1 <- cut(myData$x1, breaks = c(0, 60, 400, 30000),labels
= c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
myData$x2 <- cut(myData$x2, breaks = c(0, 160, 400, 800),labels =
c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
```

```
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```

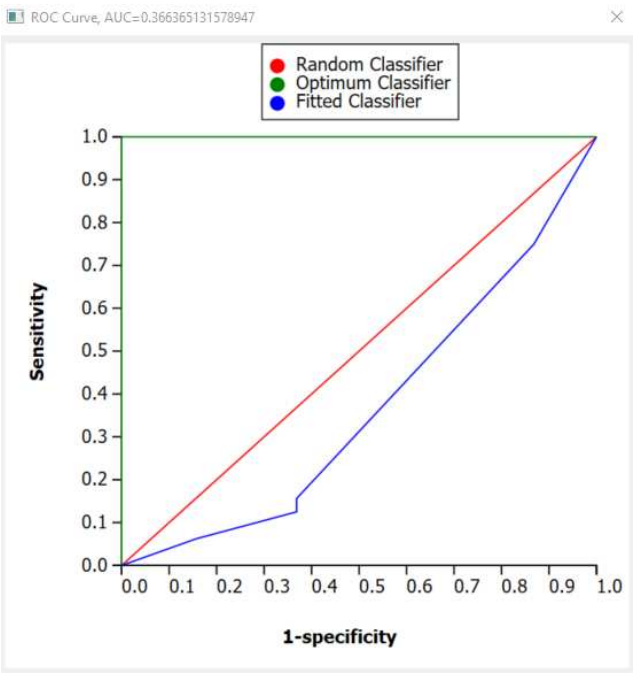
12_28. <Analytic Solver>

a.

	X1	X2	X3
Observation 1	1	1	2
Observation 2	1	1	1

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 48.57%
- Specificity = 0.8421
- Sensitivity = 0.0625
- Precision = 0.25

c.



The AUC value is 0.3664.

d. Cutoff Value = .40 (Using the validation data set)
- Accuracy rate = 41.43%
- Specificity = 0.1316
- Sensitivity = 0.75
- Precision = 0.4211

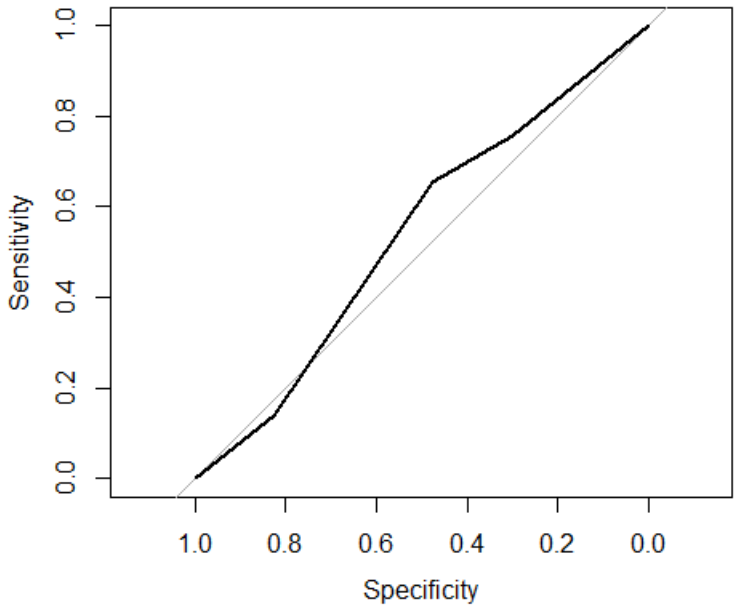
12_28. <R>

a.

	X1	X2	X3
Observation 1	1	1	2
Observation 2	1	1	1

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.5797
- Specificity = 1
- Sensitivity = 0
- Precision = NaN

c.



The AUC value is 0.5384.

d. Cutoff Value = .40 (Using the validation data set)
   - Accuracy rate = 0.5507
   - Specificity = 0.4750
   - Sensitivity = 0.6552
   - Precision = 0.4750

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
myData$x1 <- cut(myData$x1, breaks = c(0, 125, 250),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
```

```
myData$x2 <- cut(myData$x2, breaks = c(0, 30, 60),labels = c("1",
"2"), right = FALSE, include.lowest = TRUE)
myData$x3 <- cut(myData$x3, breaks = c(0, 30, 60),labels = c("1",
"2"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
confusionMatrix(as.factor(ifelse(nb_Class_prob[,2]>0.4, '1',
'0')), as.factor(validationSet$y), positive = '1')
```

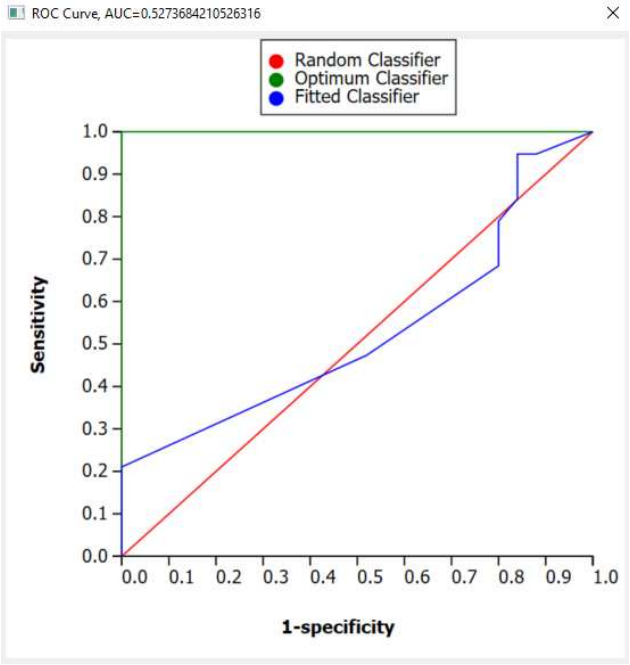

12_29. <Analytic Solver>

a.

	X1	X2	X3	X4
Observation 1	2	2	2	2
Observation 2	2	2	2	2

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 47.73%
- Specificity = 0.48
- Sensitivity = 0.4737
- Precision = 0.4091

c.



The AUC value is 0.5274.

d. Based on the performance measures, we conclude that the naïve Bayes model is not effective. Using 0.5 as the cutoff rate, the model shows low predictive accuracy. A large portion of the ROC curve lies below the baseline suggesting that the model performs slightly worse than the baseline model in terms of sensitivity and specificity for some of the possible cutoff values. The AUC value is 0.5274, which is about the same as that of the baseline model (AUC = 0.5).

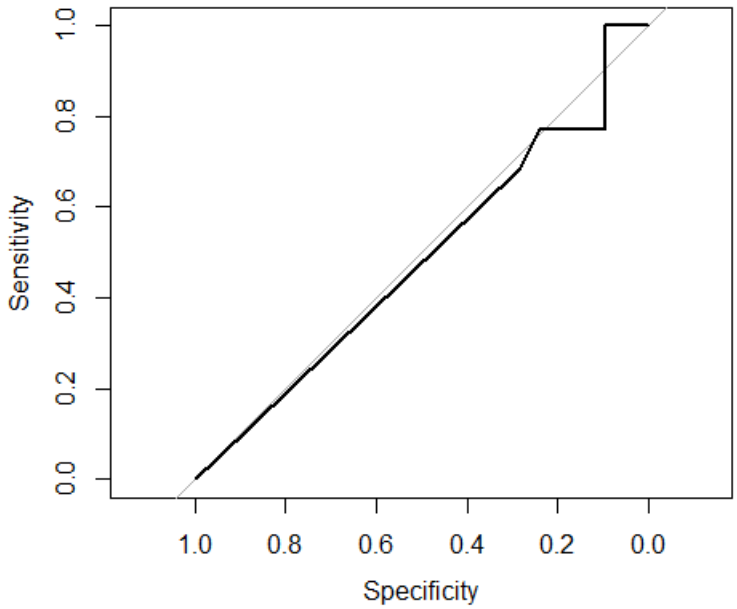
12_29. <R>

a.

	X1	X2	X3	X4
Observation 1	2	2	2	2
Observation 2	2	2	2	2

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.4419
- Specificity = 0.0952
- Sensitivity = 0.7727
- Precision = 0.4772

c.



The AUC value is 0.4838.

d. Based on the performance measures, we conclude that the naïve Bayes model is not effective. Using 0.5 as the cutoff rate, the model shows low predictive accuracy. The ROC curve lies mostly below the baseline suggesting that the model performs slightly worse than the baseline model in terms of sensitivity and specificity across all possible cutoff values.

R Code:

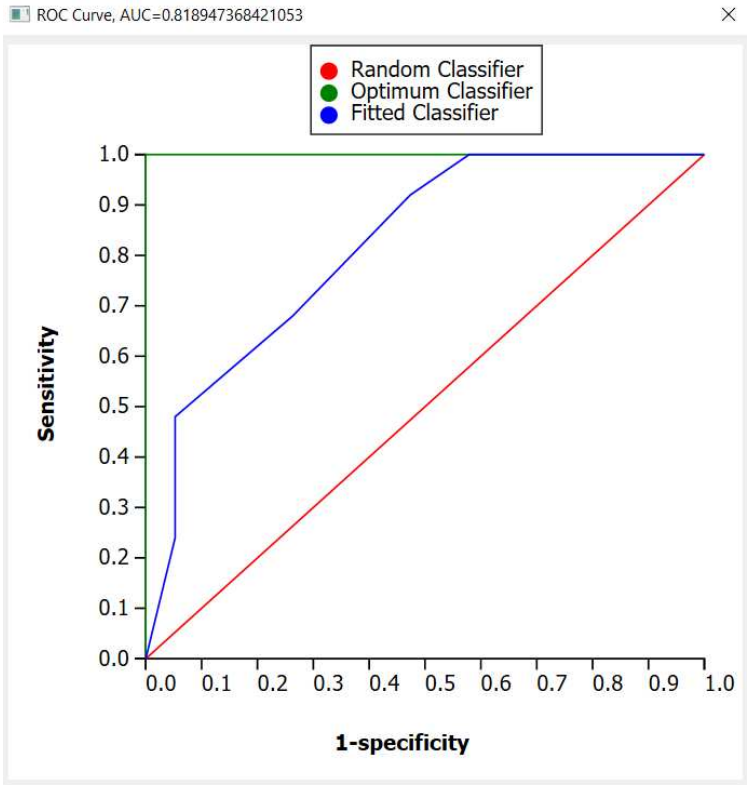
```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$y <- as.factor(myData$y)
myData$x1 <- cut(myData$x1, breaks = c(0, 40000, 80000),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
```

```
myData$x2 <- cut(myData$x2, breaks = c(0, 50, 100),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
myData$x3 <- cut(myData$x3, breaks = c(50, 75, 100),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
myData$x4 <- cut(myData$x4, breaks = c(0, 20000, 40000),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$y, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(y ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$y, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$y <- as.numeric(as.character(validationSet$y))
roc_object <- roc(validationSet$y, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


12_30. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 75%
   - Specificity = 0.5263
   - Sensitivity = 0.92
   - Precision = 0.7188

b.



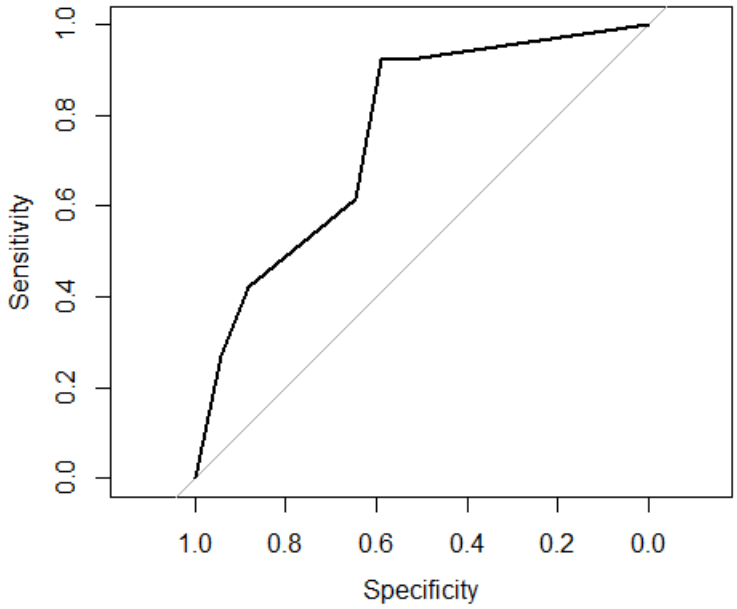
The AUC value is 0.8189.

c. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (international students who are accepted by US graduate programs) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8189, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.32 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.75 with a sensitivity of 0.92 and a specificity of 0.5263. Given that there are 25 target class cases among 44 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 25/44 = 0.5682 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is effective in identifying international students who are accepted by the US graduate programs.

12_30. <R>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.7907
   - Specificity = 0.5882
   - Sensitivity = 0.9231
   - Precision = 0.7742

b.



The AUC value is 0.759.

c. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (international students who are accepted by US graduate programs) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.759, which is slightly closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 19 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.45 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7907 with a sensitivity of

0.9231 and a specificity of 0.5882. Given that there are 26 target class cases among 43 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 26/43 = 0.6047 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. Although the accuracy measures are relatively high, they are affected by the cutoff value we use. Therefore, we place more emphasis on performance measures that are not dependent on cutoff values such as the AUC, ROC curve, lift and decile-wise charts, which all suggest moderately good performance of the naïve Bayes model. In conclusion, the naïve Bayes model is deemed moderately effective in classifying the data.

R Code:

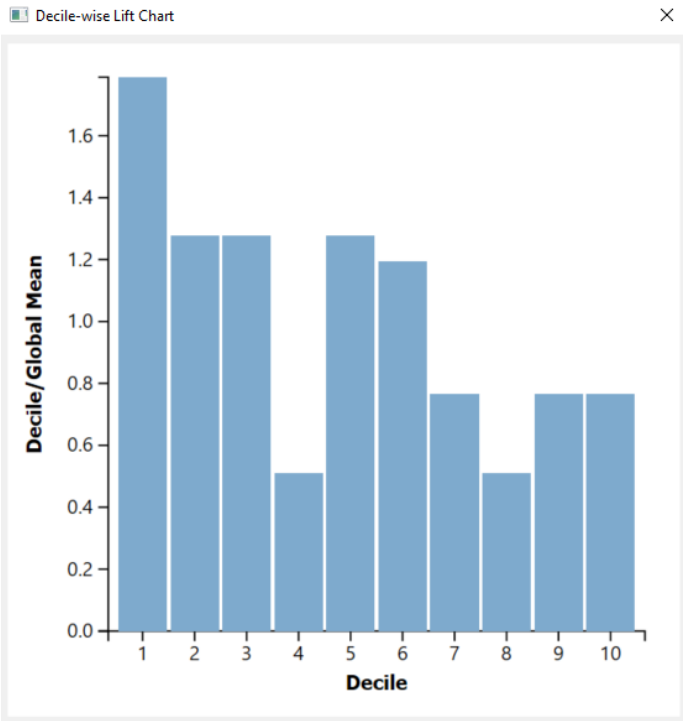
```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Accept <- as.factor(myData$Accept)
set.seed(1)
myIndex<- createDataPartition(myData$Accept, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Accept ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Accept, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Accept <-
as.numeric(as.character(validationSet$Accept))
roc_object <- roc(validationSet$Accept, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#c
gains_table <- gains(validationSet$Accept, nb_Class_prob[,2])
gains_table
```

```
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Accept))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Accept))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Accept),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
```

12_31. <Analytic Solver>

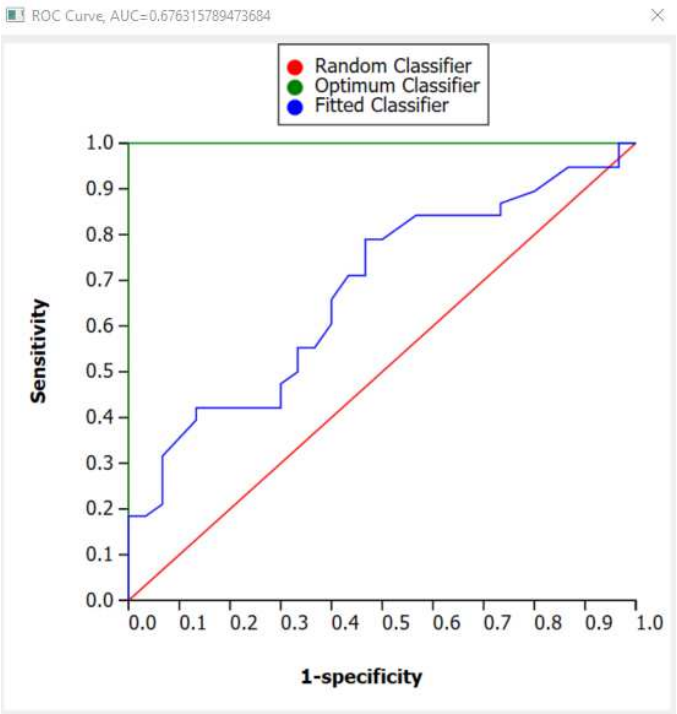
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 63.24%
   - Specificity = 0.5333
   - Sensitivity = 0.7105
   - Precision = 0.6585

b.



The lift value of the leftmost bar is 1.79. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.79 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.6763.

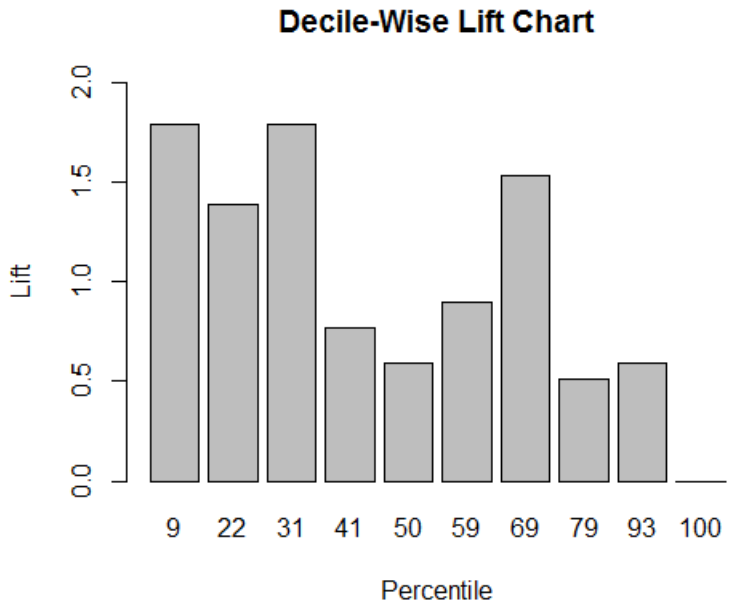
d. The performance measurements and graphs suggest that the naïve Bayes classifier is slightly effective in classifying the data. The lift curve of the naïve Bayes model lies mostly above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers who purchase) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6763, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs a little better than the baseline model in terms of sensitivity and specificity overall. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.79 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 63.24% with a sensitivity of 0.7105 and a specificity of 0.5333. Given that there are 38 target class cases among 68 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 38/68 = 0.5588 but specificity of 0. The naïve Bayes model performs slightly better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is slightly effective in identifying customers who would purchase.

12_31. <R>

a. Cutoff Value = .50 (Using the validation data set)

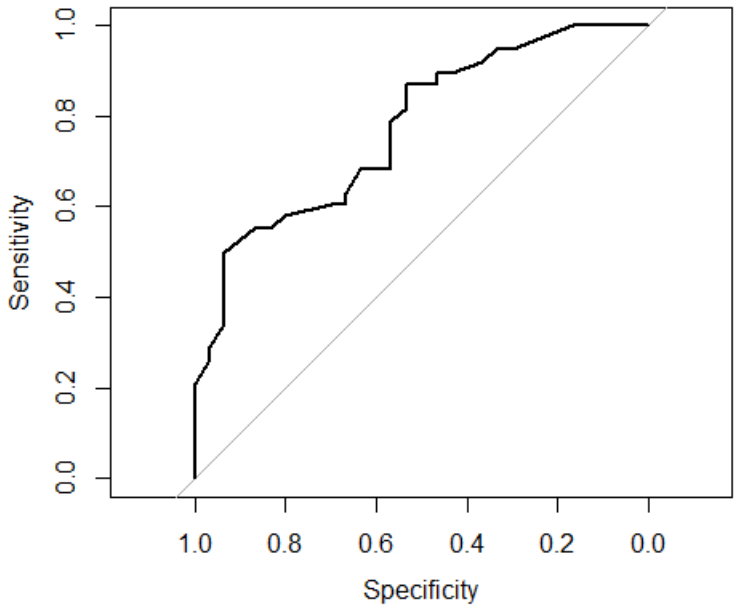
- Accuracy rate = 0.7059
- Specificity = 0.4667
- Sensitivity = 0.8947
- Precision = 0.68

b.



The lift of the leftmost bar of the decile-wise chart is 1.79. This implies that by selecting the top 9% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.79 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.7719.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers who purchase) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7719, which is slightly closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity overall. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 9 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.79 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7059 with a sensitivity of 0.8947 and a specificity of 0.4667. Given that there are 38 target class cases among 68 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 38/68 = 0.5588 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is moderately effective in identifying customers who would purchase.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
```

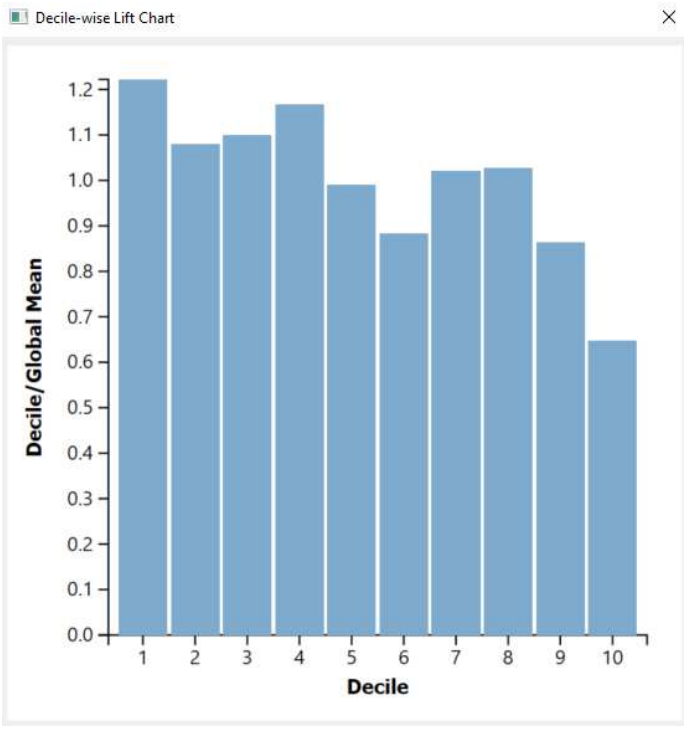
```
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Purchase <- as.factor(myData$Purchase)
set.seed(1)
myIndex<- createDataPartition(myData$Purchase, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Purchase ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Purchase, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Purchase <-
as.numeric(as.character(validationSet$Purchase))
gains_table <- gains(validationSet$Purchase, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$Purchase),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$Purchase, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Purchase))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Purchase))~c(0,
dim(validationSet)[1]), col="red", lty=2)
```


12_32. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 70.39%

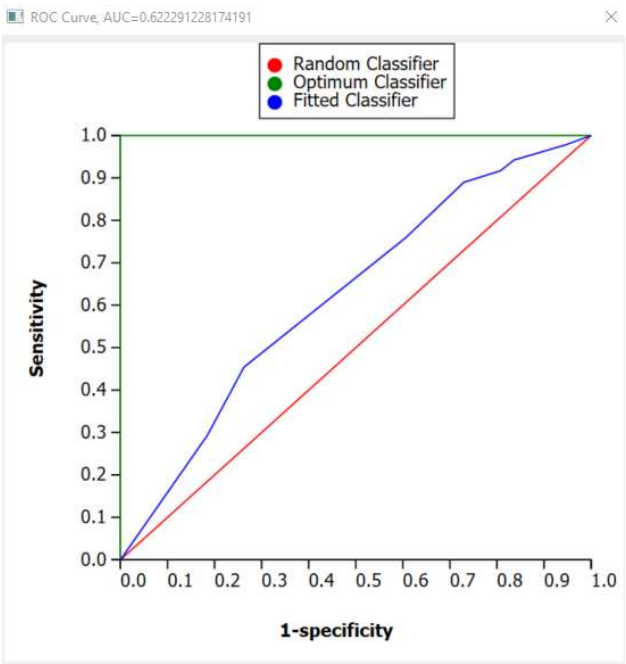
- Specificity = 0.1639
- Sensitivity = 0.9421
- Precision = 0.7186

b.



The lift value of the leftmost bar is 1.22. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.22 times as many target class cases as if the cases are randomly selected.

c.



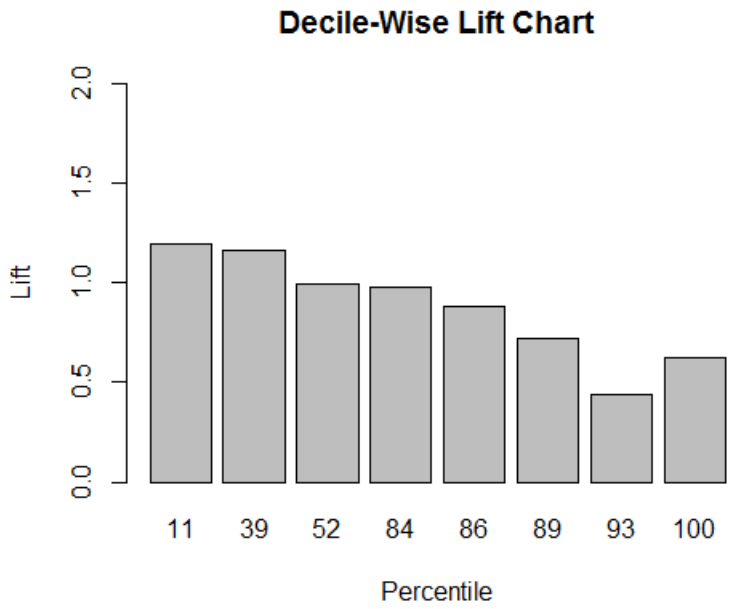
The AUC value is 0.6223.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is slightly effective in classifying the data. The lift curve of the naïve Bayes model lies slightly above the diagonal line. This suggests that the naïve Bayes classifier performs a little better than the baseline model (random classifier) and is able to identify a slightly larger percentage of target class (applicants accepted into the top 10 medical schools) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6223, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs only slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.22 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 70.38% with a sensitivity of 0.9421 and a specificity of 0.1639. The relatively high overall accuracy rate and sensitivity is due to the fact that the model classifies very few applicants in the non-acceptance category using the cutoff value of 0.5. Given that there are 553 target class cases among 797 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 553/797 = 0.6939 but specificity of 0. The naïve Bayes model performs almost the same as the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is only slightly effective in classifying the data.

12_32. <R>

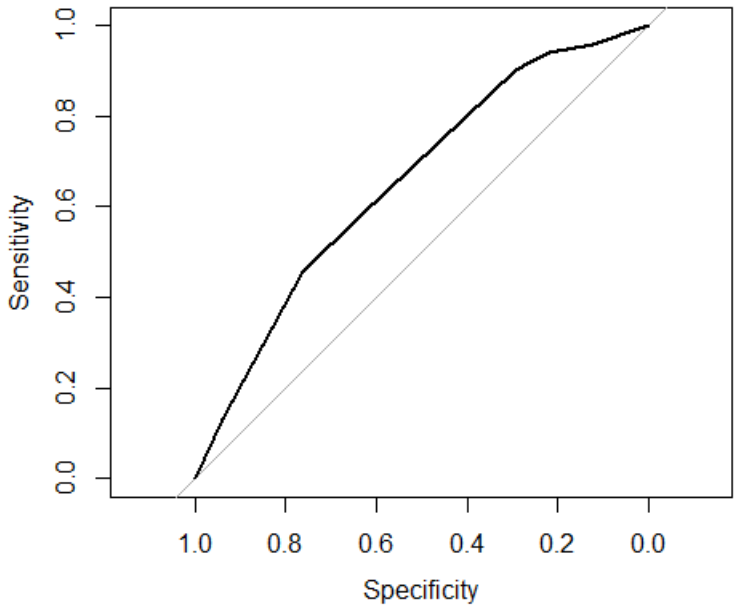
a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 69.35%
   - Specificity = 0.0
   - Sensitivity = 1
   - Precision = 0.6935

b.



The lift value of the leftmost bar is 1.20. This implies that by selecting the top 11% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.20 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.6549.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is only slightly effective in classifying the data. The lift curve of the naïve Bayes model lies slightly above the diagonal line. This suggests that the naïve Bayes classifier performs a little better than the baseline model (random classifier) and is able to identify a slightly larger percentage of target class (applicants accepted into the top 10 medical schools) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6549, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs only slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 11 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.2 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6935 with a sensitivity of 1 and a specificity of 0. The overall accuracy rate and sensitivity are inflated by the fact that the model classifies all applicants in the acceptance category using the cutoff value of 0.5. Given that there are 552 target class cases among 796 validation cases, the naïve rule (classifying all cases to the predominant class) would generate the same accuracy measures as the naïve Bayes model using the cutoff rate of 0.5. In conclusion, the naïve Bayes model is only slightly effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
```

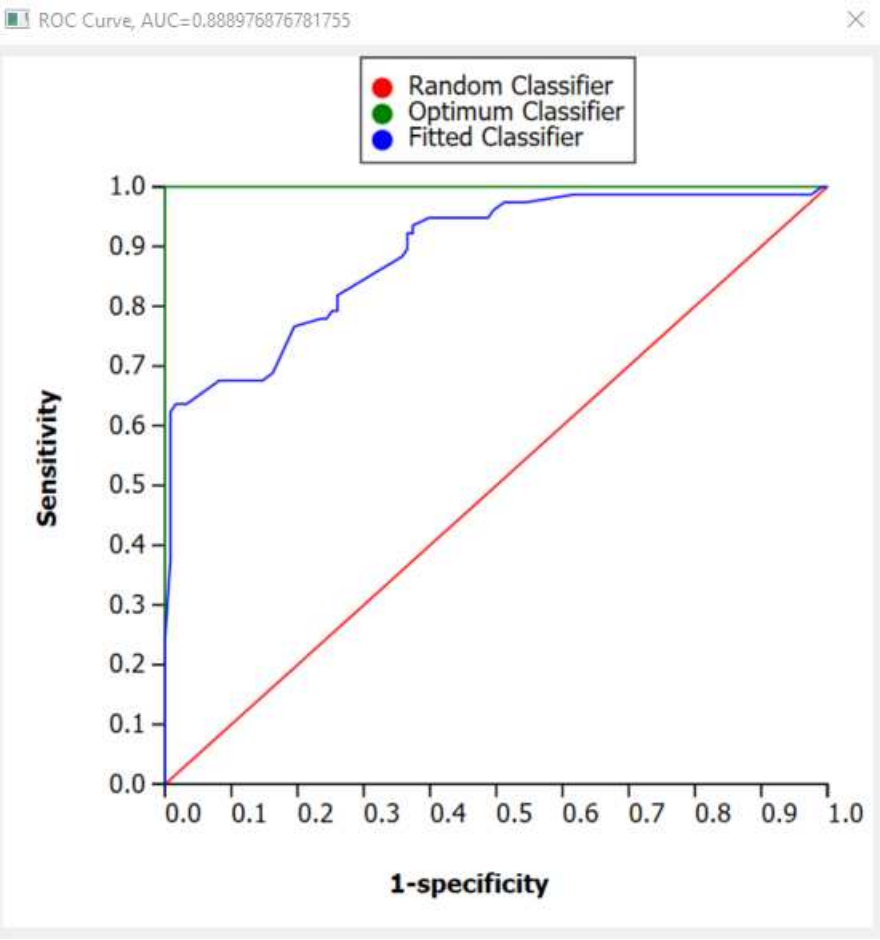
```
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Med <- as.factor(myData$Med)
set.seed(1)
myIndex<- createDataPartition(myData$Med, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Med ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Med, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Med <- as.numeric(as.character(validationSet$Med))
gains_table <- gains(validationSet$Med, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$Med),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$Med, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Med))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Med))~c(0, dim(validationSet)[1]),
col="red", lty=2)
```


12_33. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 82.5%
   - Specificity = 0.9187

- Sensitivity = 0.6753
- Precision = 0.8387

b.



The AUC value is 0.89.

c. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers with high credit card balance) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8890~~9000~~, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs much better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.6 times as many target class cases

as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 82.5% with a sensitivity of 0.6753 and a specificity of 0.9187. Given that there are 77 target class cases among 200 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy rate of (200-77)/200 = 0.615 but a sensitivity of 0. The naïve Bayes model performs much better than the naïve rule in terms of overall accuracy. In addition, the sensitivity value can be improved by reducing the cutoff value since the goal is to identify customers who are likely to carry a high credit card balance. In conclusion, the naïve Bayes model is effective in classifying the data.

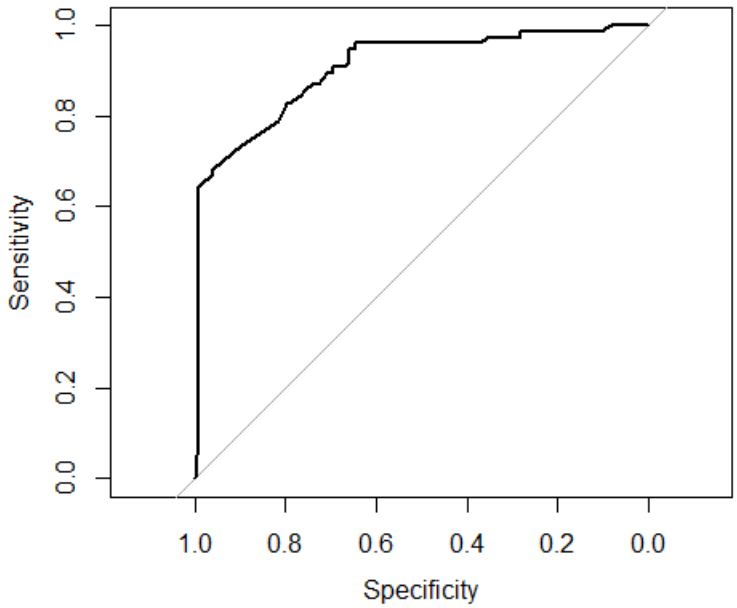
d. The scoring result of the first new customer is 1 (will likely carry a high credit card balance).

e. Cutoff Value = .30 (Using the validation data set)
   - Accuracy rate = 74.5%
   - Specificity = 0.6341
   - Sensitivity = 0.9221
   - Precision = 0.6121

12_33. <R>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.86
   - Specificity = 0.9919
   - Sensitivity = 0.6447
   - Precision = 0.98

b.



The AUC value is 0.9065.

c. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers with high credit card balance) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.9065, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs much better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.53 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.86 with a sensitivity of 0.6447 and a specificity of 0.9919. Given that there are 76 target class cases among 200 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy rate of (200-76)/200 = 0.62 but a sensitivity of 0. The naïve Bayes model performs much better than the naïve rule in terms of overall accuracy. In addition, the sensitivity value can be improved by reducing the cutoff value since the goal is to identify customers who are likely to carry a high credit card balance. In conclusion, the naïve Bayes model is effective in classifying the data.

d. The scoring result of the first new customer is 1 (will likely carry a high credit card balance).

e. Cutoff Value = .30 (Using the validation data set)
- Accuracy rate = 0.76
- Specificity = 0.6452
- Sensitivity = 0.9474
- Precision = 0.6207

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Balance <- as.factor(myData$Balance)
set.seed(1)
myIndex<- createDataPartition(myData$Balance, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Balance ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Balance, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Balance <-
as.numeric(as.character(validationSet$Balance))
roc_object <- roc(validationSet$Balance, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#c
gains_table <- gains(validationSet$Balance, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Balance))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
```

```
lines(c(0, sum(validationSet$Balance))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Balance),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
#d
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
#e
confusionMatrix(as.factor(ifelse(nb_Class_prob[,2]>0.3, '1',
'0')), as.factor(validationSet$Balance), positive = '1')
```

12_34. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 81.4%
   - Specificity = 0.9551
   - Sensitivity = 0.6593
   - Precision = 0.9305

b. Lift of the leftmost bar of decile-wise chart: 1.92
   AUC = 0.8298

c. The scoring results of the first three individuals are 0 (not likely to volunteer), 0, 1 (likely to volunteer).

12_34. <R>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.8184
   - Specificity = 0.9712
   - Sensitivity = 0.6517
   - Precision = 0.9541

b. Lift of the leftmost bar of the decile-wise chart: 1.98
   AUC = 0.8481

c. The scoring results of the first three individuals are 0 (not likely to volunteer), 0, 1 (likely to volunteer).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
```

```
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Volunteer <- as.factor(myData$Volunteer)
set.seed(1)
myIndex<- createDataPartition(myData$Volunteer, p=0.6,
list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Volunteer ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Volunteer, positive =
'1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Volunteer <-
as.numeric(as.character(validationSet$Volunteer))
gains_table <- gains(validationSet$Volunteer, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$Volunteer),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Volunteer, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
```


12_35. <Analytic Solver>

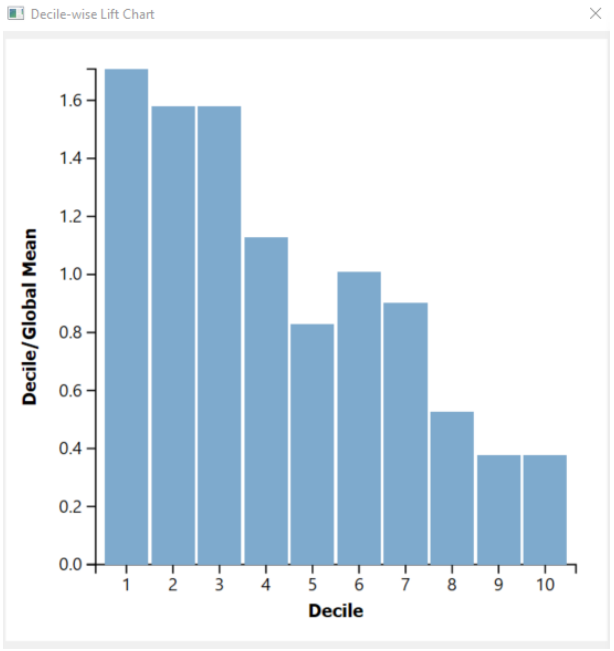
a.

	Age	Income
Observation 1	1	1
Observation 2	2	1

b. Cutoff Value = .50 (Using the validation data set)

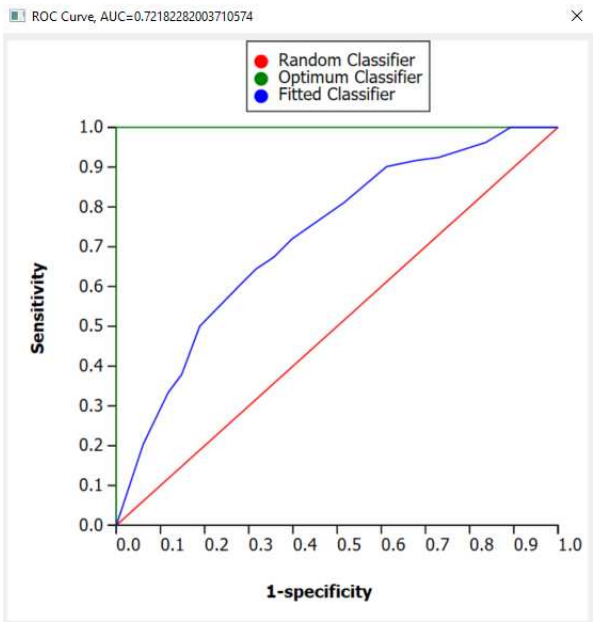
- Accuracy rate = 68.60%
- Specificity = 0.8122
- Sensitivity = 0.5
- Precision = 0.6408

c.



The lift value of the leftmost bar is 1.71.

d.



The AUC value is 0.7218.

e. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies well above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (individuals who rent a beachfront house). The AUC value is 0.7218, which is at about half way between the optimum model (AUC = 1) and the baseline model (AUC = 0.5), suggesting that the model is performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.71 times as many target class cases (individuals who rent a beachfront house) as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 68.6% with a sensitivity of 0.5 and a specificity of 0.8122. Given that there are 132 target class cases among the 328 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (328 - 132)/328 = 05976 but with a sensitivity of 0. The naïve Bayes model shows better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. While the sensitivity value of 0.5 is low, the analyst can choose to lower the cutoff value to increase the sensitivity value. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

12_35. <R>

a.

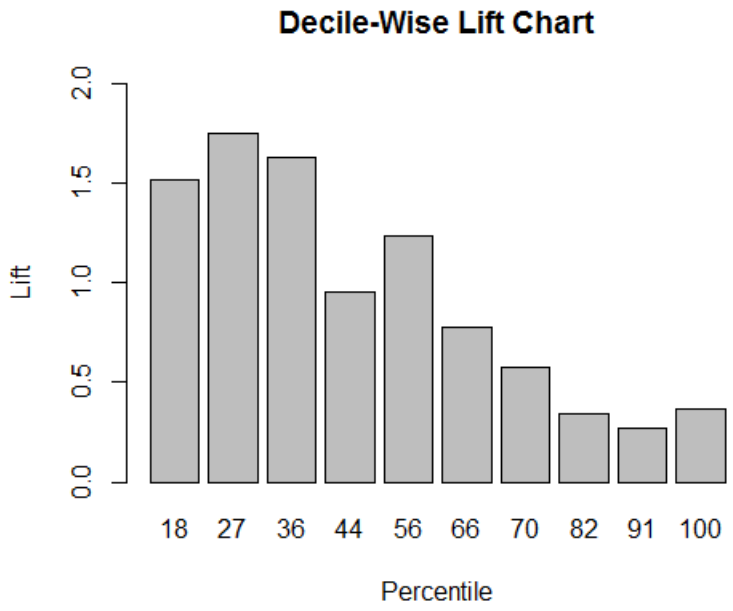
	Age	Income

Observation 1	1	1
Observation 2	2	1

b. Cutoff Value = .50 (Using the validation data set)

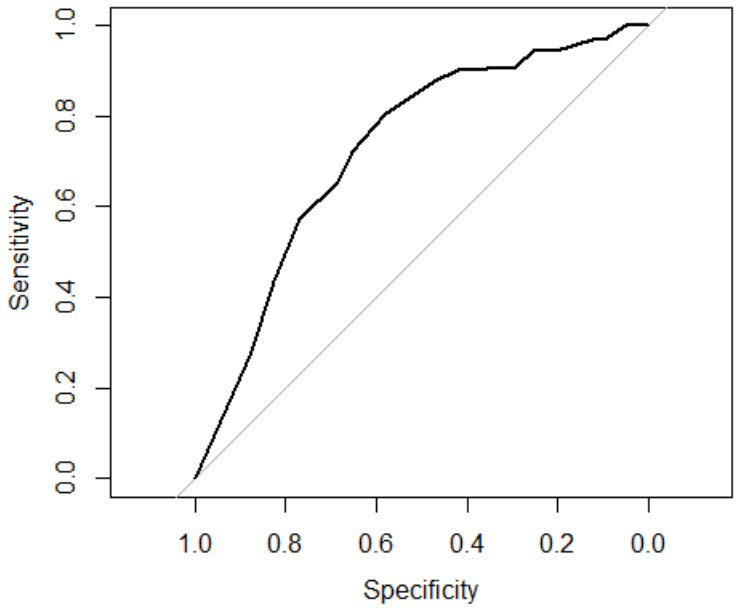
- Accuracy rate = 0.6789
- Specificity = 0.8244
- Sensitivity = 0.4344
- Precision = 0.5955

c.



The lift value of the leftmost bar is 1.52.

d.



The AUC value is 0.7264.

e. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (individuals who rent a beachfront house). The AUC value is 0.7264, which is about half way between the optimum model (AUC = 1) and the baseline model (AUC = 0.5), suggesting that the model is performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 18 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.52 times as many target class cases (individuals who rent a beachfront house) as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6789 with a sensitivity of 0.4344 and a specificity of 0.8244. Given that there are 122 target class cases among the 327 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (327 - 122)/327 = 0.6269 but with a sensitivity of 0. The naïve Bayes model shows better classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. While the sensitivity value of 0.4344 is low, the analyst can choose to lower the cutoff value to increase the sensitivity value. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Rental <- as.factor(myData$Rental)
myData$Age <- cut(myData$Age, breaks = c(22, 45, 85),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
myData$Income <- cut(myData$Income, breaks = c(0, 85000,
300000),labels = c("1", "2"),  right = FALSE, include.lowest =
TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$Rental, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Rental ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Rental, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Rental <-
as.numeric(as.character(validationSet$Rental))
gains_table <- gains(validationSet$Rental, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Rental))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Rental))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Rental),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
#d
roc_object <- roc(validationSet$Rental, nb_Class_prob[,2])
```

```
plot.roc(roc_object)
auc(roc_object)
#e
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Rental))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Rental))~c(0,
dim(validationSet)[1]), col="red", lty=2)
```

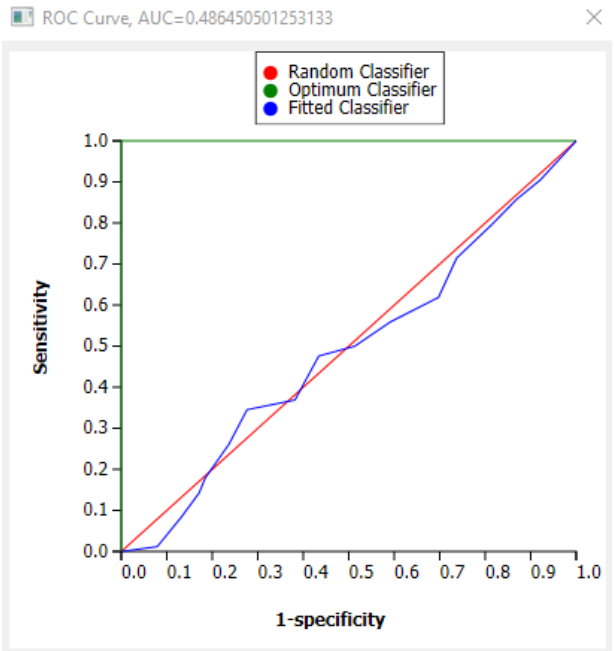
12_36. <Analytic Solver>

a.

	Age	Hours
Observation 1	1	2
Observation 2	1	1

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 48.75%
- Specificity = 0.4079
- Sensitivity = 0.5595
- Precision = 0.5109

c.



The AUC value is 0.4865.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is not effective in classifying the data. The lift curve of the naïve Bayes model lies mostly below the diagonal line. This suggests that the naïve Bayes classifier performs worse better than the baseline model (random classifier) in identifying target class (individuals who make in-game purchases) cases. The AUC value is 0.4865, which is slightly lower than that of the baseline model (AUC = 0.5), suggesting that the model performs worse than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify only 0.7143 times as many target class (individuals who make in-game purchases) cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.4875 with a sensitivity of 0.5595 and a specificity of 0.4079. All of these performance measures suggest low performance of the model. In conclusion, the naïve Bayes model is not effective in classifying the data.

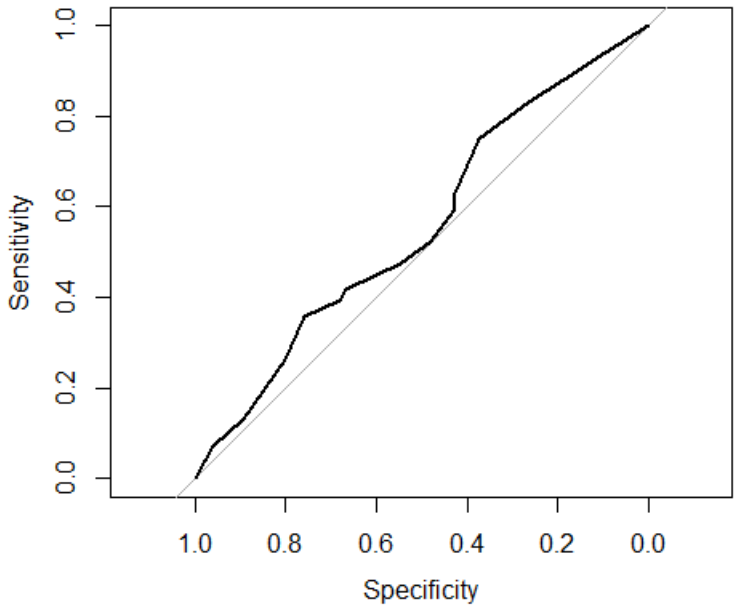
12_36. <R>

a.

	Age	Hours
Observation 1	1	2
Observation 2	1	1

b. Cutoff Value = .50 (Using the validation data set)
   a. Accuracy rate = 0.566
   b. Specificity = 0.28
   c. Sensitivity = 0.8214
   d. Precision = 0.5610

c.



The AUC value is 0.5562.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is not effective in classifying the data. The lift curve of the naïve Bayes model lies only slightly above the diagonal line. This suggests that the naïve Bayes classifier performs slightly better than the baseline model (random classifier). The AUC value is 0.5562, which is only slightly higher than that of the baseline model (AUC = 0.5), suggesting that the model performs about the same as the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 12 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.1 times as many target class (individuals who make in-game purchases) cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.566 with a sensitivity of 0.8214 and a specificity of 0.28. While the overall accuracy rate is not very high, the model is able to identify 82.14% of the target class cases correctly at the expense of misclassifying a large percentage of non-target class cases (specificity = 0.28). This is due to the fact that the naïve Bayes model classifies the great majority of the cases as target class cases (123 out of 159 validation cases). From a practical perspective, the naïve Bayes model does not provide a very targeted list of customers who are likely to make in-game purchases. Therefore, the model is not effective in classifying the data.

R Code:

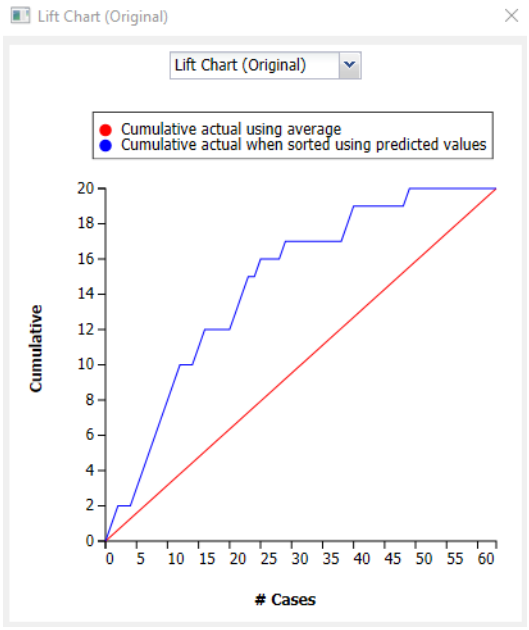
```
# Import the data file into a data frame (table) and label it
myData.
```

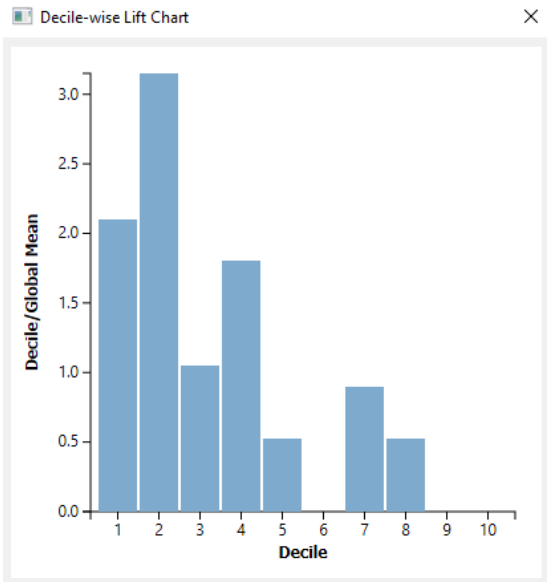
```
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Buy <- as.factor(myData$Buy)
myData$Age <- cut(myData$Age, breaks = c(15, 40, 65),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
myData$Hours <- cut(myData$Hours, breaks = c(0, 20, 40),labels =
c("1", "2"),  right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$Buy, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Buy ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Buy, positive = '1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Buy <- as.numeric(as.character(validationSet$Buy))
roc_object <- roc(validationSet$Buy, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
gains_table <- gains(validationSet$Buy, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Buy))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Buy))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Buy),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,4), main="Decile-Wise Lift Chart")
```

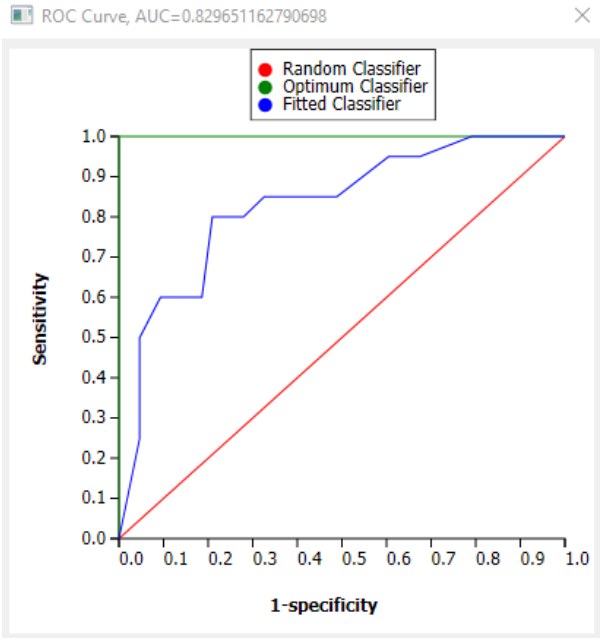
12_37. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 74.60%
   - Specificity = 0.8140
   - Sensitivity = 0.6
   - Precision = 0.6

b.







c. AUC = 0.8297

d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (employees who hold upper management positions) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8297, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.1 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7460 with a sensitivity of 0.6 and a specificity of 0.8140. Given that there are 20 target class cases among the 63 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (63 – 20)/63 = 0.6825 but with a sensitivity of 0. The naïve Bayes model shows better overall classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. The sensitivity value can be further improved if the analyst choose to lower the cutoff value to a number lower than the default 0.5. In conclusion, the naïve Bayes model is effective in classifying the data.

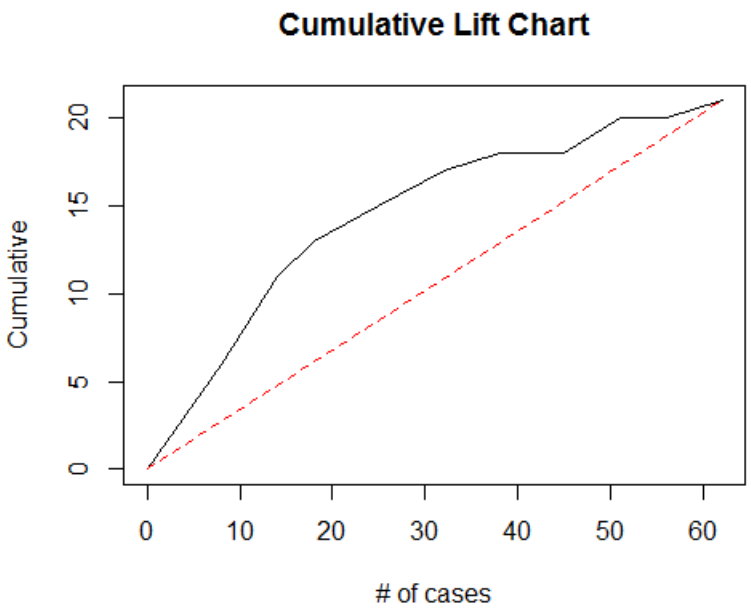
e. The predicted outcome of the first applicant is 0 (Will not hold an upper management position).

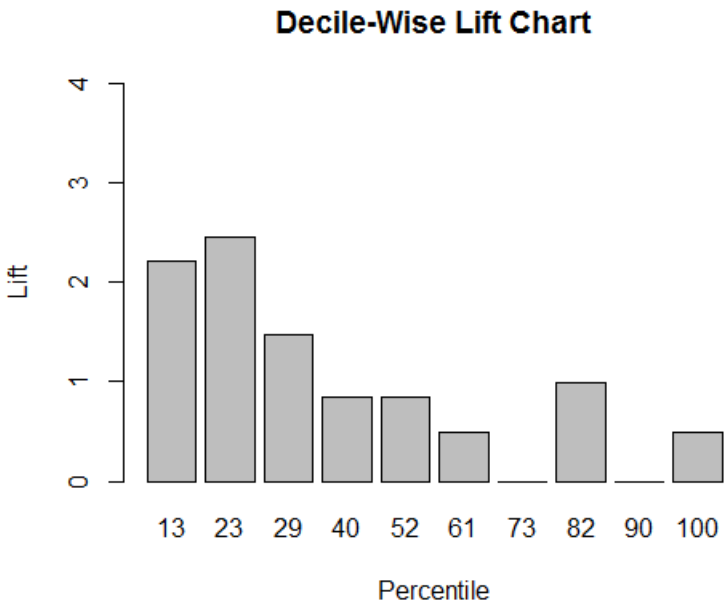
12_37. <R>

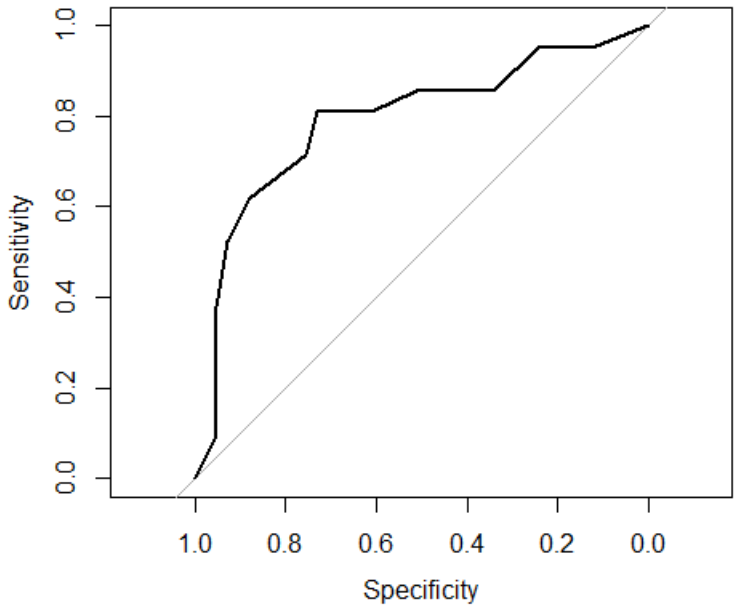
a. Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 0.7581
- Specificity = 0.9512
- Sensitivity = 0.3810
- Precision = 0.8

b.







c. AUC = 0.7909

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (employees who hold upper management positions) cases by looking at a small

percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7909, which is slightly closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 13 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.21 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7581 with a sensitivity of 0.381 and a specificity of 0.9512. Given that there are 21 target class cases among the 62 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of (62 – 21)/62 = 0.6613 but with a sensitivity of 0. The naïve Bayes model shows better overall classification accuracy than the naïve rule, especially if the goal of the classification is to identify target class cases. The sensitivity value can be further improved if the analyst choose to lower the cutoff value to a number lower than the default 0.5. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

e. The predicted outcome of the first applicant is 0 (Will not hold an upper management position).

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Success <- as.factor(myData$Success)
set.seed(1)
myIndex<- createDataPartition(myData$Success, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Success ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Success, positive = '1')
```

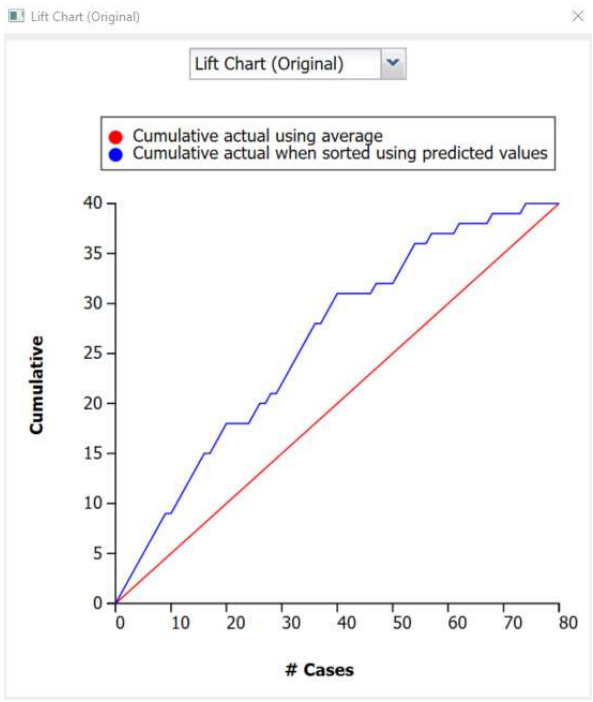
```
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Success <-
as.numeric(as.character(validationSet$Success))
gains_table <- gains(validationSet$Success, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Success))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Success))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Success),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,4), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Success, nb_Class_prob[,2])
plot.roc(roc_object)
#c
auc(roc_object)
#d No R code is needed for this question.
#e
nb_Score <- predict(nb_fit, newdata=myScoreData)
nb_Score
```


12_38. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)

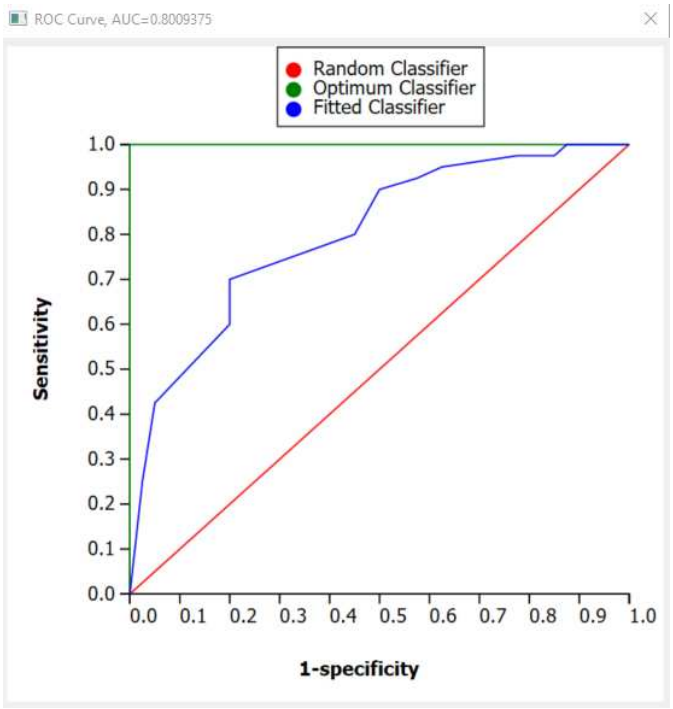
- Accuracy rate = 66.25%
- Specificity = 0.375
- Sensitivity = 0.95
- Precision = 0.6032

b.



Yes, the entire lift curve lies above the baseline.

c.



The AUC value is 0.8009.

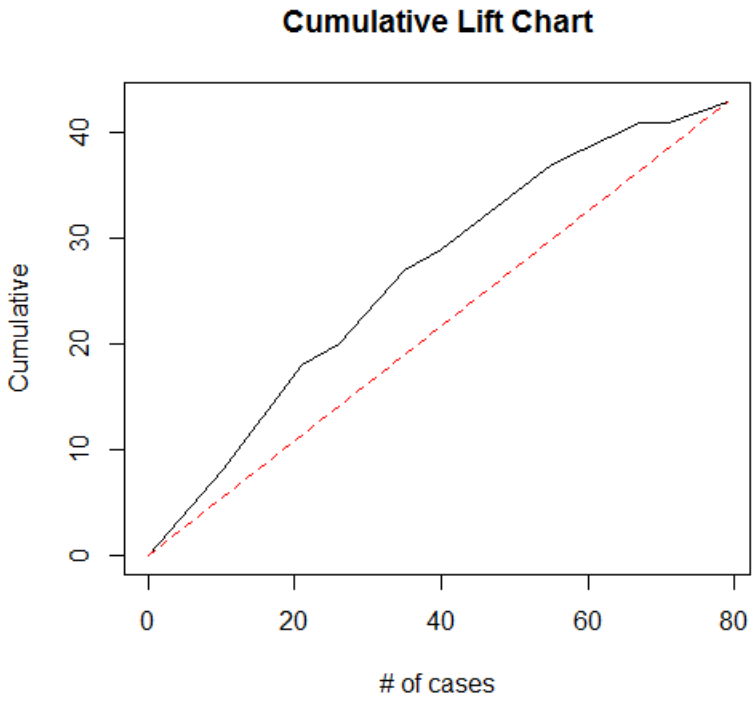
d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (students who graduate) cases by selecting a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8009, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs much better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify twice as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 66.25% with a sensitivity of 0.95 and a specificity of 0.375. Given that there are 40 target class cases among 80 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy rate of 40/80 = 0.5. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. Since the goal of the classification is likely to identify at-risk students who have high probabilities of not graduating (non-target class cases), the cutoff value may be increased in order to classify more cases into the non-target class to improve specificity, but that is done at the expense of a lower sensitivity measure. In conclusion, the naïve Bayes model is effective in classifying the data.

12_38. <R>

a. Cutoff Value = .50 (Using the validation data set)

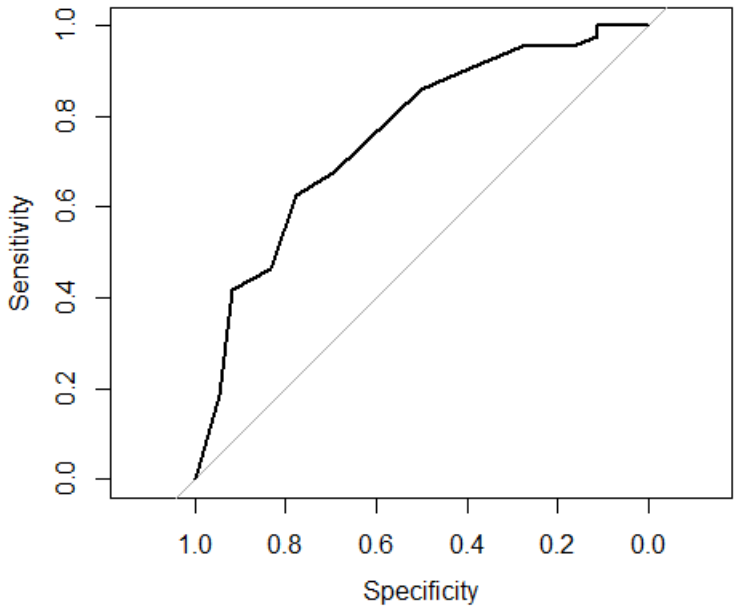
- Accuracy rate = 0.6835
- Specificity = 0.6944
- Sensitivity = 0.6744
- Precision = 0.725

b.



Yes, the entire lift curve lies above the baseline.

c.



The AUC value is 0.7565

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (students who graduate) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7565, which is slightly closer to the perfect model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 13 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.47 times as many target class cases (students who graduate) as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6835 with a sensitivity of 0.6744 and a specificity of 0.6944. Given that there are 43 target class cases among the 79 validation data cases, the naïve rule (classifying all cases to the predominant class) would have an accuracy rate of 43/79 = 0.5443 but with a specificity 0. The naïve Bayes model shows better classification accuracy than the naïve rule, especially since the goal of the classification is to identify at-risk students who have high probabilities of not graduating (non-target class cases). The specificity value can be further improved if the analyst choose to increase the cutoff value to a number higher than the default 0.5 to classify more cases into the non-target class, but that is done at the expense of a lower sensitivity measure. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Graduate <- as.factor(myData$Graduate)
set.seed(1)
myIndex<- createDataPartition(myData$Graduate, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
```

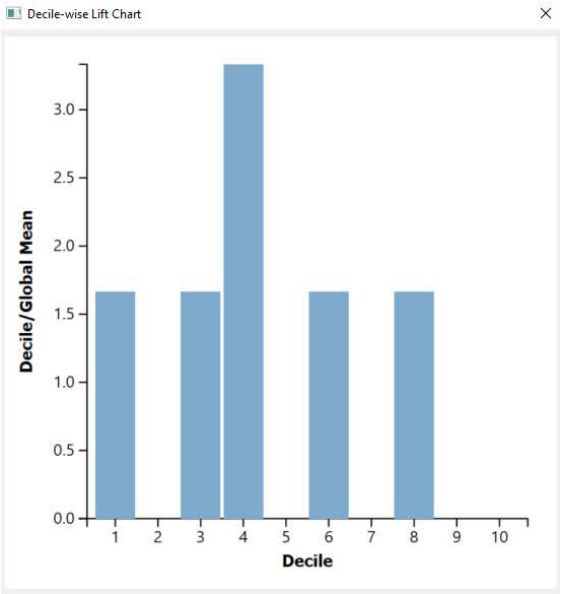
```
nb_fit <- train(Graduate ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Graduate, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Graduate <-
as.numeric(as.character(validationSet$Graduate))
gains_table <- gains(validationSet$Graduate, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Graduate))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Graduate))~c(0,
dim(validationSet)[1]), col="red", lty=2)
#c
roc_object <- roc(validationSet$Graduate, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
barplot(gains_table$mean.resp/mean(validationSet$Graduate),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,4), main="Decile-Wise Lift Chart")
```


12_39. <Analytic Solver>

a. Cutoff Value = .50 (Using the validation data set)

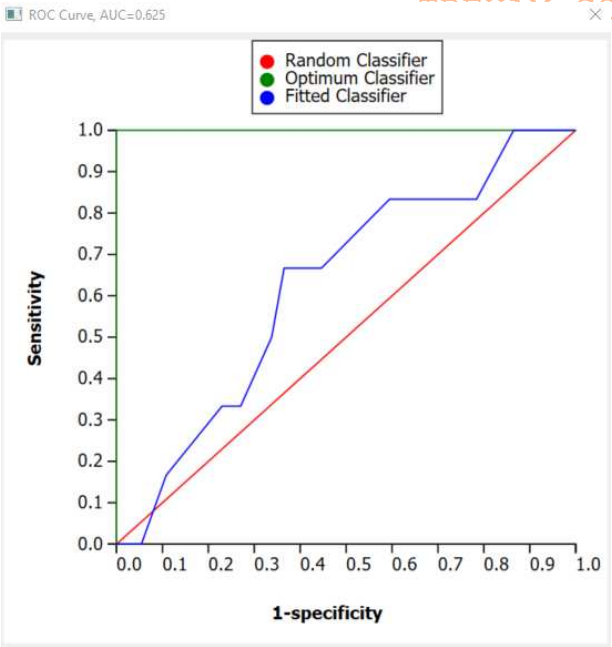
- Accuracy rate = 92.5%
- Specificity = 1
- Sensitivity = 0
- Precision = N/A

b.



The lift value of the leftmost bar is 1.6667. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 1.6667 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.625.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is only slightly effective in classifying the data. The lift curve of the naïve Bayes model lies mostly above the diagonal line. This suggests that the naïve Bayes classifier performs mostly better than the baseline model (random classifier) and is able to identify a larger percentage of target class (fraudulent activities) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class in most cases. The AUC value is 0.625, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.6667 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.925 with a sensitivity of 0 and a specificity of 1. This is due to the facts that using 0.5 as the cutoff value, all cases in the validation data set are classified as non-target class (not fraudulent) and that the target class cases only account for a very small percentage of the total cases (6.33%). Lowering the cutoff value will classify more cases as fraudulent activities but increase the likelihood of misclassifying non-fraudulent cases as fraudulent activities. This will result in a lower overall accuracy rate but allow us to identify more fraudulent activities accurately. In conclusion, the naïve Bayes model is only slightly effective in classifying the data.

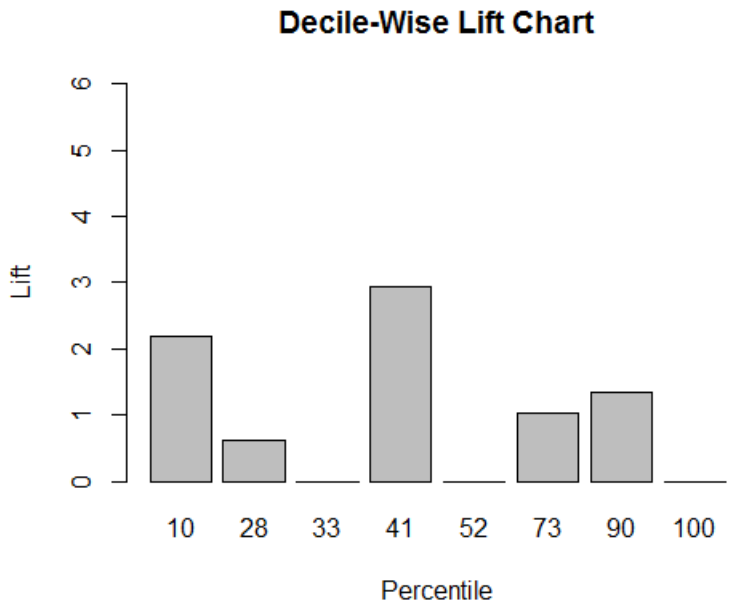
e. Cutoff Value = .10 (Using the validation data set)
- Accuracy rate = 43.75%
- Specificity = 0.4054
- Sensitivity = 0.8333
- Precision = 0.1020

12_39. <R>

a. Cutoff Value = .50 (Using the validation data set)

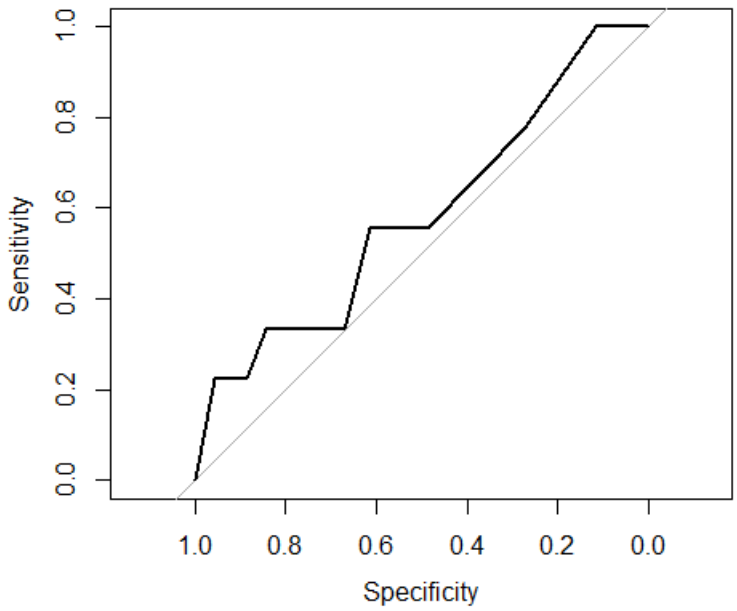
- Accuracy rate = 0.8861
- Specificity = 1
- Sensitivity = 0
- Precision = NaN

b.



The lift value of the leftmost bar is 2.19. This implies that by selecting the top 10% of the validation cases with the highest predicted probability of belonging to the target class, the naïve Bayes model would identify 2.19 times as many target class cases as if the cases are randomly selected.

c.



The AUC value is 0.5833.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is not very effective in classifying the data. The lift curve of the naïve Bayes model lies slightly above the diagonal line. This suggests that the naïve Bayes classifier performs slightly better than the baseline model (random classifier) and is able to identify a slightly larger percentage of target class (fraudulent activities) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.5833, which is much closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs only slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 2.19 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.8861 with a sensitivity of 0 and a specificity of 1. This is due to the facts that using 0.5 as the cutoff value, all cases in the validation data set are classified as non-target class (not fraudulent) and that the target class cases only account for a very small percentage of the total cases. Lowering the cutoff value will classify more cases as fraudulent activities but increase the likelihood of misclassifying non-fraudulent cases as fraudulent activities. This will result in a lower overall accuracy rate but allow us to identify more fraudulent activities accurately. In conclusion, the naïve Bayes model is not very effective in classifying the data.

e. Cutoff Value = .10 (Using the validation data set)
- Accuracy rate = 0.8734
- Specificity = 0.9571
- Sensitivity = 0.2222
- Precision = 0.4

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Fraud <- as.factor(myData$Fraud)
set.seed(1)
myIndex<- createDataPartition(myData$Fraud, p=0.6, list=FALSE)
```

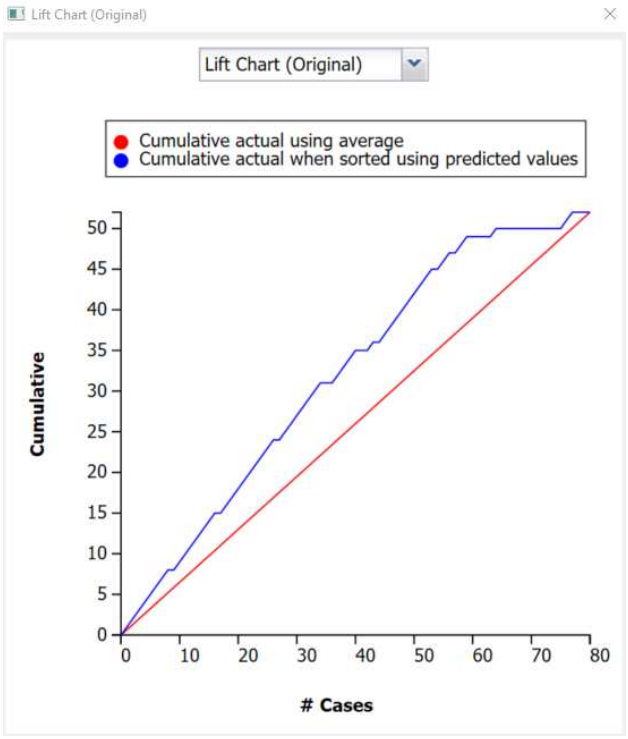
```
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Fraud ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Fraud, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Fraud <-
as.numeric(as.character(validationSet$Fraud))
gains_table <- gains(validationSet$Fraud, nb_Class_prob[,2])
gains_table
barplot(gains_table$mean.resp/mean(validationSet$Fraud),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,6), main="Decile-Wise Lift Chart")
#c
roc_object <- roc(validationSet$Fraud, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Fraud))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Fraud))~c(0, dim(validationSet)[1]),
col="red", lty=2)
#e
confusionMatrix(as.factor(ifelse(nb_Class_prob[,2]>0.1, '1',
'0')), as.factor(validationSet$Fraud), positive = '1')
```

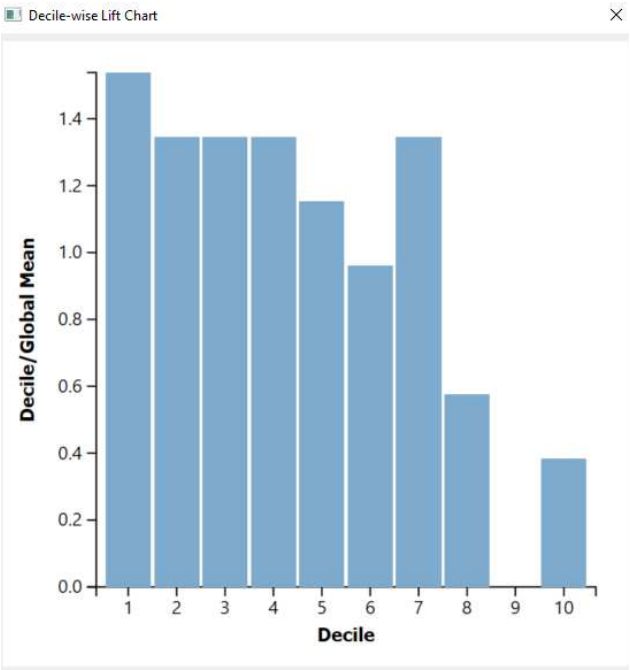

12_40. <Analytic Solver>

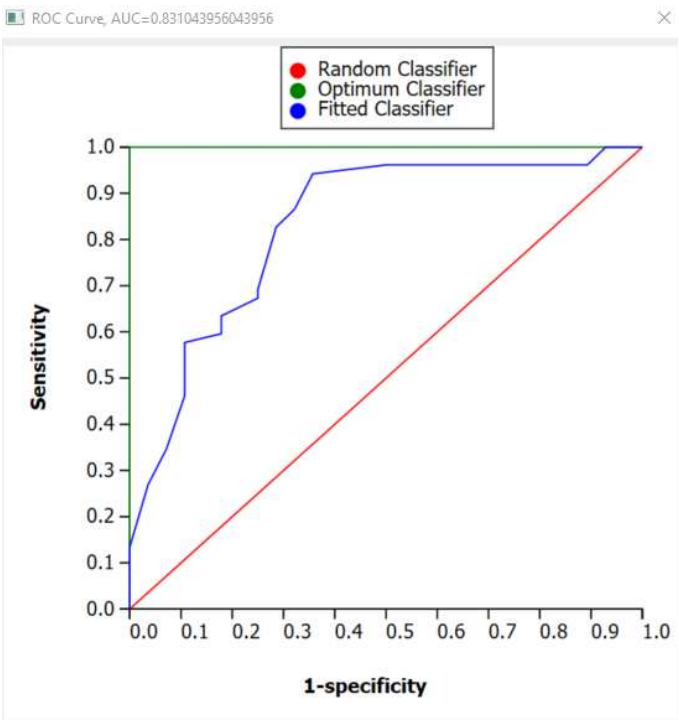
a. Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 71.25%
- Specificity = 0.75
- Sensitivity = 0.6923
- Precision = 0.8372

b.







c. AUC = 0.8310

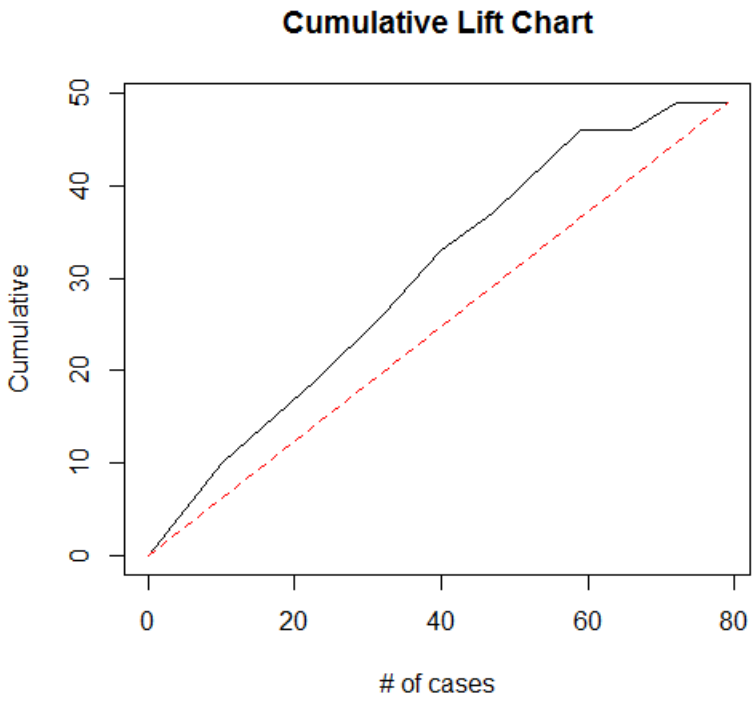
d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (Customers given discount) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8310, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.54 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 71.25% with a sensitivity of 0.6923 and a specificity of 0.75. Given that there are 52 target class cases among 80 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 52/80 = 0.65 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is effective in classifying the data.

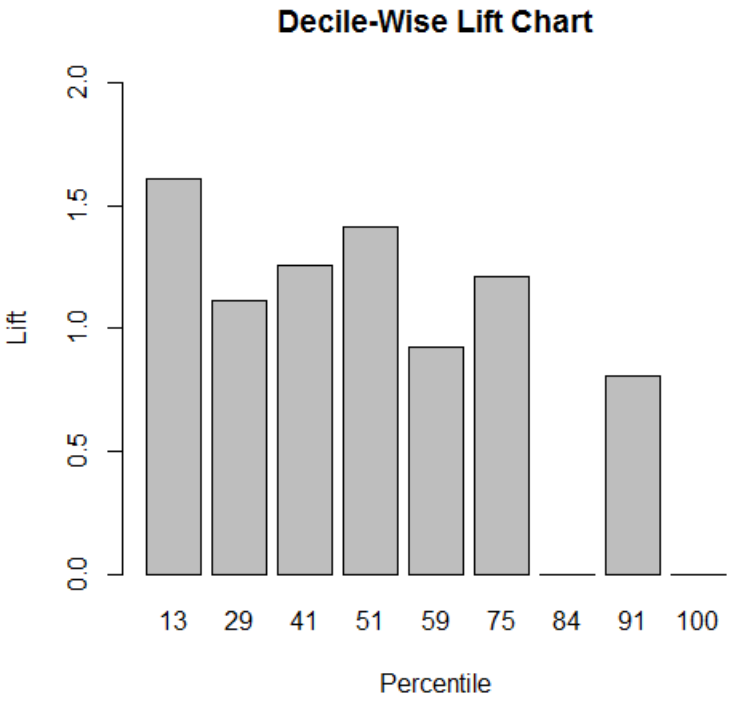
12_40. <R>

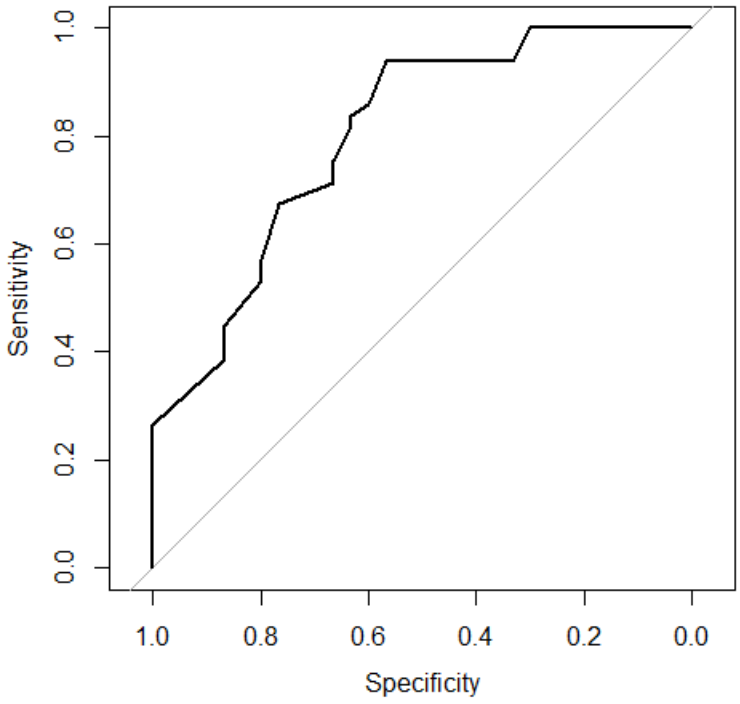
a. Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 0.7975
- Specificity = 0.5667
- Sensitivity = 0.9388
- Precision = 0.7797

b.







c. AUC = 0.802

d. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers given discount) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.802, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 13 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.61 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7975 with a sensitivity of 0.9388 and a specificity of 0.5667. This suggests that the model is better at classifying target class cases than classifying non-target class cases. However, increasing the cutoff value will classify more cases as non-target class cases thus increase specificity at the expense of sensitivity. Given that there are 49 target class cases among 79 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 0.6203 (49/79) but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Discount <- as.factor(myData$Discount)
set.seed(1)
myIndex<- createDataPartition(myData$Discount, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
```

```
nb_fit <- train(Discount ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Discount, positive = '1')
#b
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Discount <-
as.numeric(as.character(validationSet$Discount))
gains_table <- gains(validationSet$Discount, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Discount))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Discount))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Discount),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Discount, nb_Class_prob[,2])
plot.roc(roc_object)
#c
auc(roc_object)
#d No R code is needed for this question.
```


12_41. <Analytic Solver>

a.

	Income	Age
Observation 1	2	1
Observation 2	1	1

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 64.76%
   - Specificity = 0.8163
   - Sensitivity = 0.3958
   - Precision = 0.5907

c. AUC = 0.6711

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies slightly above the diagonal line. This suggests that the naïve Bayes classifier performs

slightly better than the baseline model (random classifier) and is able to identify a slightly larger percentage of target class (Customers who install solar) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6711, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs slightly better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.37 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6476 with a sensitivity of 0.3958 and a specificity of 0.8163. This suggests that the model is better at classifying non-target class cases than classifying target class cases. However, reducing the cutoff value will classify more cases as target class cases thus increase sensitivity. Given that there are 288 target class cases among 718 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (718-288)/718 = 0.5989 but sensitivity of 0. The naïve Bayes model performs slightly better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

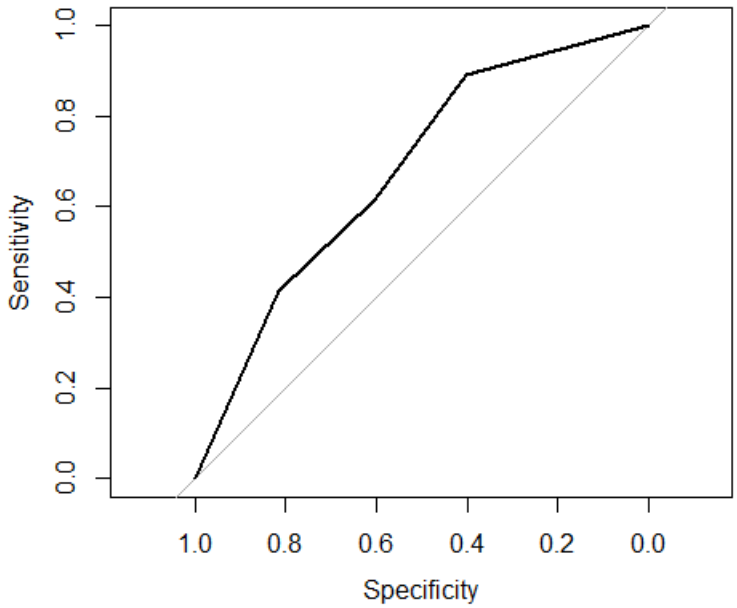
12_41. <R>

a.

	Income	Age
Observation 1	2	1
Observation 2	1	1

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.6569
- Specificity = 0.8157
- Sensitivity = 0.4134
- Precision = 0.5939

c.



The AUC value is 0.6781.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (customers who install solar) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.6781, which is closer to the baseline model (AUC = 0.5) than to the optimum model (AUC = 1), suggesting that the model performs a little better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 27 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.5 times as many target class cases as if the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.6569 with a sensitivity of 0.4134 and a specificity of 0.8157. This suggests that the model is better at classifying non-target class cases than classifying target class cases. However, reducing the cutoff value will classify more cases as target class cases thus increase sensitivity. Given that there are 283 target class cases among 717 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of (717-283)/717 = 0.6053 but sensitivity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Install <- as.factor(myData$Install)
myData$Age <- cut(myData$Age, breaks = c(30, 50, 90),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
myData$Income <- cut(myData$Income, breaks = c(30, 85,
140),labels = c("1", "2"),  right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$Install, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Install ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Install, positive = 'Y')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Install <- ifelse(validationSet$Install == "Y", 1,
0)
roc_object <- roc(validationSet$Install, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
gains_table <- gains(validationSet$Install, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Install))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Install))~c(0,
dim(validationSet)[1]), col="red", lty=2)
```

```
barplot(gains_table$mean.resp/mean(validationSet$Install),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
```

12_42. <Analytic Solver>

a.

	GPA
Observation 1	1
Observation 2	1

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 72.92%
   - Specificity = 0.6154
   - Sensitivity = 0.8636
   - Precision = 0.6552

c. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 72.92%
   - Specificity = 0.6154
   - Sensitivity = 0.8636
   - Precision = 0.6552

d. The probability is 0.1421.

12_42. <R>

a.

	GPA
Observation 1	1
Observation 2	1

b. Cutoff Value = .50 (Using the validation data set)
   - Accuracy rate = 0.766
   - Specificity = 0.7241
   - Sensitivity = 0.8333
   - Precision = 0.6522

c. Cutoff Value = .30 (Using the validation data set)
   - Accuracy rate = 0.766
   - Specificity = 0.7241
   - Sensitivity = 0.8333
   - Precision = 0.6522

d. The probability is 0.0311.

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData <- myData[, 1:4]
myData$Call <- as.factor(myData$Call)
myData$GPA <- cut(myData$GPA, breaks = c(0, 3.5, Inf),labels =
c("1", "2"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$Call, p=0.6, list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Call ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Call, positive = 'Yes')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
confusionMatrix(as.factor(ifelse(nb_Class_prob[,2]>0.3, 'Yes',
'No')), as.factor(validationSet$Call), positive = 'Yes')
#d
myScoreData <- data.frame(GPA = 1, Male = 1, Looks = 1)
myScoreData$GPA <- as.factor(myScoreData$GPA)
nb_Score <- predict(nb_fit, newdata=myScoreData, type = "prob")
nb_Score
```

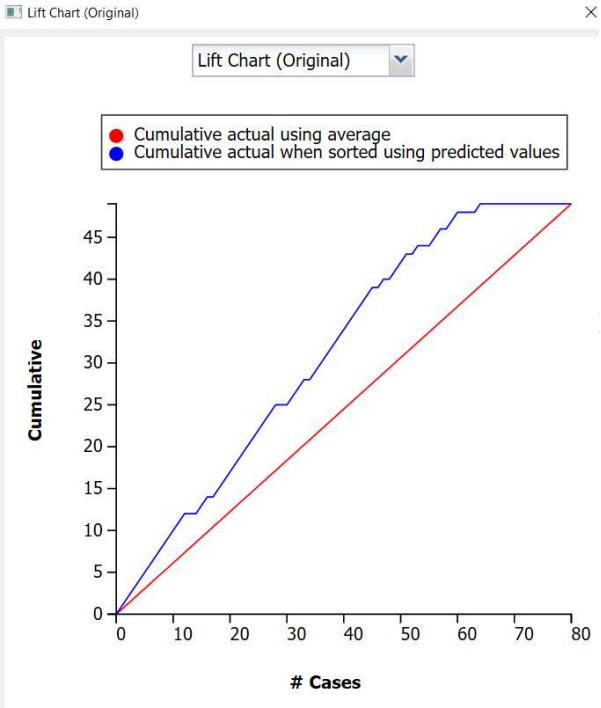

12_43. <Analytic Solver>

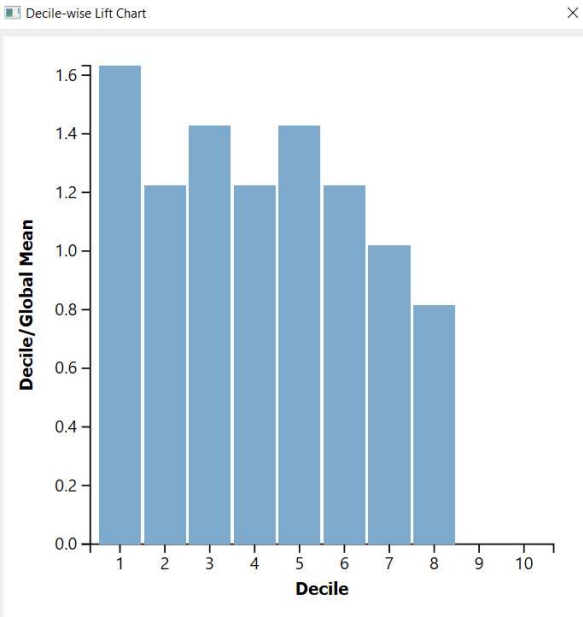
a.

	Age	Income
Observation 1	2	4
Observation 2	3	3

e. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 78.75%
- Specificity = 0.7419
- Sensitivity = 0.8163
- Precision = 0.8333

c. See the charts below. The entire lift curve lies above the baseline. The lift value of the leftmost bar is 1.63.





d. The AUC value is 0.8535.

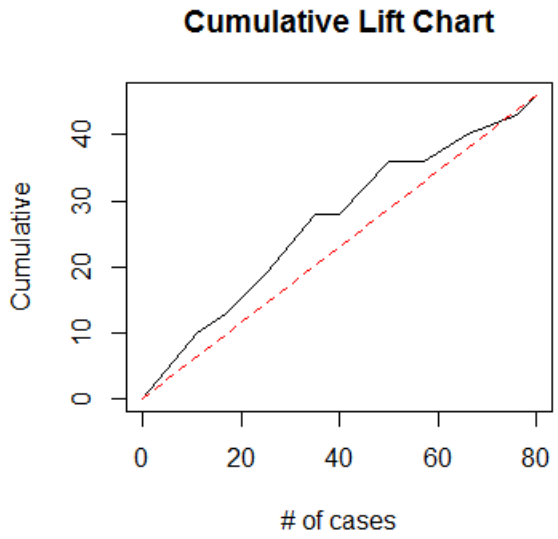
12_43. <R>

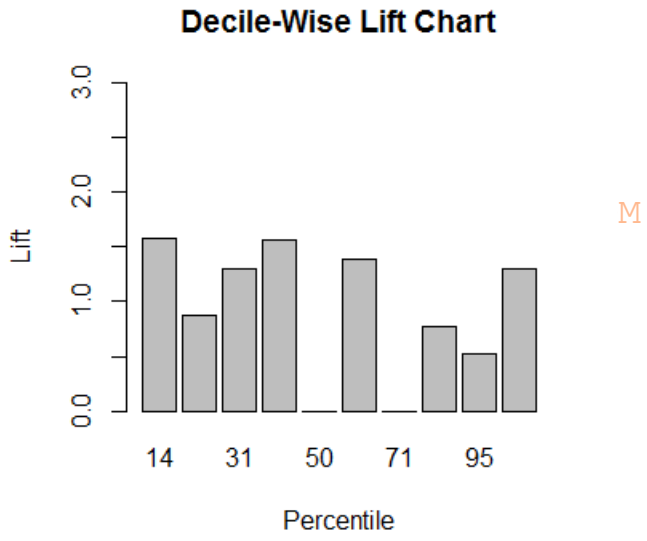
a.

	Age	Income
Observation 1	2	4
Observation 2	3	4

b. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 0.7
- Specificity = 0.5882
- Sensitivity = 0.7826
- Precision = 0.72

c. See the charts below. The lift curve does not lie above the baseline entirely. The lift value of the leftmost bar is 1.58.





d. AUC = 0.6845

R Code:

```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
```

```
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Performance <- as.factor(myData$Performance)
myData$Age <- cut(myData$Age, breaks = 4,labels = c("1", "2",
"3", "4"), right = FALSE, include.lowest = TRUE)
myData$Income <- cut(myData$Income, breaks = 4, labels = c("1",
"2", "3", "4"), right = FALSE, include.lowest = TRUE)
head(myData)
#b
set.seed(1)
myIndex<- createDataPartition(myData$Performance, p=0.6,
list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Performance ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Performance, positive =
'1')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Performance <-
as.numeric(as.character(validationSet$Performance))
gains_table <- gains(validationSet$Performance,
nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Performance))~c(0
, gains_table$cume.obs), xlab = "# of cases", ylab =
"Cumulative", main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Performance))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Performance),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
#d
roc_object <- roc(validationSet$Performance, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```

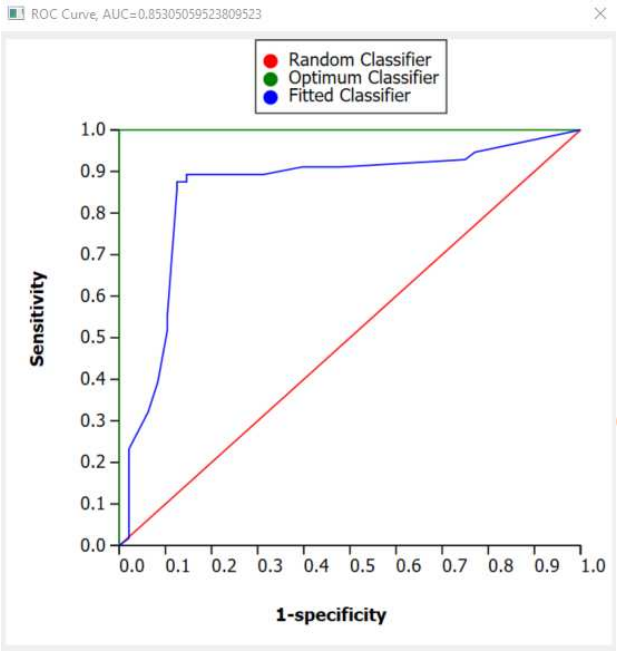

12_44. <Analytic Solver>

a.

	Age	Education	Hours
Observation 1	2	2	1
Observation 2	2	1	1

f. Cutoff Value = .50 (Using the validation data set)
- Accuracy rate = 87.5%
- Specificity = 0.854
- Sensitivity = 0.8929
- Precision = 0.8772

e.



The AUC value is 0.8531.

f. The performance measurements and graphs suggest that the naïve Bayes classifier is effective in classifying the data. The lift curve of the naïve Bayes model lies above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (individuals who suffer from depression) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.8531, which is closer to the optimum model (AUC = 1) than to the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify 1.67 times as many target class cases as if

the cases are randomly selected. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.875 with a sensitivity of 0.8929 and a specificity of 0.8542. Given that there are 56 target class cases among 104 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 56/104 = 0.5385 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is effective in classifying the data.

12_44. <R>

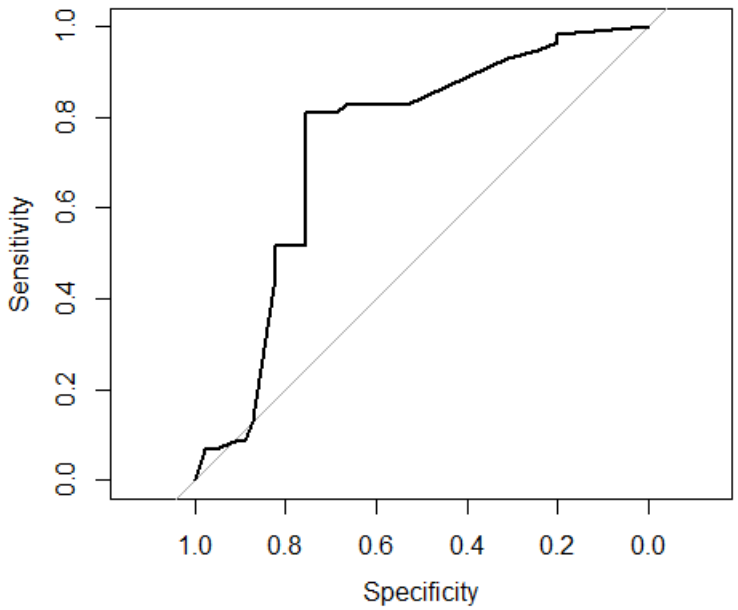
a.

	Age	Education	Hours
Observation 1	2	2	1
Observation 2	2	1	1

e. Cutoff Value = .50 (Using the validation data set)

- Accuracy rate = 0.7864
- Specificity = 0.7556
- Sensitivity = 0.8103
- Precision = 0.8103

c.



The AUC value is 0.7395.

d. The performance measurements and graphs suggest that the naïve Bayes classifier is moderately effective in classifying the data. The lift curve of the naïve Bayes model lies mostly above the diagonal line. This suggests that the naïve Bayes classifier performs better than the baseline model (random classifier) and is able to identify a larger percentage of target class (individuals who suffer from depression) cases by looking at a small percentage of the validation cases with the highest predicted probabilities of belonging to the target class. The AUC value is 0.7395, which is about half way between the optimum model (AUC = 1) and the baseline model (AUC = 0.5), suggesting that the model performs better than the baseline model in terms of sensitivity and specificity across all possible cutoff values. The leftmost bar in the decile-wise lift chart suggests that by selecting the top 10 percent of the validation cases with the highest predicted probabilities of belonging to the target class, the naïve Bayes classifier would identify only 0.89 times as many target class cases as if the cases are randomly selected. However, the lift value improves for the next few deciles. For example, the top 56% of the cases with the highest probabilities of belonging to the target class contain 81% of true target class cases. Using 0.5 as the cutoff value, the model's overall accuracy rate is 0.7864 with a sensitivity of 0.8103 and a specificity of 0.7556. Given that there are 58 target class cases among 103 validation cases, the naïve rule (classifying all cases to the predominant class) would generate an accuracy of 58/103 = 0.5631 but specificity of 0. The naïve Bayes model performs better than the naïve rule in terms of overall accuracy. In conclusion, the naïve Bayes model is moderately effective in classifying the data.

R Code:


```
# Import the data file into a data frame (table) and label it
myData.
##Install the following r packages if you have not already done
so.
install.packages("caret")
install.packages("klaR")
install.packages("gains")
install.packages("pROC")
#a
library(caret)
library(klaR)
library(gains)
library(pROC)
myData$Depression <- as.factor(myData$Depression)
myData$Age <- cut(myData$Age, breaks = c(20, 40, 60, 81),labels =
c("1", "2", "3"), right = FALSE, include.lowest = TRUE)
myData$Education <- cut(myData$Education, breaks = c(9, 12, 16,
20),labels = c("1", "2", "3"),  right = FALSE, include.lowest =
TRUE)
myData$Hours <- cut(myData$Hours, breaks = c(0, 55, 105,
150),labels = c("1", "2", "3"),  right = FALSE, include.lowest =
TRUE)
head(myData)
#b
```

```
set.seed(1)
myIndex<- createDataPartition(myData$Depression, p=0.6,
list=FALSE)
trainSet <- myData[myIndex,]
validationSet <- myData[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Depression ~., data = trainSet, method = "nb",
trControl=myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata=validationSet)
confusionMatrix(nb_class, validationSet$Depression, positive =
'Y')
#c
nb_Class_prob <- predict(nb_fit, newdata = validationSet,
type='prob')
validationSet$Depression <- ifelse(validationSet$Depression ==
"Y", 1, 0)
gains_table <- gains(validationSet$Depression, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Depression))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Depression))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Depression),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Depression, nb_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
#d
gains_table <- gains(validationSet$Depression, nb_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Depression))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Depression))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Depression),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,2), main="Decile-Wise Lift Chart")
```


Possible Output for Suggested Case Studies

There are many ways to approach these case studies. We provide numerical guidance for one possible scenario. Instructors may also be interested in Connect multiple-choice exercises (algorithmic) that we have created using the **TechSales_Rep** data. These exercises span multiple chapters, including this one.

Report 12.1

The following example uses sex, parents' education (the average of mother's and father's years of education), height, weight, number of years of education, self-esteem scale, and whether the person is outgoing as a kid and as an adult to predict whether the person would experience a divorce. The data mining technique used is k-nearest neighbors (KNN). First subset the **Longitudinal_Survey** data to include only those individuals who lived in an urban area and only the variables that will be used in the analysis. Use the omission strategy to remove observations with missing values. This results in 4848 complete cases. Create a new variable Divorce whose value is 1 if the person is divorced and 0 otherwise. Standardize all the predictor variables. Partition the data into 60% training and 40% validation.

KNN analysis suggests that the best value for k among the numbers 1 to 10 is 9. Using the cutoff value of 0.157, which is the proportion of divorces among all the observations, the accuracy of the KNN model on the validation data set is 0.5573 with a sensitivity of 0.4656 and specificity of 0.5744. Both the cumulative lift curve and ROC curve lie slightly above the diagonal line (baseline model), and the AUC value is 0.5336. This suggests that the KNN model performs about the same as the base model (random selection) in identifying people who would experience a divorce in the data set.


```
#Report 12.1
suppressWarnings(RNGversion("3.5.3"))
myData <- myData[myData$Urban == 1,]
View(myData)
myData <- myData[, c(4, 5, 12, 14, 15, 16, 17, 18, 21, 22)]
myData <- na.omit(myData)
myData$Parent_Edu <- (myData$Mother_Edu + myData$Father_Edu)/2
dim(myData)
library(caret)
library(gains)
library(pROC)
myData$Divorce <- ifelse(myData$Marital_Status == 3, 1, 0)
myData1<- scale(myData[, c(3:8, 10, 11)])
myData1<- data.frame(myData1, myData$Divorce)
colnames(myData1)[9] <- 'Marital_Status'
# myData1$Marital_Status<- ifelse(myData1$Marital_Status == 3, 1, 0)
myData1$Marital_Status<- as.factor(myData1$Marital_Status)
set.seed(1)
myIndex<- createDataPartition(myData1$Marital_Status, p=0.6,
list=FALSE)
trainSet <- myData1[myIndex,]
```

```
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Marital_Status ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Marital_Status, positive =
'1')
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet,
type='prob')
KNN_Class_prob
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.157, '1', '0')),
validationSet$Marital_Status, positive = '1')
validationSet$Marital_Status <-
as.numeric(as.character(validationSet$Marital_Status))
gains_table <- gains(validationSet$Marital_Status, KNN_Class_prob[,2])
gains_table
plot(c(0,
gains_table$cume.pct.of.total*sum(validationSet$Marital_Status))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Marital_Status))~c(0,
dim(validationSet)[1]), col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Marital_Status),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,3), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Marital_Status, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


Report 12.2

The following example uses age, sex, tenure with the company, number of professional certificates, annual evaluation score, and personality type to predict whether the sales professional would receive a high NPS. The data mining technique used is k-nearest neighbors (KNN). First subset the **TechSales_Reps** data to include only college-educated sales professionals in the software product group. You should have 9862 observations in the subset. Create a new variable NSP_Category with the value of 1 if the sales representative's NSP is 9 or more and 0 if otherwise. Transform Personality to dummy variables using Analyst as the reference category. Standardize all predictor variables. Partition the data into 60% training and 40% validation.

KNN analysis suggests that the best value for k among the numbers 1 to 10 is 10. Using the cutoff value of 0.219, which is the proportion of high NSP sales professionals among all the observations, the accuracy of the KNN model on the validation data set is 0.7409 with a sensitivity of 0.7083 and specificity of 0.7500. Both the cumulative lift curve and ROC curve lie

well above the diagonal line (baseline model), and the AUC value is 0.8017. This suggests that the KNN model performs well in identifying sales professionals with high NPS in the data set.

```
#Report 12.2
#Import the data from the TechSales_Reps data file into a data frame (table)
and label it myData.
```

```
myData1 <- myData[myData$Business =='Software' &  myData$College == 'Yes',]
dim(myData1)
suppressWarnings(RNGversion("3.5.3"))
myData1$NPS_Category <- ifelse(myData1$NPS>=9, 1, 0)
myData1$Diplomat <- ifelse(myData1$Personality == 'Diplomat', 1, 0)
myData1$Explorer <- ifelse(myData1$Personality == 'Explorer', 1, 0)
myData1$Sentinel <- ifelse(myData1$Personality == 'Sentinel', 1, 0)
View(myData1)
library(caret)
library(gains)
library(pROC)
myData1 <- myData1[, c(3,4,5,8,9,13, 14, 15,12)]
myData1_S<- scale(myData1[1:8])
myData1_S<- data.frame(myData1_S, myData1$NPS_Category)
colnames(myData1_S)[9] <- 'NPS_Category'
myData1_S$NPS_Category<- as.factor(myData1_S$NPS_Category)
set.seed(1)
myIndex<- createDataPartition(myData1_S$NPS_Category, p=0.6, list=FALSE)
trainSet <- myData1_S[myIndex,]
validationSet <- myData1_S[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(NPS_Category ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$NPS_Category, positive = '1')
prop.table(table(validationSet$NPS_Category))
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet, type = 'prob')
head(KNN_Class_prob)
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.219, '1', '0')),
validationSet$NPS_Category, positive = '1')
validationSet$NPS_Category <-
as.numeric(as.character(validationSet$NPS_Category))
gains_table <- gains(validationSet$NPS_Category, KNN_Class_prob[,2])
gains_table
plot(c(0, gains_table$cume.pct.of.total*sum(validationSet$NPS_Category))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$NPS_Category))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$NPS_Category),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift", ylim=c(0,3),
main="Decile-Wise Lift Chart")
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet, type='prob')
roc_object <- roc(validationSet$NPS_Category, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


Report 12.3

The following example uses gender, race, high school GPA, SAT/ATC score, and parent's education level to predict whether an admitted student of the College of Business & Economics

will enroll. The data mining technique used is k-nearest neighbors (KNN), First subset the **College_Admissions** data to include only the College of Business & Economics and admitted students. You should have 1470 observations. Create a new variable Edu_Parent. The values of the new variable are the averages of the values of Edu_Parent1 and Edu_Parent2. Transform Gender into a numeric variable with 1 being female and 0 being male. Standardize all the predictor variables. Partition the data into 60% training and 40% validation.

KNN analysis suggests that the best value for k among the numbers 1 to 10 is 8. Using the cutoff value of 0.306, which is the proportion of enrolled students among all the observations, the accuracy of the KNN model on the validation data set is 0.6173 with a sensitivity of 0.5889 and specificity of 0.6299. Both the cumulative lift curve and ROC curve lie above the diagonal line (baseline model), and the AUC value is 0.6364. This suggests that the KNN model performs slightly better than the base model in identifying students who will enroll in the data set.

```
#12.3
myData1 <- myData[myData$College == 'Business & Economics' &  myData$Admitted
== 'Yes',]
dim(myData1)
suppressWarnings(RNGversion("3.5.3"))
myData1$Gender <- ifelse(myData1$Gender == 'F', 1, 0)
myData1$Edu_Parent <- (myData1$Edu_Parent1 +myData1$Edu_Parent2)/2
View(myData1)
library(caret)
library(gains)
library(pROC)
myData1 <- myData1[, c(4,5,6, 7,8,13, 11)]
myData1_S<- scale(myData1[1:6])
myData1_S<- data.frame(myData1_S, myData1$Enrolled)
colnames(myData1_S)[7] <- 'Enrolled'
myData1_S$Enrolled<- as.factor(myData1_S$Enrolled)
set.seed(1)
myIndex<- createDataPartition(myData1_S$Enrolled, p=0.6, list=FALSE)
trainSet <- myData1_S[myIndex,]
validationSet <- myData1_S[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
myGrid <- expand.grid(.k=c(1:10))
set.seed(1)
KNN_fit <- train(Enrolled ~ ., data=trainSet, method = "knn",
trControl=myCtrl, tuneGrid = myGrid)
KNN_fit
KNN_Class <- predict(KNN_fit, newdata = validationSet)
confusionMatrix(KNN_Class, validationSet$Enrolled, positive = 'Yes')
prop.table(table(validationSet$Enrolled))
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet, type = 'prob')
head(KNN_Class_prob)
confusionMatrix(as.factor(ifelse(KNN_Class_prob[,2]>0.306, 'Yes', 'No')),
validationSet$Enrolled, positive = 'Yes')
validationSet$Enrolled <- ifelse(validationSet$Enrolled == 'Yes', 1, 0)
validationSet$NPS_Category <-
as.numeric(as.character(validationSet$Enrolled))
gains_table <- gains(validationSet$Enrolled, KNN_Class_prob[,2])
gains_table
plot(c(0, gains_table$cume.pct.of.total*sum(validationSet$Enrolled))~c(0,
gains_table$cume.obs), xlab = "# of cases", ylab = "Cumulative",
main="Cumulative Lift Chart", type="l")
lines(c(0, sum(validationSet$Enrolled))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Enrolled),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift", ylim=c(0,3),
main="Decile-Wise Lift Chart")
KNN_Class_prob <- predict(KNN_fit, newdata = validationSet, type='prob')
roc_object <- roc(validationSet$Enrolled, KNN_Class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```


Report 12.4

The following example uses weather condition, amount of daylight, and whether or not the accident takes place on a highway to predict whether the car crash in San Diego will result in fatality or severe injuries. The data mining technique used is naïve Bayes (NB) because all the predictors are categorical. First subset the **Car_Crash** data to include only the accidents that occurred in San Diego You should have 2380 observations. Partition the data into 60% training and 40% validation.

NB analysis suggests that using the cutoff value of 0.07, which is the proportion of accidents with fatality or severe injuries among all the observations, the accuracy of the NB model on the validation data set is 0.489 with a sensitivity of 0.7761 and specificity of 0.4671. Both the cumulative lift curve and ROC curve lie above the diagonal line (baseline model), and the AUC value is 0.6544. Although the overall accuracy rate of the NB model is not very high, the model is able to correctly classify a large portion (77.61%) of the accidents with fatality or severe injuries, which is the usually the goal of the analysis. This suggests that the NB model performs reasonably well in identifying accidents with fatality or severe injuries in the data set.

```
#12.4
myData1 <- myData[myData$City == 'SAN DIEGO',]
dim(myData1)
suppressWarnings(RNGversion("3.5.3"))
library(caret)
library(klaR)
library(gains)
library(pROC)
myData1$Severity <- as.factor(myData1$Severity)
myData1 <- myData1[, c(7, 10, 11, 5)]
set.seed(1)
myIndex<- createDataPartition(myData1$Severity, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
set.seed(1)
nb_fit <- train(Severity ~., data = trainSet, method = "nb", trControl
= myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata = validationSet)
confusionMatrix(nb_class, validationSet$Severity, positive = '1')
# If use a cutoff value other than the default 0.5 to create the
confusion matrix, use the following two commands
nb_class_prob <- predict(nb_fit, newdata=validationSet, type= 'prob')
confusionMatrix(as.factor(ifelse(nb_class_prob[,2]>0.07, '1', '0')),
validationSet$Severity, positive = '1')
# Create performance charts
nb_class_prob <- predict(nb_fit, newdata = validationSet, type =
'prob')
validationSet$Severity <-
as.numeric(as.character(validationSet$Severity))
```

```
gains_table <- gains(validationSet$Severity, nb_class_prob[,2])
gains_table
plot(c(0, gains_table$cume.pct.of.total*sum(validationSet$Severity)) ~
c(0, gains_table$cume.obs), xlab = '# of cases', ylab = "Cumulative",
type = "l")
lines(c(0, sum(validationSet$Severity))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Severity),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,1.5), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Severity, nb_class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```

Report 12.5

The following example uses symptoms, age, sex, and whether the individual had contact with confirmed COVID cases to predict whether the testing result is positive. The data mining technique used is naïve Bayes (NB) because most predictors are categorical. First, remove cases in the **COVID_Testing** data set with missing values. You should have 978493 complete cases. Randomly sample 5000 cases from the complete cases. Partition the 5000 cases into 60% training and 40% validation.

NB analysis suggests that using the cutoff value of 0.105, which is the proportion of positive cases among all the observations, the accuracy of the NB model on the validation data set is 0.914 with a sensitivity of 0.5095 and specificity of 0.9615. Both the cumulative lift curve and ROC curve lie above the diagonal line (baseline model), and the AUC value is 0.7356. This suggests that the NB model performs moderately well in classifying the positive and negative cases in the data set. The model is especially good at identifying negative cases.

```
#12.5
suppressWarnings(RNGversion("3.5.3"))
myData[!complete.cases(myData), ]
myData1 <- na.omit(myData)
dim(myData1)
set.seed(1)
myData1 <- myData1[sample(nrow(myData1), 5000), ]
dim(myData1)
library(caret)
library(klaR)
library(gains)
library(pROC)
myData1$Result <- as.factor(myData1$Result)
set.seed(1)
myIndex<- createDataPartition(myData1$Result, p=0.6, list=FALSE)
trainSet <- myData1[myIndex,]
validationSet <- myData1[-myIndex,]
myCtrl <- trainControl(method="cv", number=10)
```

```
set.seed(1)
nb_fit <- train(Result ~., data = trainSet, method = "nb", trControl =
myCtrl)
nb_fit
nb_class <- predict(nb_fit, newdata = validationSet)
confusionMatrix(nb_class, validationSet$Result, positive = 'positive')
# If use a cutoff value other than the default 0.5 to create the
confusion matrix, use the following two commands
nb_class_prob <- predict(nb_fit, newdata=validationSet, type= 'prob')
confusionMatrix(as.factor(ifelse(nb_class_prob[,2]>0.105, 'positive',
'negative')), validationSet$Result, positive = 'positive')
# Create performance charts
nb_class_prob <- predict(nb_fit, newdata = validationSet, type =
'prob')
validationSet$Result <- ifelse(validationSet$Result == 'positive', 1,
0)
validationSet$Result <- as.numeric(as.character(validationSet$Result))
gains_table <- gains(validationSet$Result, nb_class_prob[,2])
gains_table
plot(c(0, gains_table$cume.pct.of.total*sum(validationSet$Result)) ~
c(0, gains_table$cume.obs), xlab = '# of cases', ylab = "Cumulative",
type = "l")
lines(c(0, sum(validationSet$Result))~c(0, dim(validationSet)[1]),
col="red", lty=2)
barplot(gains_table$mean.resp/mean(validationSet$Result),
names.arg=gains_table$depth, xlab="Percentile", ylab="Lift",
ylim=c(0,1.5), main="Decile-Wise Lift Chart")
roc_object <- roc(validationSet$Result, nb_class_prob[,2])
plot.roc(roc_object)
auc(roc_object)
```

Chapter 2. Data Management and Wrangling

Solutions

1. a, c, and d. Choice b is the definition of data modeling.
2. a, c, and d. Choice b describes a primary key, not a foreign key.
3. a, b, and d. Choice c: a relational database does not necessarily include an advanced visualization tool.
4. c
5. a. (M:N)
   b. (1:M)
   c. (1:M)
   d. (1:M)
   e. (M:N)
   f. (1:M)
6. b, c, and d. Choice a describes NoSQL.
7. b, c, and d. Choice a describes a data mart, not a data warehouse.
8. a. 50 observations of x_2 are equal to two.
   b. When sorted in ascending order the first observation is $x_1 = 0$ and $x_2 = 1$.
   c. When sorted in descending order the first observation is $x_1 = 1$ and $x_2 = 2$.
   d. When x_1 is sorted in ascending order and x_2 is sorted in descending order, the first observation is $x_1 = 0$ and $x_2 = 1$.
   e. 0 values are missing for x_1 and x_2.

   R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[ , -1]
length(which(myData$x2==2))
```

```
#b
sortedData1 <- myData[order(myData$x1, myData$x2), ]
#c
sortedData2 <- myData[order(myData$x1, myData$x2, decreasing = TRUE),]
#d
sortedData3 <- myData[order(myData$x1, -myData$x2), ]
#e
myData[!complete.cases(myData), ]
```

9.

a. 17 observations of x_1 are greater than 30.
b. When sorted in ascending order the first observation is $x_1 = 2$, $x_2 = 540$, and $x_3 = 201$.
c. When x_1 and x_2 are sorted in descending order, and x_3 is sorted in ascending order, the first observation is $x_1 = 52$, $x_2 = 218$, and $x_3 = 154$.
d. One value is missing for x_1, three values are missing for x_2, and five values are missing for x_3.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[ , -1]
length(which(myData$x1>30))
sortedData1 <- myData[order(myData$x1, myData$x2, myData$x3), ]
View(sortedData1)
#b
sortedData2 <- myData[order(myData$x1, myData$x2, decreasing = TRUE,
myData$x3), ]
View(sortedData2)
#c
sortedData3 <- myData[order(myData$x1, -myData$x2), ]
View(sortedData3)
myData[!complete.cases(myData), ]
```


10.

a. 85 observations of x_4 are less than 3.
b. When sorted in ascending order, the first observation is $x_1 = 0.0121$, $x_2 = 24$, $x_3 = 27$, and $x_4 = 3$.
c. When sorted in descending order, the first observation is $x_1 = 0.9980$, $x_2 = 196$, $x_3 = 13$, and $x_4 = 3$.
d. 0 values are missing for x_1, x_2, x_3, and x_4.
e. The categorical variable x_4 has three categories: 1, 2 and 3. Category 1 has 40 observations, Category 2 has 45 observations, and Category 3 has 33 observations.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[ , -1]
length(which(myData$x4<3))
#b
sortedData1 <- myData[order(myData$x1, myData$x2, myData$x3,
myData$x4), ]
View(sortedData1)
#c
sortedData2 <- myData[order(myData$x1, myData$x2, myData$x3, myData$x4,
decreasing = TRUE), ]
View(sortedData2)
#d
sortedData3 <- myData[order(myData$x1, -myData$x2), ]
View(sortedData3)
#e
myData[!complete.cases(myData), ]
length(which(myData$x4==1))
length(which(myData$x4==2))
length(which(myData$x4==3))
```

11.

a. Minnesota has the highest average writing score of 643. The average math score of Minnesota is 655.
b. The Virgin Islands has the lowest average math score of 445. The average writing score of the Virgin Islands is 490.
c. 13 states reported average math scores above 600.
d. 25 states reported average writing scores below 550.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
sortedData1 <- myData[order(myData$Writing, decreasing = TRUE),]
#b
sortedData2 <- myData[order(myData$Math),]
#c
length(which(myData$Math > 600))
#d
length(which(myData$Writing < 550))
```

12.

a. One of the 10 highest income earners is married and always exercises.

b. Nine individuals are married, exercise sometimes, and earn more than $110,000 per year.
c. Five values are missing for Exercise, two for Marriage, and three for Income.
d. 281 individuals are married, and 134 are not.
e. 69 married individuals always exercise, and 74 unmarried individuals never exercise.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
sortedData1 <- myData[order(myData$Income, decreasing = TRUE),]
sortedData1 <- sortedData1[1:10, ]
length(which(sortedData1$Married=="Yes" &
sortedData1$Exercise=="Always"))
#b
sortedData2 <- myData[order(myData$Married, myData$Exercise, decreasing =
TRUE),]
View(sortedData2)
#c
length(which(is.na(myData$Exercise)))
length(which(is.na(myData$Married)))
length(which(is.na(myData$Income)))
#d
length(which(myData$Married =="Yes"))
length(which(myData$Married == "No"))
#e
length(which(myData$Married == "Yes" & myData$Exercise == "Always"))
length(which(myData$Married == "No" & myData$Exercise == "Never"))
```


13.

a. The first customer in this list spent $3,362.86 on food.
b. Of the 10 customers who spent the most of travel, three were home owners, and one was both a home and car owner.
c. Three values are missing for the variable OwnHome, three values are missing for the variable OwnCar, 0 values are missing for the variable FoodSpend, and one value is missing for the variable TravelSpend.
d. 133 of the customers own homes.
e. 88 of the customers own homes but do not own cars.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
sortedData1 <- myData[order(myData$OwnHome, myData$OwnCar,
myData$TravelSpend, decreasing = TRUE),]
```

```
sortedData1[1,]$FoodSpend
#b
sortedData2 <- myData[order(myData$TravelSpend, decreasing = TRUE),]
sortedData2 <- sortedData2[1:10,]
length(which(sortedData2$OwnHome=="Yes"))
length(which(sortedData2$OwnHome=="Yes" & sortedData2$OwnCar=="Yes"))
#c
length(which(is.na(myData$OwnHome)))
length(which(is.na(myData$OwnCar)))
length(which(is.na(myData$FoodSpend)))
length(which(is.na(myData$TravelSpend)))
#d
length(which(myData$OwnHome =="Yes"))
#e
length(which(myData$OwnHome == "Yes" & myData$OwnCar == "No"))
```

14.

a. In the data there are 508 males and 382 females.
b. $\frac{336}{508} \times 100\% = 66.1417\%$ of males in the data are married, and $\frac{248}{382} \times 100\% = 64.9215\%$ of the females in the data are married.
c. Of the 10 individuals with the highest income 7 are married males.
d. The largest income among females is $138,000, the smallest is $22,000. The largest income among males is $147,000, the smallest is $35,000.
e. The largest income among married males is $147,000, the smallest is $35,000. The largest income among unmarried males is $140,000, the smallest is $39,000.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a. Finding the number of males and females in the data
length(which(myData$Sex=='M'))
length(which(myData$Sex=='F'))
#b. Finding the percentage of males and females who are married
length(which(myData$Sex=='M' &
myData$Married=='Y'))/length(which(myData$Sex=='M'))*100
length(which(myData$Sex=='F' &
myData$Married=='Y'))/length(which(myData$Sex=='F'))*100
#c. Of the 10 individuals with the highest income, finding the number who are
married males
sortedData <- myData[order(myData$Income, decreasing = TRUE),]
View(sortedData)
length(which(sortedData[1:10,]$Sex=='M' & sortedData[1:10,]$Married=='Y'))
#d. Find the highest and the lowest incomes of males and females by inspecting
sortedData
#e. Find the highest and lowest incomes of married and unmarried males by
inspecting sortedData
```

15.

a. In the data there are 616 males and 614 females.

b. $\frac{304}{616} \times 100\% = 49.3506\%$ of males in the data, and $\frac{352}{614} \times 100\% = 57.3290\%$ of the females in the data are admitted.

c. Two of the 10 students with the highest high school GPAs are male.

d. Six of the 10 students with the lowest SAT score are female.

e. The highest SAT score of admitted males is 1,600, the lowest is 1,055. The highest SAT score of admitted females is 1,599, the lowest is 1,062.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a.
length(which(myData$Female=='Yes'))
length(which(myData$Female=='No'))
#b.
length(which(myData$Decision=='Admit' &
myData$Female=='Yes'))/length(which(myData$Female=='Yes'))*100
length(which(myData$Decision=='Admit' &
myData$Female=='No'))/length(which(myData$Female=='No'))*100
#c.
sortedData <- myData[order(myData$HSGPA, decreasing = TRUE),]
length(which(sortedData[1:10,]$Female == "No"))
#d.
sortedData <- myData[order(myData$SAT),]
length(which(sortedData[1:10,]$Female =="Yes"))
#e.
sortedData <- myData[myData$Female=="Yes" & myData$Decision=="Admit",]
sortedData <- sortedData[order(sortedData$SAT),]
sortedData[1,]$SAT
sortedData[dim(sortedData)[1],]$SAT
sortedData <- myData[myData$Female=="No" & myData$Decision=="Admit",]
sortedData <- sortedData[order(sortedData$SAT),]
sortedData[1,]$SAT
sortedData[dim(sortedData)[1],]$SAT
```

16.

a.

x_1	x_2	x_3
124	3.8	55
196	4.5	32

Two observations remain with complete data sets

b.

x_1	x_2	x_3

248	3.5	56
124	3.8	55
150	4.5	74
196	4.5	32
179.5	6.2	63

Three missing values are replaced. The means of x_1, x_2, and x_3 are 179.5, 4.5, and 56, respectively.

17.

a. There are 25 observations in the subset.

x_1	x_2	x_3	x_4
544	616	5/14/85	1
⋮	⋮	⋮	⋮
186	218	3/8/81	1

b.

The average for x_1 in the $x4 = 1$ subset is 293.2083.
The average for x_1 in the $x4 = 0$ subset is 286.25.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData$x3 <- as.Date(myData$x3, format = "%m/%d/%Y")
cutoffdate <- as.Date("05/01/1975", format = "%m/%d/%Y")
subset1 <- myData[myData$x3>=cutoffdate, ]
dim(subset1)
#b
subset2 <- split(myData, myData$x4)
mean(subset2$`0`$x1)
mean(subset2$`1`$x1)
```


18.

a.

x_2	x_3	x_4
Own	189	33
Rent	120	28
…	…	…
Own	136	23

There are 0 missing values in the three remaining variables

b.

x_2	x_3	x_4
Rent	120	28

Rent	128	23
…	…	…
Rent	128	13

35 observations remain. The average values for x_3 and x_4 are 170.6571 and 21.2286, respectively.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[, -c(1,5)]
myData[!complete.cases(myData),]
#b
myData <- myData[myData$x2!="Own" & myData$x3 >= 150, ]
dim(myData)
mean(myData$x3)
mean(myData$x4)
```

19.

a. x_1–x_5 all have at least one missing value
b. Observations 11, 19, 26, 30, 36, 53, 65, 90, and 91 have missing values.
c. There are 11 missing values in the data set
d.

x_1	x_2	x_3	x_4	x_5
1	Own	189	33	0.0546
0	Rent	120	28	0.0091
…	…	…	…	…
1	Own	189	33	0.0546

91 complete observations remain.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData[!complete.cases(myData),]
#b
which(!complete.cases(myData))
#c
Table(is.na(myData))
#d
omissionData <- na.omit(myData)
dim(omissionData)
```

20.

a. There are two missing values for x_1, four for x_2, two for x_3, one for x_4, and two for x_5.

b. Missing values for x_1, x_2, x_3, x_4, and x_5 are imputed as 0, "Rent," 146.4184, 21.9697, and 0.1438, respectively.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData[!complete.cases(myData), ]
#b
myData$x2 <- as.factor(myData$x2)
summary(myData)
```

21.

a. There are two missing values for x_1, these are imputed as 292.24. After imputation the mean is 292.24.
b. There are two missing values for x_2, these are imputed as 288.16. After imputation the mean is 288.16.
c. There are three missing values for x_4, these are imputed as 22. After imputation the median is 22.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
is.na(myData$x1)
myData$x1[!complete.cases(myData$x1)] <- mean(myData$x1, na.rm = TRUE)
mean(myData$x1)
#b
is.na(myData$x2)
myData$x2[!complete.cases(myData$x2)] <- mean(myData$x2, na.rm = TRUE)
mean(myData$x2)
#c
is.na(myData$x4)
myData$x4[!complete.cases(myData$x4)] <- median(myData$x4, na.rm =
TRUE)
median(myData$x4)
```


22.

a. Variables x_1, x_2 and x_4 have at least one missing value.
b. Observations 13, 18, 21, 25, 35, 40, and 48 have missing values.
c. There are seven missing values.
d. 45 observations remain with complete data sets.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#b
```

```
myData[!complete.cases(myData), ]
omissionData <- na.omit(myData)
View(omissionData)
```

23.

a.

x_1	x_2	x_3
29	Female	42,000
23	Female	66,000
…	…	…
27	Female	47,000

b. Variables $x_1 - x_3$ have at least one missing value.

c. Observations 21, 50, 60, 71, 87, 101, 141, 151, and 162 have missing values.

d. There are 10 missing values in the data set.

e. 293 observations remain with complete data sets.

f.

Male

x_1	x_3
56	174,000
34	95,000
…	…
37	61,000

There are 163 observations in this subset.

Female

x_1	x_3
29	42,000
23	66,000
…	…
27	47000

There 130 are observations in this subset.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[,-4]
#b
myData[!complete.cases(myData), ]
#c
which(!complete.cases(myData))
#d
```

```
table(is.na(myData))
#e
omissionData <- na.omit(myData)
dim(omissionData)
#f
subset1 <- split(omissionData, omissionData$x2)
dim(subset1$Female)
dim(subset1$Male)
```

24.

a. Variables x_1, x_3–x_5 all have at least one missing value.
b. Observations 15, 24, 33, 34, 43, 63, 66, 78, and 95 have missing values.
c. There are nine missing values in the data set.
d. Two missing values of x_1 were replaced.
e. After replacing missing values of x_3, x_4, and x_5, their means in the new data set are 3.8832, 1264.804, and 3.1330 respectively.
f. There are 99 observations remain in the dataset.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[,-c(2, 6, 7)]
myData[!complete.cases(myData), ]
#b
which(!complete.cases(myData))
#c
table(is.na(myData))
#d
myData$x1[!complete.cases((myData$x1))] <- "Unknown"
length(which(myData$x1 == "Unknown"))
#e
myData$x3[!complete.cases(myData$x3)] <- mean(myData$x3, na.rm = TRUE)
mean(myData$x3)
myData$x4[!complete.cases(myData$x4)] <- median(myData$x4, na.rm =
TRUE)
mean(myData$x4)
myData$x5[!complete.cases(myData$x5)] <- mean(myData$x5, na.rm = TRUE)
mean(myData$x5)
#f
which(myData$x1 =="F" & myData$x4 <1020)
subset1 <- myData[-c(which(myData$x1 =="F" & myData$x4 <1020)), ]
dim(subset1)
```

25.

a.

2018 Population ≥ 5,000,000

There are 23 observations in this subset.

2018 Population < 5,000,000

There are 27 observations in this subset.

b.

There are nine observations to be removed.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
subset1 <- myData[myData$'2018' >= 5000000,  ]
subset2 <- myData[myData$'2018' < 5000000,  ]
#b
subset3 <- subset1[subset1$'2018' <= 10000000,  ]
```

26.

a. There are missing values in the variable Travel Plan. All other variables are complete.

b. 300 observations were removed.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
is.na(myData)
which(!complete.cases(myData))
myData[!complete.cases(myData), ]
#b
omissionData <- na.omit(myData)
View(omissionData)
dim(myData)
dim(omissionData)
```

27.

a.

College	FoodSpend	Income	TravelPlan
No	1,892.37	77,626	1
No	6,865.77	77,626	1

2 observations remain in the subset.

b. There are no missing values for either variable, but if there were missing values of the variable FoodSpend would be impute as its mean, 4420.278, and missing values for the variable income would be impute as its median, 55,400. The

average values for variables FoodSpend and Income remain unchanged as 4,420.278 and 55,400, respectively.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
myData <- myData[ , -2]
#a
subset1 <- myData[myData$Income>75000 & myData$TravelPlan == 1, ]
dim(subset1)
#b
FoodSpendMean <- mean(myData$FoodSpend, na.rm = TRUE)
IncomeMedian <- median(myData$Income, na.rm = TRUE)
myData$FoodSpend[is.na(myData$FoodSpend)] <- FoodSpendMean
myData$Income[is.na(myData$Income)] <- IncomeMedian
FoodSpendMean
IncomeMedian
```

28.

a. There are three missing values in the data set, the variable Yards is missing for observation 25, Attempts for observation 28, and Interceptions for observation 29. All other variables and observations are complete.
b. Three observations were removed using the omission strategy.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
is.na(myData)
which(!complete.cases(myData))
myData[!complete.cases(myData), ]
#b
omissionData <- na.omit(myData)
View(omissionData)
dim(myData)
dim(omissionData)
```

29.

a. Four of the observations had less than five touchdowns and were removed, additionally one of the observations had more than 20 interceptions and were removed. A total of five observations were removed from this process.
b. There are no missing values for the variable Comp, but there were missing values they would be input as its mean, 239.5349. The missing value in the variable A_{tt} was input as its mean, 384.3333. There are no missing values in the variable P_{ct}, but there were missing values they would be input as its mean, 61.3465. The missing value in the variable Y_{ds} was input as its mean, 2,718.762.

There are no missing values in the variable Avg, but there were missing values they would be input as its mean, 6.9256. There are no missing values in the variable Y_{ds}/G, but there were missing values they would be input as its mean, 221.407. There are no missing values in the variable Td, but there were missing values they would be input as its median, 18. The missing value for Int was input as its median, 8. After these replacements, the new means for the variables Comp, A_{tt}, P_{ct}, Y_{ds}, Avg, Y_{ds}/G, T_d, and Int are 239.5349, 384.3333, 61.3465, 2718.762, 6.9256, 221.407, 16.5581, and 9.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
subset1 <- myData[myData$TD >= 5,  ]
subset2 <- subset1[subset1$Int <= 20,  ]
#b
TDMedian <- median(myData$TD, na.rm = TRUE)
IntMedian <- median(myData$Int, na.rm = TRUE)
myData$TD[is.na(myData$TD)] <- TDMedian
myData$Int[is.na(myData$Int)] <- IntMedian
mean(myData$TD)
mean(myData$Int)
```


30.

a. 4 subsets were created: “Dept of HR,” “IT Dept,” “Public Utilities,” and “Sustainability and Environ Dept.”
b. There are two missing values in the “IT Dept” Hourly rate variable, all other subsets and variables are complete.
c. The missing values for Hourly rate in “IT Dept” were input as its mean, 52.1508. After this input, the average Hourly rates in the “Dept of HR,” “IT Dept,” “Public Utilities,” and “Sustainability and Environ Dept” are 47.0281, 52.1508, 41.1517, and 46.8476 respectively.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
subset1 <- myData[myData$Department == "Public Utilities",  ]
subset2 <- myData[myData$Department == "Sustainability & Environ Dept",  ]
subset3 <- myData[myData$Department == "IT Dept",  ]
subset4 <- myData[myData$Department == "Dept Of HR",  ]
#b
subset1[!complete.cases(subset1),  ]
subset2[!complete.cases(subset2),  ]
subset3[!complete.cases(subset3),  ]
subset4[!complete.cases(subset4),  ]
```

```
#c
ITMedian <- median(subset3$`Hourly Rate`, na.rm = TRUE)
subset3$`Hourly Rate`[is.na(subset3$`Hourly Rate`)] <- ITMedian
mean(subset3$`Hourly Rate`)
```

31.

a.

Name	Dividend	PE	Book Value	52 Week Low	52 Week high	Market Cap	EBITDA
Acuity Brands Inc	0.25	29.68	39.5	193.06	280.89	9	0.5862
Advance Auto Parts	0.15	24.51	39.66	132.98	177.83	11.18	1.12
…	…	…	…	…	…	…	…
Zions Bancorp	0.71	22.75	34.10	23.02	48.33	9.17	0.0000

189 observations remain.

b. 130 observations remain.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData <- myData[, -c(2, 5)]
myData1 <- myData[-which(myData$`Book Value`<myData$`Market Cap`),]
dim(myData1)
#b
myData2 <- myData1[-which(myData1$PE > 25),]
dim(myData2)
```

32.

a. All variables other than Name and Market Cap have missing values. Observation 20 is missing for the variable Price; Observation 173 is missing for variable Dividend; Observations 38 is missing for the variable PE; Observation 26 is missing for variable EPS; Observation 98 is missing for the variable Book Value; Observation 26 and 154 are missing for the variable 52 week low; Observation 46 is missing for the variable 52 week high; and Observation 51 is missing for variable the EBITDA. There are nine missing values in the data set.

b. Eight observations were removed.

c. Missing values for variable PE, EPS, and EBITDA were imputed as 21.915, 3.255, and 1.845, respectively. There were no missing values for Market Cap, but if there were, they would be input as 21.17.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
is.na(myData)
which(!complete.cases(myData))
myData[!complete.cases(myData), ]
#b
omissionData <- na.omit(myData)
View(omissionData)
dim(myData)
dim(omissionData)
#c
PEMedian <- median(myData$PE, na.rm = TRUE)
EPSMedian <- median(myData$EPS, na.rm = TRUE)
MCMedian <- median(myData$`Market Cap`, na.rm = TRUE)
EBITDAMedian <- median(myData$EBITDA, na.rm = TRUE)
myData$PE[is.na(myData$PE)] <- PEMedian
myData$EPS[is.na(myData$EPS)] <- EPSMedian
myData$`Market Cap`[is.na(myData$`Market Cap`)] <- MCMedian
myData$EBITDA[is.na(myData$EBITDA)] <- EBITDAMedian
myData$Price[is.na(myData$Price)] <- 'M'
myData$Dividend[is.na(myData$Dividend)] <- 'M'
myData$`Book Value`[is.na(myData$`Book Value`)] <- 'M'
myData$`52 week high`[is.na(myData$`52 week high`)] <- 'M'
myData$`52 week low`[is.na(myData$`52 week low`)] <- 'M'
PEMedian
EPSMedian
MCMedian
EBITDAMedian
```

33.

a. 39 individuals live in an urban area and have more than three members in their households.

b.

Urban	Siblings	White	Christian	FamilySize	Income
1	4	1	1	6	52,000
1	1	1	1	3	55,000
…	…	…	…	…	…
1	2	1	0	2	43,000

This subset contains 21 observations

Urban	Siblings	White	Christian	FamilySize	Income

1	8	1	1	5	0
1	1	1	1	4	40,000
…	…	…	…	…	…
1	2	1	1	1	0

This subset contains 48 observations

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
length(which(myData$Urban ==1 & myData$FamilySize>3))
#b
subset1 <- myData[which(myData$Urban ==1 & myData$Income > 40000), ]
subset2 <- myData[-which(myData$Urban ==1 & myData$Income > 40000), ]
```

34.

a. Observation 26 is missing for the variable Siblings, Observation 13 is missing for the variable Height, Observation 47 is missing for the variable Weight, and Observations 17 and 51 are missing for the variable Income. All other variables are complete. There is a total of five missing values.
b. Five observations were removed
c. Observation 13 is missing for the variable Height, and Observation 47 is missing for the variable Weight. The missing value for height was imputed with its mean 67.3088, and the missing value for weight was imputed with its mean, 150.8971. The missing value for siblings was imputed with its median, 2, and the missing values for income were imputed with its median, 33361.There were no missing values for FamilySize, but if there were, they would be imputed with its median, 4.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
is.na(myData)
which(!complete.cases(myData))
myData[!complete.cases(myData), ]
#b
omissionData <- na.omit(myData)
View(omissionData)
dim(myData)
dim(omissionData)
#c
heightMean <- mean(myData$Height, na.rm = TRUE)
weightMean <- mean(myData$Weight, na.rm = TRUE)
siblingMedian <- median(myData$Siblings, na.rm = TRUE)
familysizeMedian <- median(myData$FamilySize, na.rm = TRUE)
```

```
incomeMedian <- median(myData$Income, na.rm = TRUE)
myData$Height[is.na(myData$Height)] <- heightMean
myData$Weight[is.na(myData$Weight)] <- weightMean
myData$Siblings[is.na(myData$Siblings)] <- siblingMedian
myData$FamilySize[is.na(myData$FamilySize)] <- familysizeMedian
myData$Income[is.na(myData$Income)] <- incomeMedian
heightMean
weightMean
siblingMedian
familysizeMedian
incomeMedian
```

35.

a. Average of x_1 values assigned to Group 1: 156.6667
b. Number of observations are assigned to Group 1 –$x2$ is: 1
c. Number of observations are assigned to Group 2 –$x3$ is: 5

R Code:

```
# Import the data file into a data frame (table) and label it myData.
myData <- data.frame("x1" = c(248,124,210,150,196,234),
                     "x2" = c(3.5,3.8,1.6,4.8,4.5,6.2),
                     "x3" = c(78,55,66,74,32,63))
#a
x1Bins <- quantile(myData$x1, probs = seq(0,1, by =.5))
myData$x1Binned <- cut(myData$x1, breaks = x1Bins, labels = c("1", "2"),
include.lowest = TRUE, right = TRUE)
mean(myData[myData$x1Binned == 1, ]$x1)
#b
myData$x2Binned <- cut(myData$x2, breaks = 3, labels = c("1", "2", "3"),
include.lowest = TRUE)
length(which(myData$x2Binned ==1))
#c
x3Bins <- c(min(myData$x3),50, max(myData$x3))
myData$x3Binned <- cut(myData$x3, breaks = x3Bins, labels = c("1", "2"),
include.lowest = TRUE)
length(which(myData$x3Binned == 2))
```


36.

a. Number of observations are assigned to Group 1- $x1$: 11
b. Number of observations are assigned to Group 2- $x2$: 13
c. Number of observations are assigned to Group 1- $x3$: 8

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
```

```
x1Bins <- quantile(myData$x1, probs = seq(0,1, by =1/3))
myData$x1Binned <- cut(myData$x1, breaks = x1Bins, labels = c("1", "2", "3"),
include.lowest = TRUE, right = TRUE)
length(which(myData$x1Binned == 1))
#b
myData$x2Binned <- cut(myData$x2, breaks=3, labels = c("1", "2", "3"),
include.lowest = TRUE)
length(which(myData$x2Binned == 2))
#c
x3Bins <- c(min(myData$x3),50000, 100000, max(myData$x3))
myData$x3Binned <- cut(myData$x3, breaks = x3Bins, labels = c("1", "2", "3"),
include.lowest = TRUE)
length(which(myData$x3Binned == 1))
```

37.

a. Number of observations assigned to group medium is: 9
b. Number of observations assigned to group high is: 4
c. Number of observations assigned to group low is: 13

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
x1Bins <- quantile(myData$x1, probs = seq(0,1, by =1/3))
myData$x1Binned <- cut(myData$x1, breaks = x1Bins, labels = c("low",
"medium", "high"), include.lowest = TRUE, right = TRUE)
length(which(myData$x1Binned == "medium"))
#b
myData$x2Binned <- cut(myData$x2, breaks=3, labels = c("low", "medium",
"high"), include.lowest = TRUE)
length(which(myData$x2Binned == "high"))
#c
x3Bins <- c(min(myData$x3),20, 30, max(myData$x3))
myData$x3Binned <- cut(myData$x3, breaks = x3Bins, labels = c("low",
"medium", "high"), include.lowest = TRUE)
length(which(myData$x3Binned == "low"))
```

38.

a. Binned
b. The number of observations has score 431 is: 0;
   The number of observations has score 222 is: 1

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
x1Bins <- quantile(myData$x1, probs = seq(0,1, by =1/5))
```

```
myData$x1Binned <- cut(myData$x1, breaks = x1Bins, labels = c("1", "2", "3",
"4", "5"), include.lowest = TRUE, right = TRUE)
x2Bins <- quantile(myData$x2, probs = seq(0,1, by =1/5))
myData$x2Binned <- cut(myData$x2, breaks = x2Bins, labels = c("1", "2", "3",
"4", "5"), include.lowest = TRUE, right = TRUE)
x3Bins <- quantile(myData$x3, probs = seq(0,1, by =1/5))
myData$x3Binned <- cut(myData$x3, breaks = x3Bins, labels = c("1", "2", "3",
"4", "5"), include.lowest = TRUE, right = TRUE)
#b
myData$score <- paste(myData$x1Binned, myData$x2Binned,
myData$x3Binned, sep = "")
length(which(myData$score == "431"))
length(which(myData$score == "222"))
```

39.

a. The average of sum is: 382.2
b. The average of difference is: 51.4

R Code:

```
myData <- data.frame("x1" = c(248,124,150,196,240), "x2" =
c(350,148,130,145,180))
#a
myData$Sum <- myData$x1 + myData$x2
mean(myData$Sum)
#b
myData$Difference <-  abs(myData$x1 - myData$x2)
mean(myData$Difference)
```


40.

a. The average of difference: −1.91
b. The average percent difference is: 0.2273
c. The average logarithm: 12.84

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
myData$Difference <- myData$x1-myData$x2
mean(myData$Difference)
#b
myData$PercentDifference <- (myData$x1-myData$x2)/myData$x1
mean(myData$PercentDifference)
#c
myData$Log <- log(myData$x3)
mean(myData$Log)
```

41.

a. The average of difference is: −44.1
b. The average percent difference is: 0.0235
c. The number of observations in Group 2- log: 30;

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
myData$Difference <- myData$x2-myData$x1
mean(myData$Difference)
#b
myData$PercentDifference <- (myData$x2-myData$x1)/myData$x1
mean(myData$PercentDifference)
#c
myData$Log <- log(myData$x3)
myData$Logx3bin <- cut(myData$Log, breaks=5, labels = c("1", "2", "3", "4",
"5"), include.lowest = TRUE)
length(which(myData$Logx3bin == "2"))
```

42.

a. The average difference is: 29.01
b. The average month value is: 6
c. Group 3 has the most observations (27 observations).

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
myData$Date1 <-as.Date(myData$Date1,format='%m-%d-%Y')
myData$Date2 <- as.Date(myData$Date2, format="%m/%d/%Y")
myData$DifferenceInDays <-
abs(round(difftime(myData$Date1,myData$Date2),digit=2))
myData$DifferenceInYear <- myData$DifferenceInDays/365.25
mean(myData$DifferenceInYear)
#b
myData$Month <- months(myData$Date1)
myData$Month <-match(myData$Month, month.name)
mean(myData$Month)
#c
myData$Monthbin <- cut(myData$Month, breaks=4, labels = c("1", "2", "3",
"4"), include.lowest = TRUE)
table(myData$Monthbin)
```

43.

a. Number of observations assigned to Group 4- 2017: 13
b. Number of observations assigned to Group 2- 2018: 6;

There are no states in higher groups for the 2018 population than the 2017 population

c. Number of observations assigned to Group 2- 2018: 21

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
Bins2017 <- quantile(myData$`2017`, probs =seq(0,1, by=1/4))
myData$Binned_2017 <- cut(myData$`2017`, breaks = Bins2017, labels =
c("1","2","3","4"), include.lowest=TRUE, right=TRUE)
table(myData$Binned_2017)
#b
myData$Binned_2018 <- cut(myData$`2018`, breaks=4, labels =
c("1","2","3","4"), include.lowest=TRUE)
table(myData$Binned_2018)
#c
myData$Binned_2018_2 <-cut(myData$`2018`, breaks=c(0,1000000,
5000000,Inf), labels=c("1","2","3"))
table(myData$Binned_2018_2)
```

44.

a. The average of difference is: 40271
b. The average % difference is: 0.0056
c. Number of observations in Group 2- log: 10
d. Number of observations in Group 2- square root: 18
e. They are different.

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
myData$Difference <- myData$`2018`-myData$`2017`
mean(myData$Difference)
#b
myData$PercentDifference <- myData$Difference/myData$`2017`
mean(myData$PercentDifference)
#c
myData$Log <- log(myData$`2018`)
myData$Binned_Log <- cut(myData$Log, breaks=5, labels = c("1","2","3","4",
"5"), include.lowest=TRUE)
table(myData$Binned_Log)
#d
myData$SquareRoot <- sqrt(myData$`2018`)
myData$Binned_SR <- cut(myData$SquareRoot, breaks=5, labels =
c("1","2","3","4", "5"), include.lowest=TRUE)
table(myData$Binned_SR)
```

```
#e
myData[, c("Binned_Log", "Binned_SR")]
```

45.

a. The average day since purchase: 411.94
b. Number of observations with RFM = 555: 1; the total spending is 78,658.
c. Number of observations in Group 2- log: 10
d. Number of observations in Group 2- average order size: 23
e. The groups are different.

R Code:

```
# Import the data file into a data frame (table) and label it myData
#a
myData$Date <-as.Date(myData$LastTransactionDate,format='%m-%d-%Y')
EndDate <- as.Date("01/01/2022", format="%m/%d/%Y")
myData$DSL <- abs(round(difftime(myData$Date,EndDate),digit=2))
round(mean(myData$DSL), digit=2)
#b
myData$DateReverse <- as.numeric(myData$DSL * -1)
recencyBins <- quantile(myData$DateReverse, probs =seq(0,1, by=1/5))
myData$Binned_Recency <- cut(myData$DateReverse, breaks = recencyBins,
labels = c("1","2","3","4","5"), include.lowest=TRUE, right=TRUE)
FrequencyBins <- quantile(myData$Frequency, probs =seq(0,1, by=1/5))
myData$Binned_Frequency <- cut(myData$Frequency, breaks = FrequencyBins,
labels = c("1","2","3","4","5"), include.lowest=TRUE, right=TRUE)
MonetaryBins <- quantile(myData$TotalSpending, probs =seq(0,1, by=1/5))
myData$Binned_Monetary <- cut(myData$TotalSpending, breaks =
MonetaryBins, labels = c("1","2","3","4","5"), include.lowest=TRUE,
right=TRUE)
myData$RFM <-paste(myData$Binned_Recency, myData$Binned_Frequency,
myData$Binned_Monetary)
table(myData$RFM)
mean(myData$TotalSpending[which(myData$RFM=="5 5 5")])
#c
myData$LogSpending <-round(log(myData$TotalSpending),digit=2)
myData$Binned_Log <- cut(myData$LogSpending, breaks=5, labels =
c("1","2","3","4","5"), include.lowest=TRUE, right=FALSE)
table(myData$Binned_Log)
#d
myData$AverageOrderSize <-myData$TotalSpending/myData$Frequency
myData$Binned_AOS <- cut(myData$AverageOrderSize, breaks=5, labels =
c("1","2","3","4","5"), include.lowest=TRUE, right=FALSE)
table(myData$Binned_AOS)
#e
table(myData$Binned_Log)
```

table(myData$Binned_AOS)

46.

a. Number of observations for Group 4- spending last year: 25
b. Number of observations are assigned to Group 3- this year: 21
c. Number of observations are assigned to Group 2- this year: 36
d. Average difference: 124.7
e. The average percentages difference: 56.42%
f. Average age: 28.2
g. Month with the most frequent: September

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
SpendingLastBins <- quantile(myData$SpendingLastYear, probs =seq(0,1,
by=1/4))
myData$Binned_SpendingLast <- cut(myData$SpendingLastYear, breaks =
SpendingLastBins, labels = c("1","2","3", "4"), include.lowest=TRUE,
right=TRUE)
table(myData$Binned_SpendingLast)
#b
myData$Binned_SpendingThis <- cut(myData$SpendingThisYear, breaks=4,
labels = c("1","2","3","4"), include.lowest=TRUE)
table(myData$Binned_SpendingThis)
#c
myData$Binned_SpendingThis_2 <-cut(myData$SpendingThisYear,
breaks=c(0,250,500,Inf), labels=c("1","2","3"))
table(myData$Binned_SpendingThis_2)
#d
myData$Difference <- myData$SpendingThisYear-myData$SpendingLastYear
mean(myData$Difference)
#e
myData$PercentDifference <- (myData$SpendingThisYear-
myData$SpendingLastYear)/myData$SpendingLastYear
mean(myData$PercentDifference)
#f
myData$Date <-as.Date(myData$BirthDate,format='%m-%d-%Y')
EndDate <- as.Date("01/01/2022", format="%m/%d/%Y")
myData$Age <- abs(round(difftime(myData$Date,EndDate)/365.25,digit=2))
myData$Age <- floor(myData$Age)
round(mean(myData$Age), digit=2)
#g
myData$BirthMonth <- months(myData$Date)
table(myData$BirthMonth)
```

47.

a. Average age of the engineer: 49.75
b. Number of observations in Group 3: 31
c. The number of observations in Group 4- salary: 2
d. The number of engineers with High: 20

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData$Date <-as.Date(myData$BirthDate,format='%m-%d-%Y')
EndDate <- as.Date("01/01/2022", format="%m/%d/%Y")
myData$Age <- abs(round(difftime(myData$Date,EndDate)/365.25,digit=2))
myData$Age <- floor(myData$Age)
round(mean(myData$Age), digit=2)
#b
SalaryBins <- quantile(myData$Salary, probs =seq(0,1, by=1/3))
myData$Binned_Salary <- cut(myData$Salary, breaks = SalaryBins, labels =
c("1","2","3"), include.lowest=TRUE, right=TRUE)
table(myData$Binned_Salary)
#c
myData$Binned_Salary <- cut(myData$Salary, breaks=4, labels =
c("1","2","3","4"), include.lowest=TRUE)
table(myData$Binned_Salary)
#d
myData$Binned_Certificates <-cut(myData$Certficates, breaks=c(0,2,4,Inf),
labels=c("Low","Medium","High"))
table(myData$Binned_Certificates)
```


48.

a. Average BMI: 29.50
b. Average age: 45.29
c. Number of observations in Group 4- binned-age: 80
d. Number of observations in Group 5- binned-exercise: 83
e. Number of observations in Group 1- BMI: 80
f. Number of “555”: 2

R Code:

```
## Import the data file into a data frame (table) and label it myData.
#a
myData$BMI <- myData$Weight/myData$Height^2
mean(myData$BMI)
#b
endDate<-as.Date("01/01/2022", "%m/%d/%Y")
myData$Age <- as.integer(floor(difftime(endDate, myData$BirthDate)/365.25))
mean(myData$Age)
```

```
#c
AgeBins <- quantile(myData$Age, probs=seq(0, 1, by=0.20))
myData$Binned_Age <- cut(myData$Age, breaks=AgeBins, labels=c("1", "2",
"3", "4","5"),include.lowest = TRUE, right = TRUE)
length(which(myData$Binned_Age=="4"))
#d
ExerciseBins <- quantile(myData$Exercise, probs=seq(0, 1, by=0.20))
myData$Binned_Exercise <- cut(myData$Exercise, breaks=ExerciseBins,
labels=c("5", "4", "3", "2","1"),include.lowest = TRUE, right = TRUE)
length(which(myData$Binned_Exercise=="5"))
mean(myData[which(myData$Binned_Exercise == '5'),]$Exercise)
#e
BMIBins <- quantile(myData$BMI, probs=seq(0, 1, by=0.20))
myData$Binned_BMI <- cut(myData$BMI, breaks=BMIBins, labels=c("1", "2",
"3", "4","5"),include.lowest = TRUE, right = TRUE)
length(which(myData$Binned_BMI=="1"))
mean(myData[which(myData$Binned_BMI == '1'),]$BMI)
#f
myData$Risk <- paste(myData$Binned_Age, myData$Binned_Exercise,
myData$Binned_BMI)
head(myData$Risk)
length(which(myData$Risk=="5 5 5"))
```

49.

a.

Sex	Income	Decision
1	95,000	Approve
0	65,000	Approve
1	55,000	Need More Information
0	72,000	Reject
0	58,000	Approve
0	102,000	Approve

Male is the reference category

b.

Sex	Income	Decision
Female	95,000	0
Male	65,000	0
Female	55,000	1
Male	72,000	1
Male	58,000	0
Male	102,000	0

Approve is the reference category

c.

Sex	Income	Decision
Female	95,000	1
Male	65,000	1
Female	55,000	3
Male	72,000	2
Male	58,000	1
Male	102,000	1

One observation has a category score of 2.

50.

a. There are 22 observations in the "Other" category. Prior to the reduction category "E" was in the least observations, appearing just 8 times, category "D" appears 14 times, and category "B" appear 16 times. "D" and "E" were combined because they are the least frequent categories.

x_1	x_2	x_3
B	D	Category4
A	F	Category4
…	…	…
B	B	Category2

b.

x_1	x_2	x_3
Other	4	Category4
A	6	Category4
…	…	…
Other	2	Category2

The average category score for x_2 is 3.6200.

c. Three dummy variables should be created.

x_1	x_2	x_3_Cat1	x_3_Cat2	x_3_Cat3
Other	4	0	0	0
A	6	0	0	0
…	…	…	…	…
Other	2	0	1	0

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
table(myData$x1)
myData$x1 <- ifelse(myData$x1 %in% c("D", "E"), "Other", myData$x1)
```

```
table(myData$x1)
#b
myData$x2 <- ifelse(myData$x2 == "A", 1, ifelse(myData$x2 == "B", 2,
ifelse(myData$x2 == "C", 3, ifelse(myData$x2 == "D", 4, ifelse(myData$x2 ==
"E", 5, 6)))))
mean(myData$x2)
#c
myData$x2_Category2 <- ifelse(myData$x3 == "Category2", 1, 0)
myData$x2_Category3 <- ifelse(myData$x3 == "Category3", 1, 0)
myData$x2_Category4 <- ifelse(myData$x3 == "Category4", 1, 0)
```

51.

a.

Birthdate	Reject	Approve
2/4/84	1	0
9/21/58	0	0
…	…	…
8/20/97	0	0

2 dummy variables should be created.

b.

Birthdate	Approve	Not Approve
2/4/84	0	1
9/21/58	0	1
…	…	…
8/20/97	0	1

There are 21 observations in the "Not approve" category.

R Code:

```
# Exercise_2.51
# Import the data file into a data frame (table) and label it myData.
#a
myData$LoanDecision_Approve <- ifelse(myData$LoanDecision == "Approve",
1, 0)
myData$LoanDecision_Reject <- ifelse(myData$LoanDecision == "Reject", 1, 0)
#b
myData$LoanDecisionNew <- ifelse(myData$LoanDecision %in% c("Reject",
"Need more information"), "Not approve", myData$LoanDecision)
table(myData$LoanDecisionNew)
```

52.

a.

x_1	x_2
2	Europe
3	Asia

…	…
1	Australia

15 observations have a category score of 3.

b.

x_1	x_2
2	Europe
3	Asia
…	…
1	Other

There are three observations in the "Other" category.

c.

x_1	x_2 Europe	x_2 Asia	x_2 NorthAmerica	x_2 Africa
2	1	0	0	0
3	0	1	0	0
…	…	…	…	…
1	0	0	0	0

Four dummy variables should be created.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData$x1_Score <- ifelse(myData$x1 == "Low", 1, ifelse(myData$x1 ==
"Medium", 2, 3))
table(myData$x1_Score)
#b
table(myData$x2)
myData$x2 <- ifelse(myData$x2 %in% c("Australia", "South America"), "Other",
myData$x2)
table(myData$x2)
#c
myData$x2_Africa <- ifelse(myData$x2 == "Africa", 1, 0)
myData$x2_Asia <- ifelse(myData$x2 == "Asia", 1, 0)
myData$x2_Europe <- ifelse(myData$x2 == "Europe", 1, 0)
myData$x2_North_America <- ifelse(myData$x2 == "North America", 1, 0)
```


53.

a.

22 observations have a category score of 3.

b.

x_1	Yes	x_3
1	1	C
2	1	B
…	…	…

3	1	A

1 dummy variable should be created.

c.

There are 15 observations in the "E" category.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData$x1_Score <- ifelse(myData$x1 == "S", 1, ifelse(myData$x1 == "M", 2,
3))
table(myData$x1_Score)
#b
table(myData$x2)
myData$x2_Yes <- ifelse(myData$x2 == "Yes", 1, 0)
#c
table(myData$x3)
myData$x3 <- ifelse(myData$x3 %in% c("B", "D"), "E", myData$x3)
table(myData$x3)
```

54.

a. The variables LoanType, PropertyType, and Purpose are each nominal data because they do not have naturally ordered categories.

b. "Conventional" is the most frequent category for LoanType.
"Multi-family" is the most frequent category for PropertyType.
"Refinancing" is the most frequent category for Purpose.

c.

Application	Federal Housing	Single-Family	Purchase
1	0	1	1
2	0	0	0
…	…	…	…
103	1	1	1

Three dummy variables are created in total.
"Conventional" is the reference category of LoanType.
"Multi-family" is the reference category of PropertyType.
"Refinancing" is the reference category of Purpose.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#b
table(myData$LoanType)
table(myData$PropertyType)
table(myData$Purpose)
#c
myData$LoanType_FederalHousing <- ifelse(myData$LoanType == "Federal
Housing", 1, 0)
```

```
myData$PropertyType_SingleFamily <- ifelse(myData$PropertyType ==
"Single-family", 1, 0)
myData$Purpose_Purchase <- ifelse(myData$Purpose == "Purchase", 1, 0)
```

55.

a.

Package	Status	Express	Same Day	Size
1	Delivered	0	0	XL
2	Damaged	1	0	S
…	…	…	0	…
75	Lost	0	1	M

Two dummy variables should be created. “Standard” is the reference category of Delivery.

b.

There are 21 observations in the “Other” category.

c.

Package	Status	Express	Same Day	Size
1	3	0	0	Other
2	2	1	0	S
…	…	…	0	…
75	1	0	1	M

The average status score of the 75 packages is 2.7067.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
table(myData$Delivery)
myData$Delivery_Express <- ifelse(myData$Delivery == "Express", 1, 0)
myData$Delivery_SameDay <- ifelse(myData$Delivery == "Same day", 1, 0)
#b
table(myData$Size)
myData$Size <- ifelse(myData$Size %in% c("XL", "L"), "Other", myData$Size)
table(myData$Size)
#c
myData$Status <- ifelse(myData$Status == "Lost", 1, ifelse(myData$Status ==
"Damaged", 2, 3))
mean(myData$Status)
```

56.

a.

Tower	*A*	Damage
1	1	Minor
2	1	None
…	…	…

98	0	Partial

One dummy variable should be created.

b.

Tower	*A*	Damage
1	1	1
2	1	0
…	…	…
98	0	2

The average damage score of the cell towers is 0.7959.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
table(myData$Model)
myData$Model_B <- ifelse(myData$Model == "B", 1, 0)
#b
myData$Damage_Score <- ifelse(myData$Damage == "Total", 4,
ifelse(myData$Damage == "Severe", 3, ifelse(myData$Damage == "Partial", 2,
ifelse(myData$Damage == "Minor", 1, 0))))
mean(myData$Damage_Score)
```

57.

a.

The average satisfaction score of the players is 3.7600.

b.

The average enjoyment score of the players is 4.7400.

c.

The average recommendation score of the players is 3.3800

d.

The “Role Play” category is the reference category. Two dummy variables should be created.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
myData <- data.frame(Game_Players)
#a
myData$Satisfaction <- ifelse(myData$Satisfaction == "Very Dissatisfied", 1,
ifelse(myData$Satisfaction == "Dissatisfied", 2, ifelse(myData$Satisfaction ==
"Neutral", 3, ifelse(myData$Satisfaction == "Satisfied", 4, 5))))
mean(myData$Satisfaction)
#b
myData$Enjoyment <- ifelse(myData$Enjoyment == "Very Low", 1,
ifelse(myData$Enjoyment == "Low", 2, ifelse(myData$Enjoyment == "Neutral",
3, ifelse(myData$Enjoyment == "High", 4, 5))))
mean(myData$Enjoyment)
```

```
#c
myData$Recommend <- ifelse(myData$Recommend == "Definitely Will Not", 1,
ifelse(myData$Recommend == "Will Not", 2, ifelse(myData$Recommend ==
"Neutral", 3, ifelse(myData$Recommend == "Will", 4, 5))))
mean(myData$Recommend)
#d
table(myData$Type)
myData$Type_Action <- ifelse(myData$Type == "Action", 1, 0)
myData$Type_Sports <- ifelse(myData$Type == "Sports", 1, 0)
```

58.

a.

BirthDate	Analyst	Diplomat	Explorer	Salary	Level	Certificates
7/31/73	0	0	1	48,000	Engineer I	0
8/29/67	0	1	0	44,000	Engineer I	5
…	…	…	…	…	…	…
6/9/72	0	0	1	76,000	Engineer II	4

Three dummy variables should be created. There are 14 observations in the reference category.

b.

The average level score of the engineers is 1.7600.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
myData$Personality_Diplomat <- ifelse(myData$Personality == "Diplomat", 1, 0)
myData$Personality_Explorer <- ifelse(myData$Personality == "Explorer", 1, 0)
myData$Personality_Sentinel <- ifelse(myData$Personality == "Sentinel", 1, 0)
table(myData$Personality)
#b
myData$Level <- ifelse(myData$Level == "Engineer I", 1, ifelse(myData$Level
== "Engineer II", 2, 3))
mean(myData$Level)
```

59.

a.

There are 97 observations in the “Other” category.

b.

Hispanic	Non-Hispanic Black	Other	Education	Weight	Height	BirthDate	Exercise
0	0	0	College	57	1.58	3/1/82	138
0	1	0	HS	80	1.71	2/14/60	249
…	…	…	…	…	…	…	…

0	0	1	Graduate	50	1.69	7/11/92	264

"Non-Hispanic white" is the reference category. Three dummy variables should be created.

c.

The average category score for Education is 2.4925.

R Code:

```
# Import the data file into a data frame (table) and label it myData.
#a
table(myData$Race)
myData$Race <- ifelse(myData$Race %in% c("American Indian", "Asian/Pacific Islander"), "Other", myData$Race)
table(myData$Race)
#b
myData$Race_Hispanic <- ifelse(myData$Race == "Hispanic", 1, 0)
myData$Race_NonHispanicBlack <- ifelse(myData$Race == "Non-Hispanic Black", 1, 0)
myData$Race_Other <- ifelse(myData$Race == "Other", 1, 0)
#c
myData$Education <- ifelse(myData$Education == "HS", 1,
ifelse(myData$Education == "Some College", 2, ifelse(myData$Education == "College", 3, 4)))
mean(myData$Education)
```

Chapter 4. Data Visualization Solutions

1.

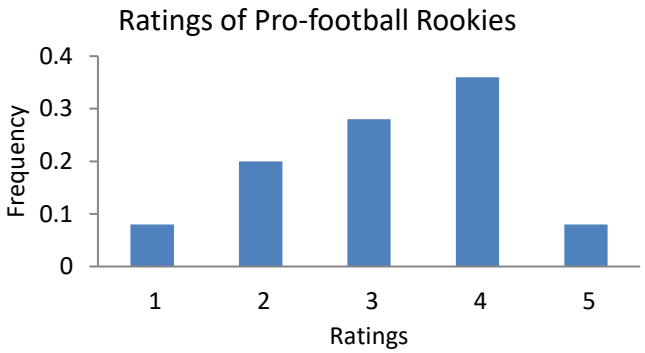
a. 22 (18 + 4) out of 50 rookies received a rating of 4 or better; 14 (10 + 4) out of 50 rookies received a rating of 2 or worse.

b.

Rating	Relative Frequency
1	4/50 = 0.08
2	10/50 = 0.20
3	14/50 = 0.28
4	18/50 = 0.36
5	4/50 = 0.08

8% of the rookies received a rating of 5.

c.



The rating of 4 has the highest frequency; the distribution is not symmetric, rather it is negatively skewed.

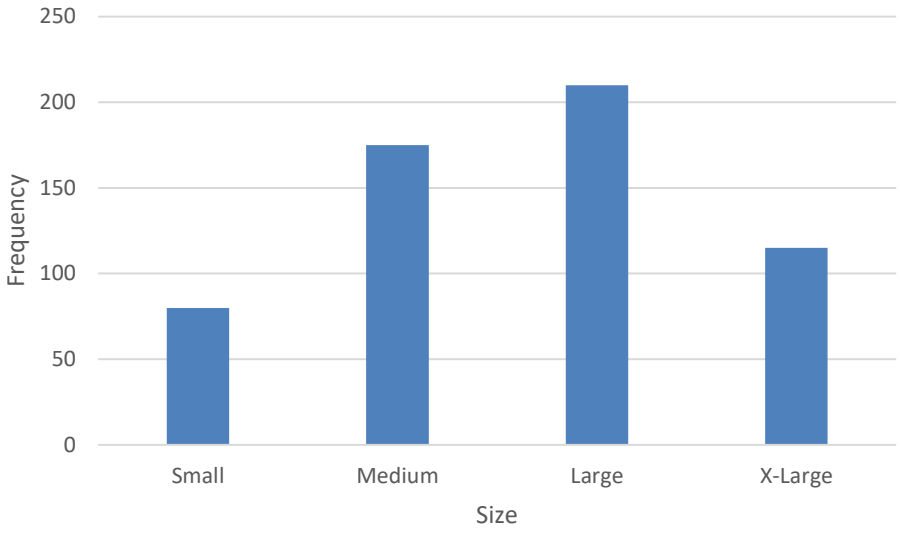
2.

a.

Size	Frequency	Relative Frequency
Small	80	80/580=0.138
Medium	175	175/580=0.302
Large	210	210/580=0.362
X-Large	115	115/580=0.198

The proportion of the sales for medium-sized shirts was 0.302.

b.



Sales of large-sized shirts had the highest frequency and sales of small-sized shirts had the lowest frequency.

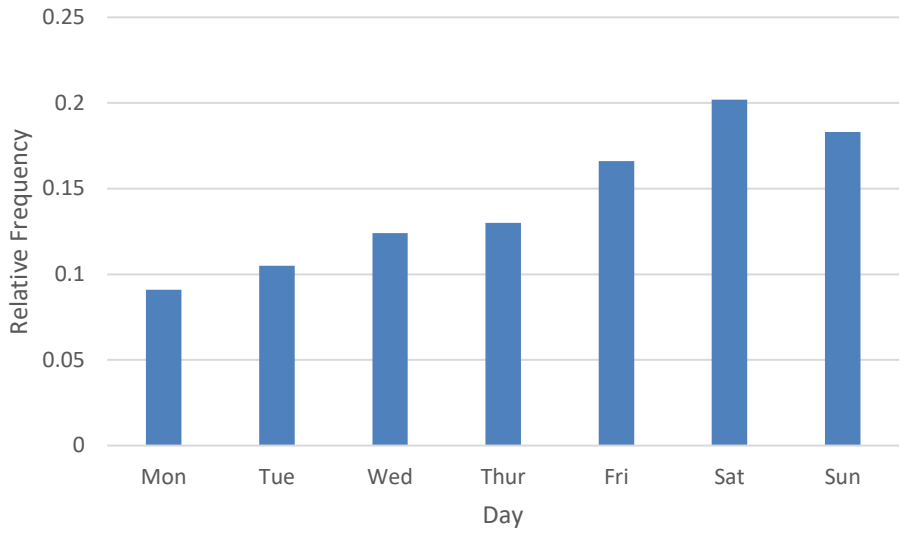
3.

a.

Day	Frequency	Relative Frequency
Mon	2504	2504/27500=0.091
Tue	2880	2880/27500=0.105
Wed	3402	3402/27500=0.124
Thur	3566	3566/27500=0.130
Fri	4576	4576/27500=0.166
Sat	5550	5550/27500=0.202
Sun	5022	5022/27500=0.183

The proportion of purchases that occurred on Wednesday was 0.124.

b.



The retailer saw the highest proportion of sales on the weekend. The lowest proportion of sales occurred on Monday.

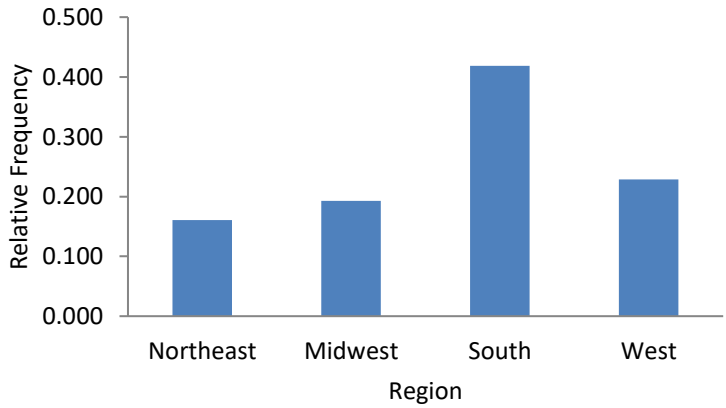
4.

a.

Region	Relative Frequency
Northeast	0.161
Midwest	0.193
South	0.418
West	0.228

19.3% of people in the Midwest are living below the poverty level.

b.



The South has the highest relative frequency as compared to the other three regions, which are roughly equal.

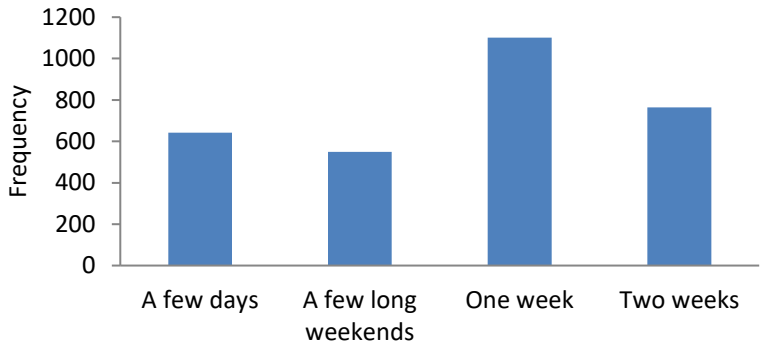
5.

a.

Response	Frequency
A few days	0.21(3057) = 642
A few long weekends	0.18(3057) = 550
One week	0.36(3057) = 1,101
Two weeks	0.25(3057) = 764

Approximately 1,101 people are going to take a one-week vacation.

b.



"One week" has the highest frequency as compared to the other categories, which are roughly equal.

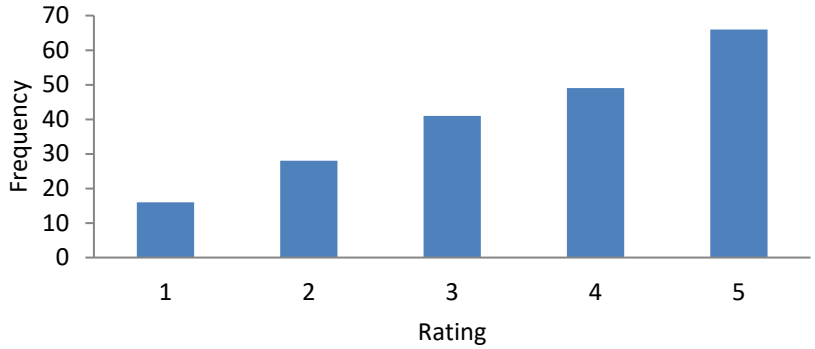
6.

a.

Rating	Frequency
1	16
2	28
3	41
4	49
5	66

A rating of 5 has the highest frequency

b.



The higher ratings have the higher frequencies.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency distribution
Rating_Frequency <- table(myData$Rating)
Rating_Frequency
View(Rating_Frequency)

#Constructing bar chart
barplot(Rating_Frequency,main="Bar chart for the Rating variable",
ylab="Number of Responses", col="blue")
abline(h=0)
```


7.

a.

Quality	Frequency
Poor	18
Fair	26
Good	75
Excellent	31

Good is the most common response.

b.

Patients seem to be healthy, with the majority rated Good or Excellent.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency distribution
Quality_Frequency <- table(myData$Quality)
Quality_Frequency
View(Quality_Frequency)

#Constructing bar chart
barplot(Quality_Frequency,main="Bar chart for the Quality variable", ylab="Number of Responses", col="blue")
abline(h=0)
```

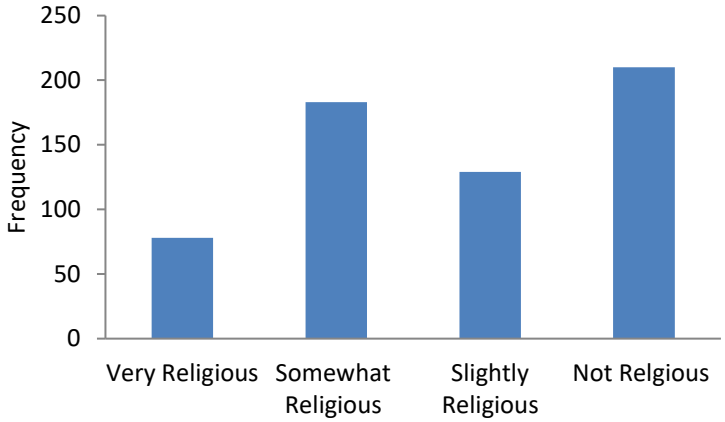

8.

a.

Faith	Frequency
Very Religious	78
Somewhat Religious	183
Slightly Religious	129
Not Religious	210

Not Religious is the most common response.

b.



210/600 = 0.35; about 35% responded "Not Religious" which is consistent with the Pew Research survey.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency and relative distributions
Faith_Frequency <- table(myData$Faith)
Faith_Frequency
View(Faith_Frequency)
Faith_Proportion <- prop.table(Faith_Frequency)
Faith_Proportion
View(Faith_Proportion)

#Constructing bar chart and using cex.names options so that category names are not truncated
barplot(Faith_Frequency,main="Bar chart for the Faith variable", ylab="Number of Responses", cex.names=0.75, col="blue")
abline(h=0)
```


9.

a.

Sex	Frequency	Relative Frequency
Female	492	0.492
Male	508	0.508

492 of the employees are females; the proportion of males is 0.508.

b.

Personality	Frequency	Relative Frequency
Analyst	116	0.116

Diplomat	324	0.324
Explorer	404	0.404
Sentinel	156	0.156

Explorer is the most frequent personality type (404); the proportion of Diplomats is 0.324.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency and relative distributions for the Sex variable
Sex_Frequency <- table(myData$Sex)
Sex_Frequency
View(Sex_Frequency)
Sex_Proportion <- prop.table(Sex_Frequency)
Sex_Proportion
View(Sex_Proportion)
#Creating frequency and relative distributions for the Personality variable
Personality_Frequency <- table(myData$Personality)
Personality_Frequency
View(Personality_Frequency)
Personality_Proportion <- prop.table(Personality_Frequency)
Personality_Proportion
View(Personality_Proportion)
```

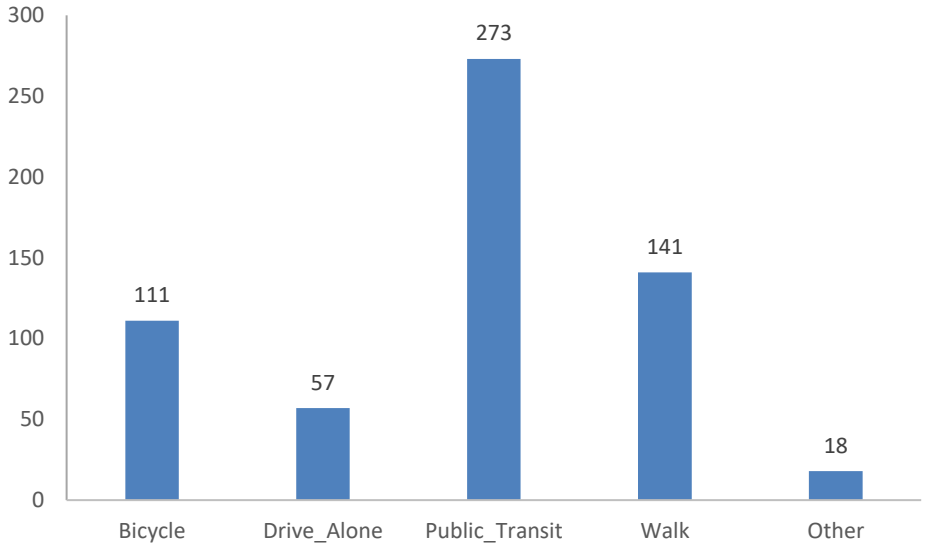

10.

a.

Mode	Frequency	Relative Frequency
Bicycle	111	0.185
Drive_Alone	57	0.095
Public_Transit	273	0.455
Walk	141	0.235
Other	18	0.030

57 of the respondents commute by driving alone; the proportion of respondents who commute by bicycle is 0.185.

b.



Using public transit is the most common way of commuting to work. This is not surprising given that the university is in an urban setting.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency and relative distributions for the Mode variable
Mode_Frequency <- table(myData$Mode)
Mode_Frequency
View(Sex_Mode)
Mode_Proportion <- prop.table(Mode_Frequency)
Mode_Proportion
View(Mode_Proportion)
#Constructing bar chart and using cex.names and ylim options so that category
names and y-axis are not truncated
barplot(Mode_Frequency,main="Bar chart for the Mode variable", ylab="Number of
Responses", cex.names=0.75, ylim=c(0,300), col="blue")
abline(h=0)
```


11.

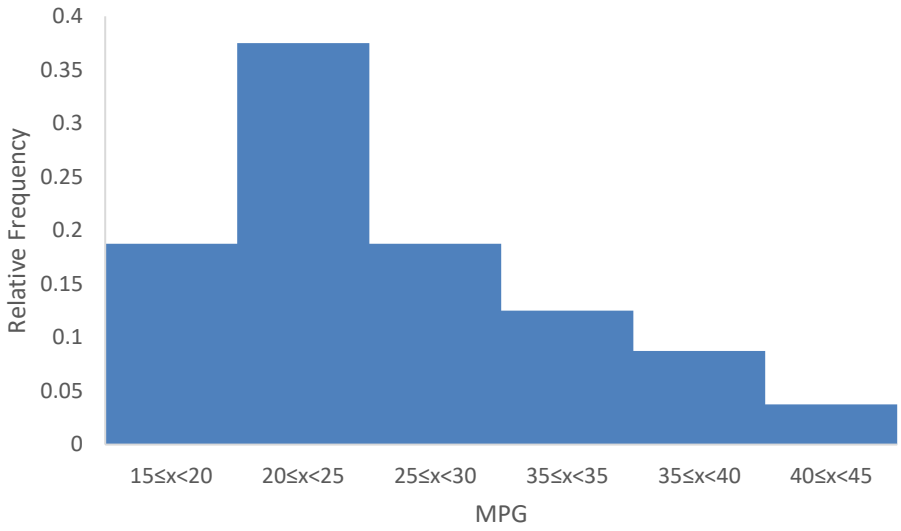
a.

Average MPG	Relative frequency
$15 \le x < 20$	15/80 = 0.1875
$20 \le x < 25$	30/80 = 0.3750
$25 \le x < 30$	15/80 = 0.1875
$30 \le x < 35$	10/80 = 0.1250
$35 \le x < 40$	7/80 = 0.0875

$40 \leq x < 45$	3/80 = 0.0375

0.375 of the cars got at least 20 but less than 25 mpg; 0.875 of the cars got less than 35 mpg. Since 0.875 got less than 35 mpg, 0.125 of the cars got 35 mpg or more.

b.



The distribution is not symmetric; it is positively skewed.

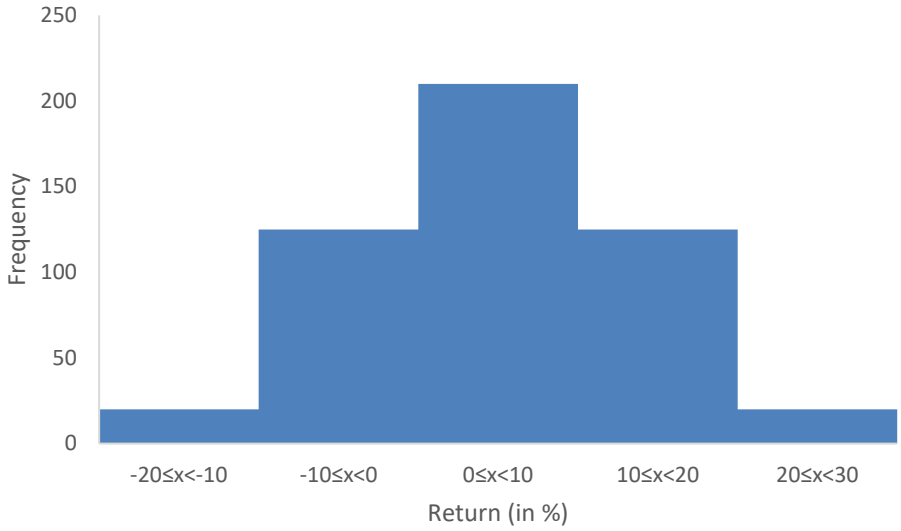
12.

a.

Return (%)	Frequency
$-20 \le x < -10$	0.04(500) = 20
$-10 \le x < 0$	0.25(500) = 125
$0 \le x < 10$	0.42(500) = 210
$10 \le x < 20$	0.25(500) = 125
$20 \le x < 30$	0.04(500) = 20

125 stocks had returns of at least 10% but less than 20%.

b.



The distribution is symmetric.

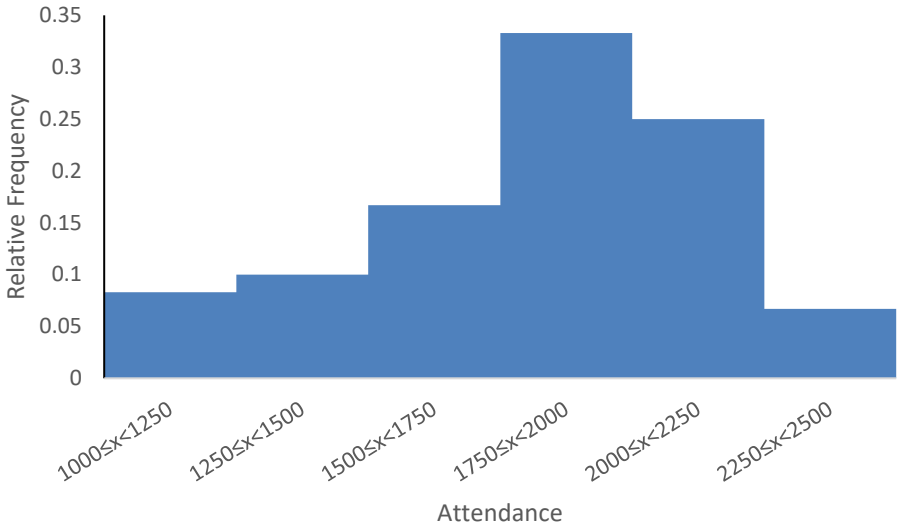
13.

a.

Attendance	Relative Frequency
$1{,}000 \le x < 1{,}250$	5/60 = 0.083
$1{,}250 \le x < 1{,}500$	6/60 = 0.100
$1{,}500 \le x < 1{,}750$	10/60 = 0.167
$1{,}750 \le x < 2{,}000$	20/60 = 0.333
$2{,}000 \le x < 2{,}250$	15/60 = 0.250
$2{,}250 \le x < 2{,}500$	4/60 = 0.067

The proportion of time that attendance was at least 1,750 but less than 2,000 was 0.33; the proportion of time that attendance was less than 1,750 was 0.35; therefore, the proportion of time that attendance was 1,750 or more was 0.65.

b.



The distribution is not symmetric; it is negatively skewed.

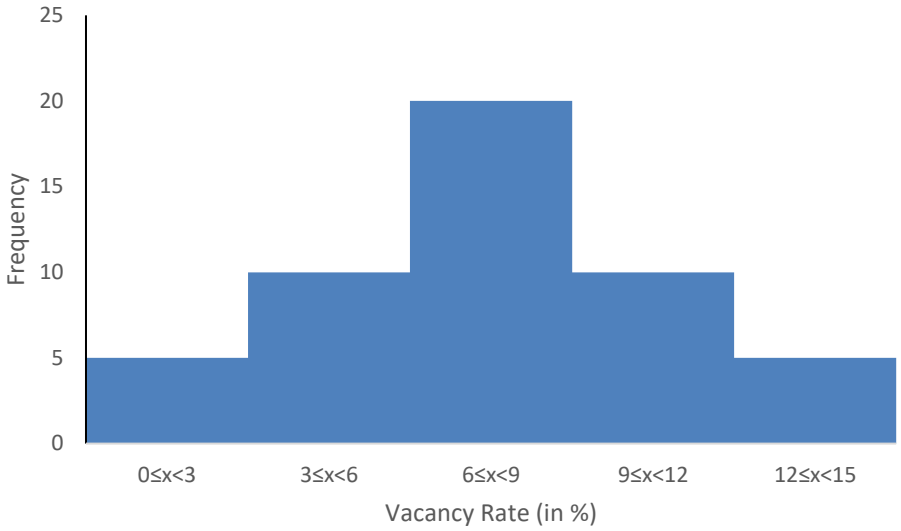
14.

a.

Vacancy Rate (%)	Relative Frequency
$0 \le x < 3$	0.10(50) = 5
$3 \le x < 6$	0.20(50) = 10
$6 \le x < 9$	0.40(50) = 20
$9 \le x < 12$	0.20(50) = 10
$12 \le x < 15$	0.10(50) = 5

Twenty cities had vacancy rates that were at least 6% but less than 9%. Fifteen cities had a vacancy rate of at least 9%.

b.



The distribution is symmetric.

15.

a. No. The distribution is not symmetric. It is positively skewed.
b. Forty-four percent of the states had median household income between $45,000 and $55,000.
c. Sixty-six percent of the states had median household income between $35,000 and $55,000.

16.

a. No. The distribution is not symmetric. It is positively skewed.
b. Over this time period, the stock price was between $50 and $250.
c. The $100 up to $150 interval has the highest relative frequency, which is about 0.44.

17.

a. No. The distribution is not symmetric. It is positively skewed.
b. Three (0.10×30) of the portfolio managers earned between $2,000,000 and $2,400,000.
c. About 26 (0.43×30+0.43×30=25.8) of the portfolio managers earned between $1,200,000 and $2,000,000.

18.

a.

Expenditures	Frequency	Relative Frequency
400 < x ≤ 700	5	0.05
700 < x ≤ 1000	14	0.14
1000 < x ≤ 1300	33	0.33
1300 < x ≤ 1600	25	0.25
1600 < x ≤ 1900	19	0.19
1900 < x ≤ 2200	4	0.04

14 customers spent between $701 and $1,000.

b. 0.52; 0.48.

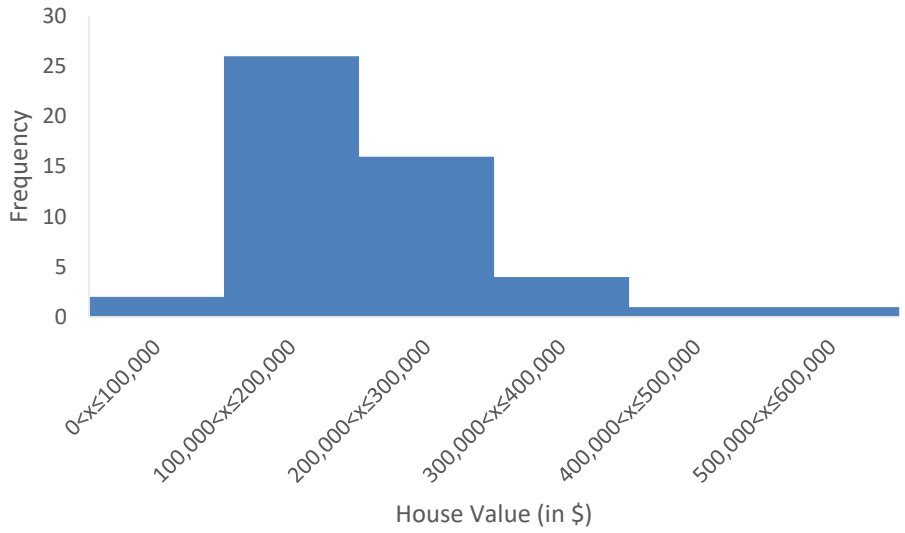
R Code:

```
#Import the data file into a data frame (table) and label it myData
#Creating frequency and relative distributions for the Expenditures variable
intervals <- seq(400, 2200, by=300)
Expenditures.cut <- cut(myData$Expenditures, intervals, left=FALSE, right=TRUE)
Expenditures.frequency <- table(Expenditures.cut)
Expenditures.frequency
View(Expenditures.frequency)
Expenditures.prop<-prop.table(Expenditures.frequency)
Expenditures.prop
View(Expenditures.prop)
```


19.

a.

House Value	Frequency
0 < x ≤ 100,000	2
100,000 < x ≤ 200,000	26
200,000 < x ≤ 300,000	16
300,000 < x ≤ 400,000	4
400,000 < x ≤ 500,000	1
500,000 < x ≤ 600,000	1



The interval 100,000 < x ≤ 200,000 has the highest frequency. Forty-four states (2+26+16=44) have median house values less than $300,000.

b. No, the distribution is not symmetric. It is positively skewed.

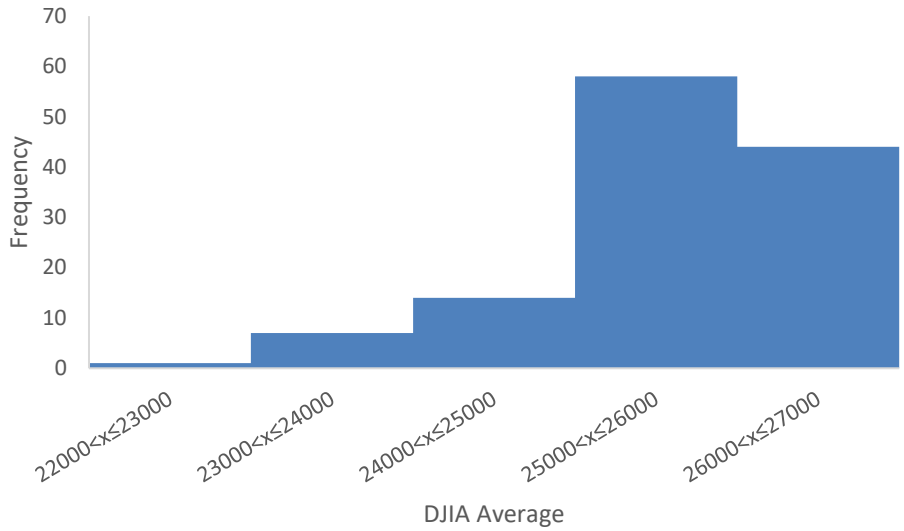
R Code:


```
#Import the data file into a data frame (table) and label it myData
#creating frequency distribution
intervals <- seq(0, 600000, by=100000)
House_Value.cut <- cut(myData$House_Value, intervals, left=FALSE, right=TRUE)
House_Value.frequency <- table(House_Value.cut)
House_Value.frequency
View(House_Value.frequency)

#creating histogram
hist(myData$House_Value, breaks=intervals, right=TRUE, main="Histogram for the
  House_Value variable", xlab="Median House Value (in $)", col="blue")
```

20.

a.

DJIA	Frequency
22000 < x ≤ 23000	1
23000 < x ≤ 24000	7
24000 < x ≤ 25000	14
25000 < x ≤ 26000	58
26000 < x ≤ 27000	44



The DJIA was more than 26,000 on 44 days in the first half of 2019.

b.

The distribution is not symmetric; it is negatively skewed.

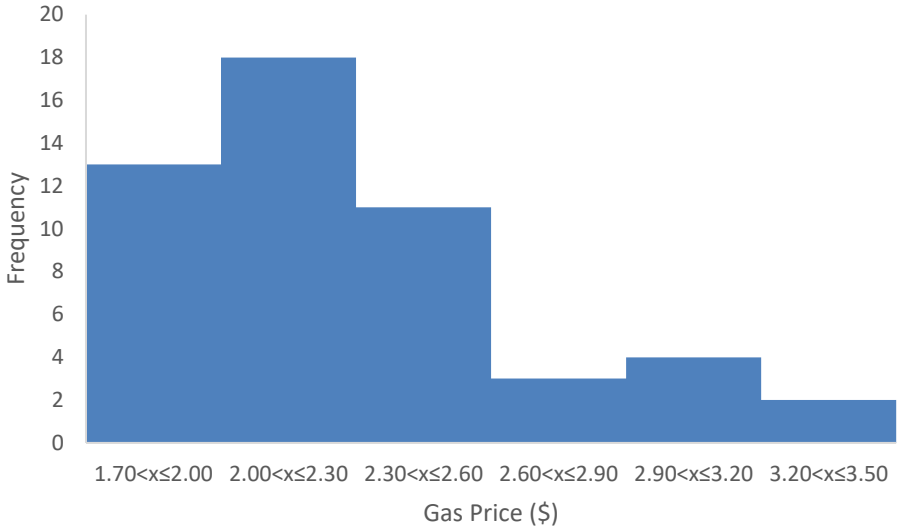
R Code:


```
#Import the data file into a data frame (table) and label it myData
#creating frequency distribution
intervals <- seq(22000, 27000, by=1000)
DJIA.cut <- cut(myData$DJIA, intervals, left=FALSE, right=TRUE)
DJIA.frequency <- table(DJIA.cut)
DJIA.frequency
View(DJIA.frequency)

#creating histogram
hist(myData$DJIA, breaks=intervals, right=TRUE, main="Histogram for
the DJIA variable", xlab="DJIA Daily Price Index", col="blue")
```

21.

a.

Gas Price	Frequency
1.70 < x ≤ 2.00	13
2.00 < x ≤ 2.30	18
2.30 < x ≤ 2.60	11
2.60 < x ≤ 2.90	3
2.90 < x ≤ 3.20	4
3.20 < x ≤ 3.50	2



The interval $2.00 < x \leq 2.30$ has the highest frequency. Nine states (3+4+2=9) have average gas prices that are greater than $2.60.

b. No, the distribution is not symmetric. It is positively skewed.

R Code:


```
#Import the data file into a data frame (table) and label it myData
 #creating frequency distribution
 intervals <- seq(1.70, 3.50, by=0.3)
 Price.cut <- cut(myData$Price, intervals, left=FALSE, right=TRUE)
 Price.frequency <- table(Price.cut)
 Price.frequency
 View(Price.frequency)

 #creating histogram
 hist(myData$Price, breaks=intervals, right=TRUE, main="Histogram for the Price
 variable", xlab="Price per Gallon (in $)", col="blue")
```

22.

a.

Salary	Frequency	Relative Frequency
$0 < x \leq 50{,}000$	29	0.29
$50{,}000 < x \leq 100{,}000$	41	0.41
$100{,}000 < x \leq 150{,}000$	18	0.18

150,000 < x ≤ 200,000	11	0.11
200,000 < x ≤ 250,000	1	0.01

18 employees earn between $100,000 and $150,000; the proportion that earns at most $200,000 is 0.99.

b.

Experience	Frequency	Relative Frequency
0 < x ≤ 6	22	0.22
6 < x ≤ 12	19	0.19
12 < x ≤ 18	20	0.20
18 < x ≤ 24	16	0.16
24 < x ≤ 30	23	0.23

19 employees have between 6 and 12 years of experience; the proportion of employees that has more than 24 years of experience is 0.23.

.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating frequency and relative distributions for the Salary variable
intervals <- seq(0, 250000, by=50000)
Salary.cut <- cut(myData$Salary, intervals, left=FALSE, right=TRUE)
Salary.frequency <- table(Salary.cut)
Salary.frequency
View(Salary.frequency)
Salary.prop<-prop.table(Salary.frequency)
Salary.prop
View(Salary.prop)
#creating frequency and relative distributions for the Experience variable
intervals <- seq(0, 30, by=6)
Experience.cut <- cut(myData$Experience, intervals, left=FALSE, right=TRUE)
Experience.frequency <- table(Experience.cut)
Experience.frequency
View(Experience.frequency)
Experience.prop<-prop.table(Experience.frequency)
Experience.prop
View(Experience.prop)
```

23. This graph does not correctly depict what has happened to company's stock price over this period. By using a relatively high value as an upper limit on the vertical axis ($500), the rise in stock price appears dampened.

24. This graph does not correctly depict what has happened to sales over the most recent five-year period. The vertical axis has been stretched so that the increase in sales appears more pronounced than warranted.

25.

	Sex		
Personality	Female	Male	Total
Analyst	55	61	**116**
Diplomat	164	160	**324**
Explorer	194	210	**404**
Sentinel	79	77	**156**
Total	**492**	**508**	**1,000**

.

a. 492; Explorers

b. 0.194; 0.160

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Personality, myData$Sex)
myTable
prop.table(myTable)
```

26.

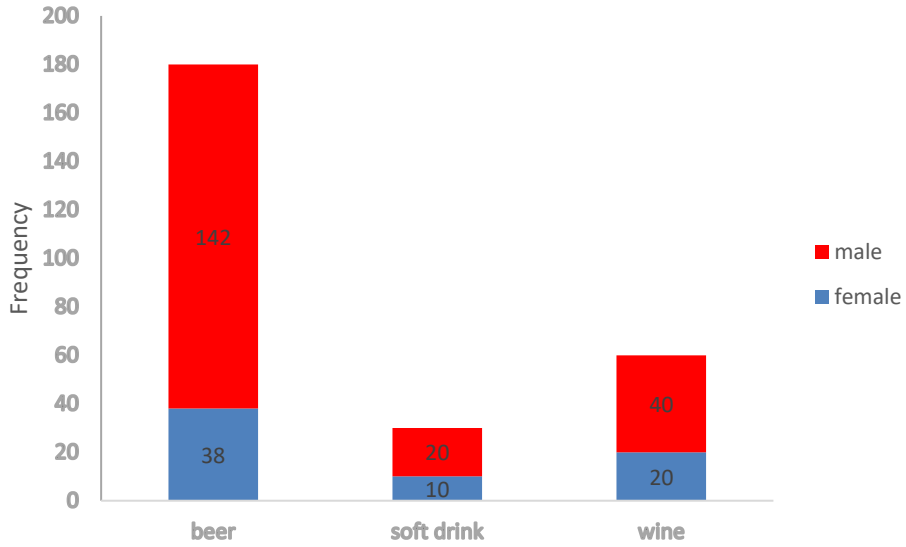
a.

	Sex		
Drink Choice	female	male	Total
beer	38	142	**180**
soft drink	10	20	**30**
wine	20	40	**60**
Total	**68**	**202**	**270**

202 of the customers were male; 60 of the customers drank wine.

b. 142/202 = 0.7030; 38/68 = 0.5588.

c.



The stacked column chart shows that beer is the popular drink at this bar, followed by wine and then soft drinks. Both men and women are more likely to choose beer over the other two options.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Sex, myData$Drink_Choice)
myTable
#creating a stacked column chart
barplot(myTable, col=c('blue','red'), legend=rownames(myTable), xlab="Drink
Choice", ylab="Count", ylim = c(0,200))
```

27.

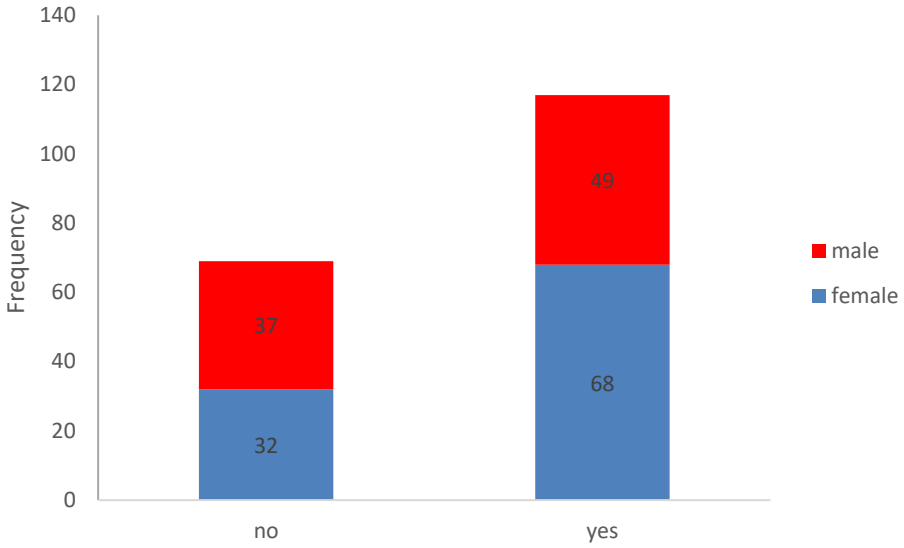
a.

	Sex		
Feasible	female	male	Total
no	32	37	**69**
yes	68	49	**117**
Total	**100**	**86**	**186**

100 of the students were female; 117 of the students thought it was feasible for men and women to be just friends.

b. 49/86=0.5698; 68/100 = 0.6800.

c.



The stacked column chart shows that yes is the most common answer. Both men and women are more likely to say that it is feasible for men and women to be friends. However, women are more likely to say that it is feasible for men and women to be friends (0.68 > 0.57).

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Sex, myData$Feasible)
myTable
#creating a stacked column chart
barplot(myTable, col=c('blue','red'), legend=rownames(myTable),
      ylab="Count", ylim=c(0,150))
```

28.

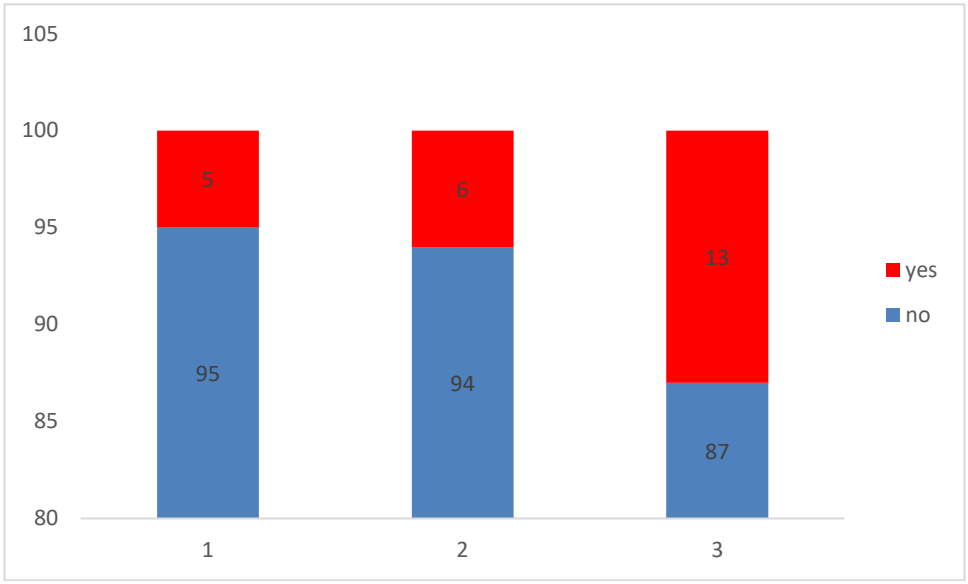
a.

	Shift			
Defective	1	2	3	**Total**
no	95	94	87	**276**
yes	5	6	13	**24**
Total	**100**	**100**	**100**	**300**

Five components from shift 1 were defective; 94 components from shift 2 were not defective.

b. 6/24=0.2500; 13/24 = 0.5417; components constructed during shift 3 seem to be defective at a higher rate.

c.



Defect rates seem to be highest in shift 3.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Defective, myData$Shift)
myTable
#creating a stacked column chart
barplot(myTable, col=c('blue','red'), legend=rownames(myTable),
        ylab="Count")
```

29.

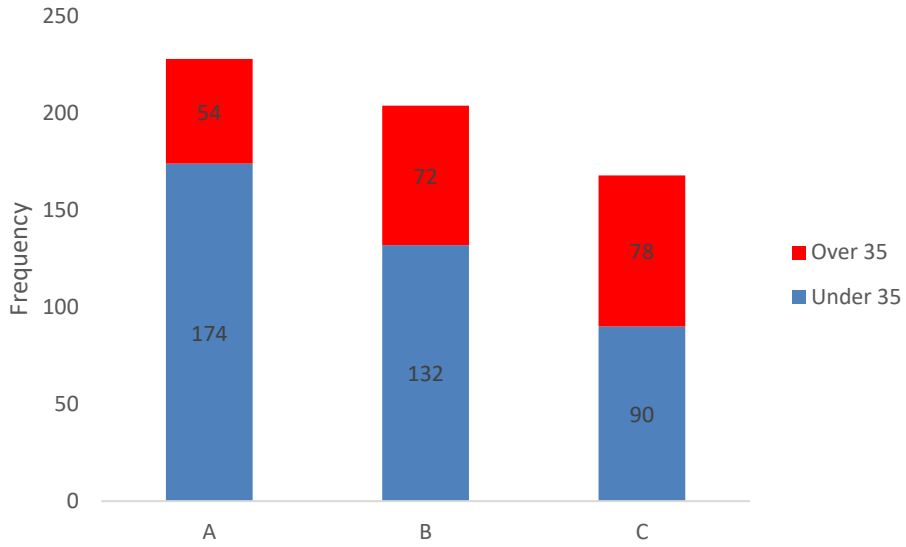
a.

	Under 35		
Brand	no	yes	Total
A	54	174	228
B	72	132	204
C	78	90	168
Total	**204**	**396**	**600**

228 purchases were from brand A; 396 purchases were from customers under 35.

b. 174/396=0.4394; 132/396=0.3333; 90/396=0.2273; The data suggests that customers under 35 prefer brand A.

c.



Younger customers seem to prefer A over B, and B over C, while older Customers seem to prefer C over B, and B over A

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Age, myData$Brand)
myTable
#creating a stacked column chart
barplot(myTable, col=c('blue','red'), legend=rownames(myTable),
      ylab="Count", ylim=c(0,250))
```

30.

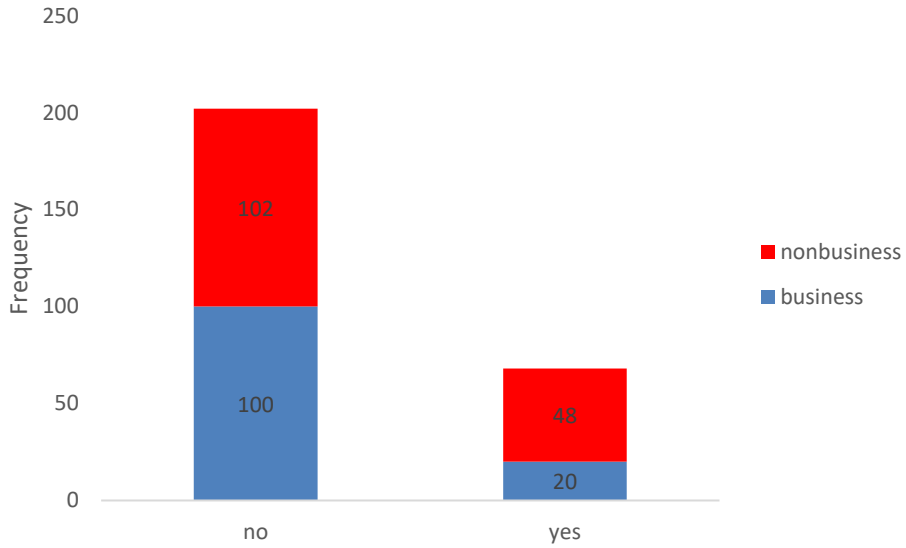
a.

	Major		
Study hard	business	nonbusiness	Total
no	100	102	202
yes	20	48	68
Total	120	150	270

120 students are business majors; 68 students study hard.

b. 20/120=0.1667; 48/150=0.3200; The data suggest that nonbusiness majors are more likely to study hard, which supports the report.

c.

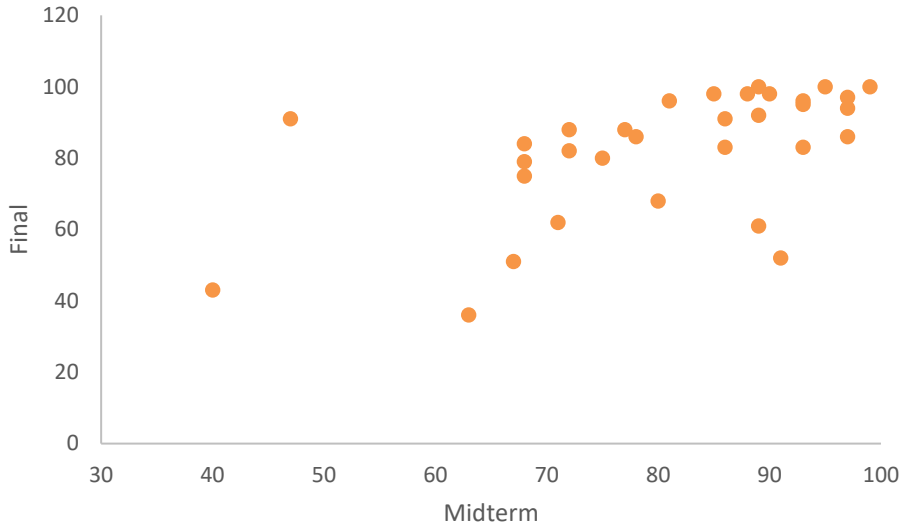


The majority of both business and nonbusiness students do not study hard, but nonbusiness students are more likely to study hard.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#creating a contingency table
myTable <- table(myData$Major, myData$Study_Hard)
myTable
#creating a stacked column chart
barplot(myTable, col=c('blue','red'), legend=rownames(myTable),
      ylab="Count")
```

31.

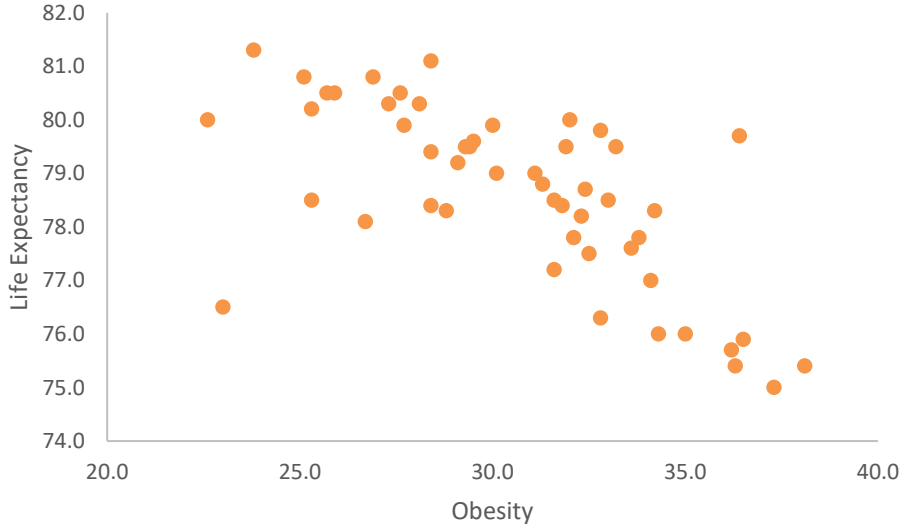


There appears to be a positive relationship between a student's midterm score and a student's final score.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing a scatterplot
plot(myData$Final ~ myData$Midterm, main ="Scatterplot of Final against Midterm", xlab = "Midterm", ylab = "Final", col="chocolate", pch=16)
```

32.

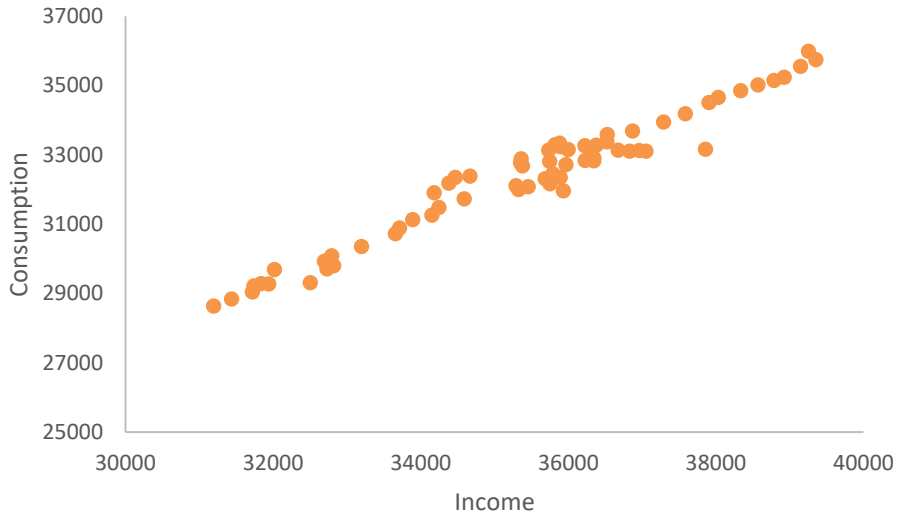


There appears to be a negative relationship between obesity and life expectancy.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing a scatterplot
plot(myData$Life_Expectancy ~ myData$Obesity, main ="Scatterplot of
Life_Expectancy against Obesity", xlab = "Obesity", ylab = "Life Expectancy",
col="chocolate", pch=16)
```

33.

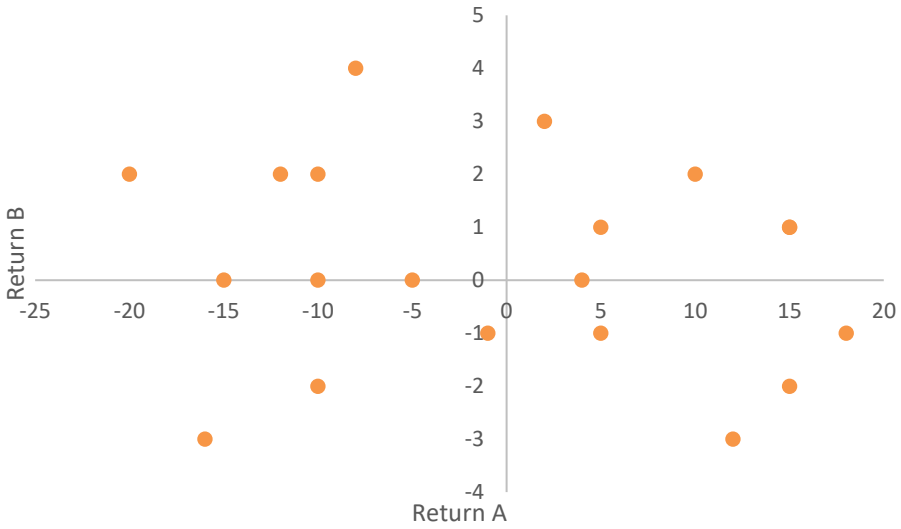


There appears to be a positive relationship between income and consumption.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing a scatterplot
plot(myData$Consumption ~ myData$Income, main ="Scatterplot of
Consumption against Income", xlab = "Income", ylab = "Consumption",
col="chocolate", pch=16)
```

34.



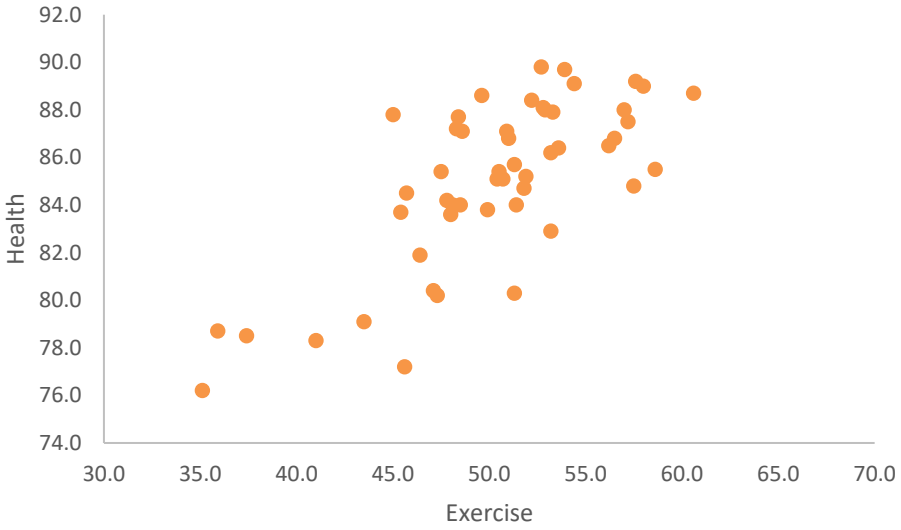
There appears to be no relationship between the returns of A and B, so investing in both would diversify risk.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing a scatterplot
plot(myData$Return_B ~ myData$Return_A, main ="Scatterplot of Return_B
against Return_A", xlab ="Annual Return of A (in %)", ylab = "Annual Return of
B (in %)", col="chocolate", pch=16)
abline(h=0)
abline(v=0)
```

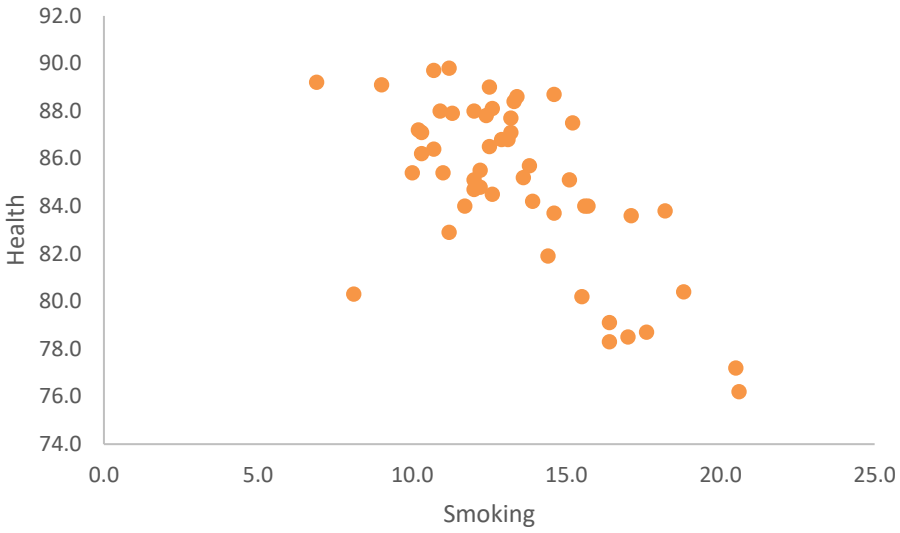
35.

a.



There appears to be a positive relationship between health and exercise.

b.



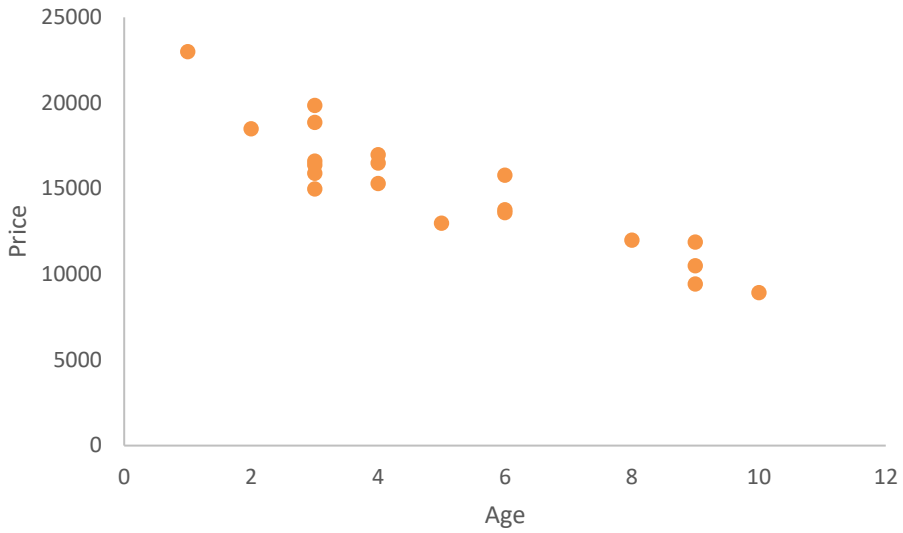
There appears to be a negative relationship between health and smoking.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing scatterplots
plot(myData$Health ~ myData$Exercise, main ="Scatterplot of Health against Exercise", xlab = "Exercise", ylab = "Health", col="chocolate", pch=16)
plot(myData$Health ~ myData$Smoking, main ="Scatterplot of Health against Smoking", xlab = "Smoking", ylab = "Health", col="chocolate", pch=16)
```

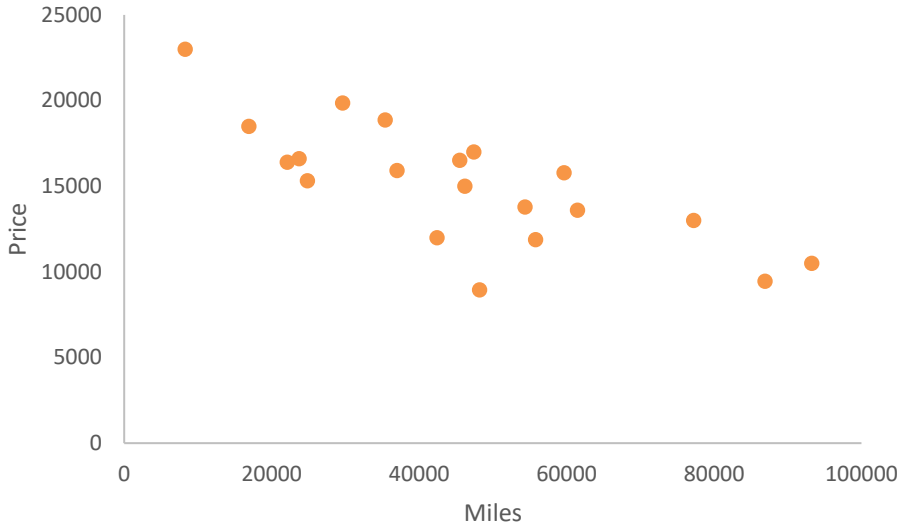
36.

a.



There appears to be a negative relationship between the price of a car and its age.

b.

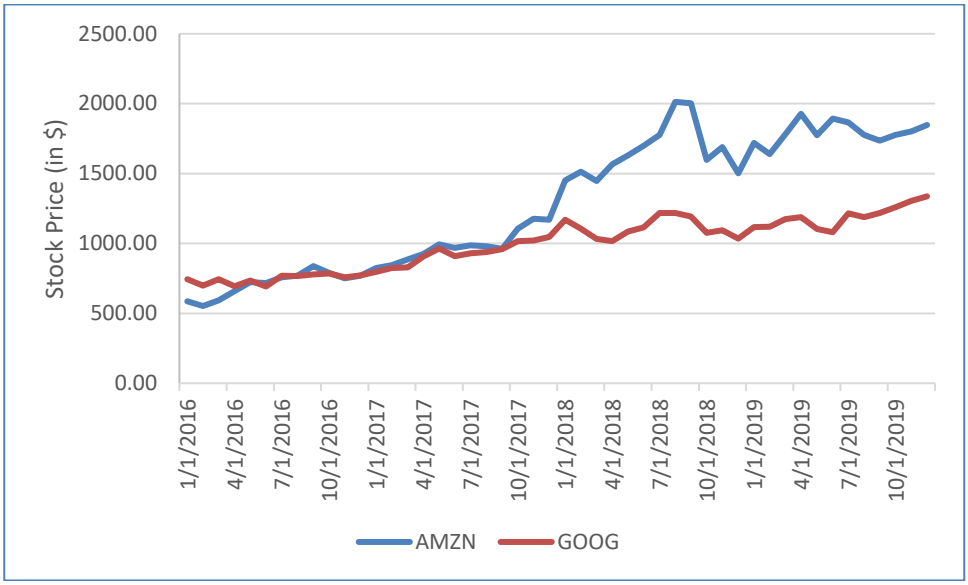


There appears to be a negative relationship between the price of a car and its mileage.

R Code:

```
#Import the data file into a data frame (table) and label it myData
#constructing scatterplots
plot(myData$Price ~ myData$Age, main ="Scatterplot of Price against Age", xlab
= "Age (in years)", ylab = "Price (in $)", col="chocolate", pch=16)
plot(myData$Price ~ myData$Mileage, main ="Scatterplot of Price against
Mileage", xlab = "Mileage", ylab = "Price (in $)", col="chocolate", pch=16)
```

37.

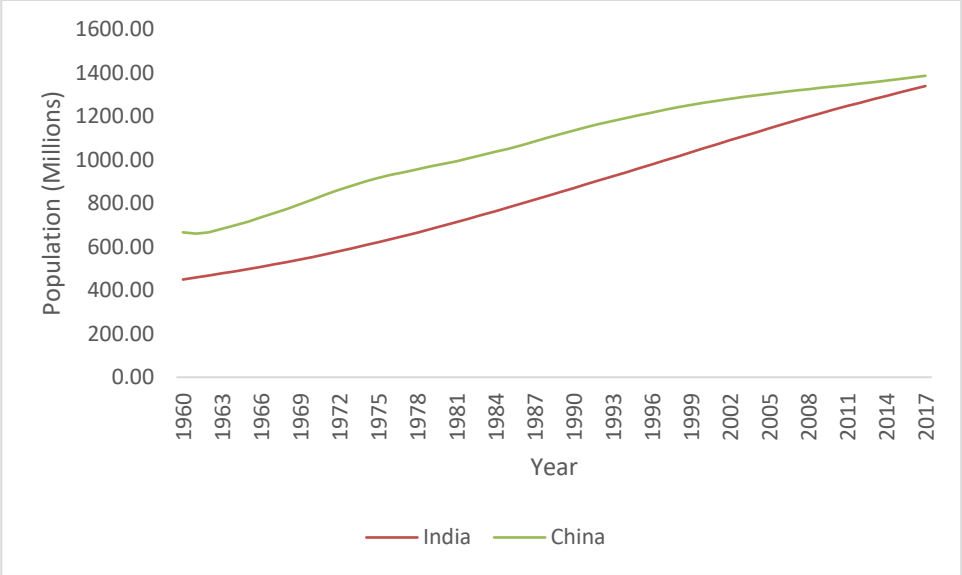


GOOG has an upward trend. Amazon seems to have a greater trajectory, even though it had a steep drop in the latter half of 2018.

The R Code

```
# Import the data into a data frame (table) and label it myData.
# Obtain line chart for AMZN
plot(myData$AMZN~ myData$Date, col="blue", type="l", ylab="Stock Price (in $)",
     xlab="Date")
#Incorporate red line for GOOG
lines(myData$GOOG~myData$Date, col="red", type="l")
#Incorporate legend
legend("topleft", legend=c("AMZN","GOOG"), col=c("blue","red"), lty=1)
```


38.



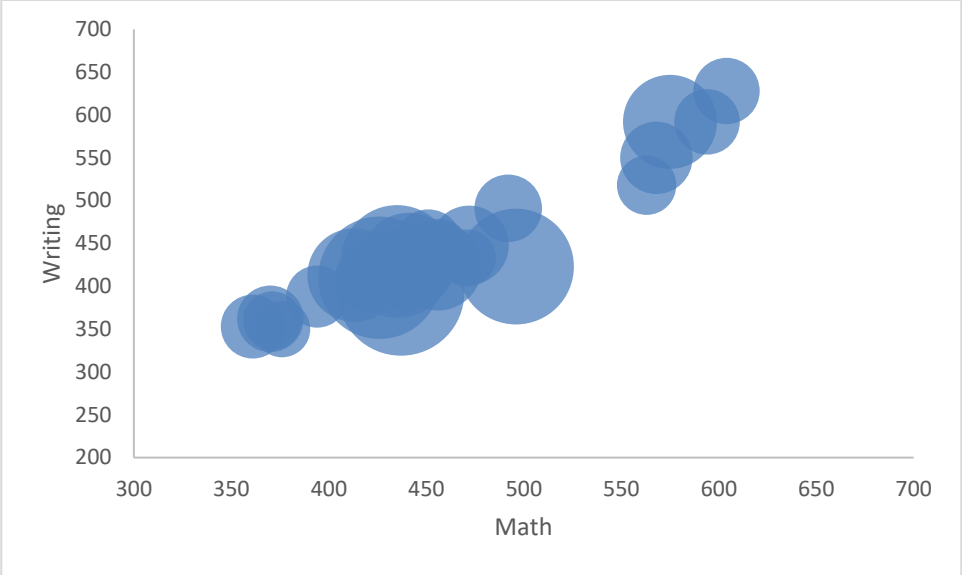
Both countries have a clear and consistent upward trend, but China's begins to stall slightly around 2000. India has slightly higher population growth over the past 40 years.

The R Code

```
# Import the data into a data frame (table) and label it myData.
#obtain blue line for India
plot(myData$India~ myData$Year, col="blue", type="l", ylab="Population (in millions)", xlab="Date")
#Incorporate red line for China
lines(myData$China~myData$Year, col = "red", type="l")
#Incorporate legend
legend("topleft", legend=c("India","China"), col=c("blue","red"), lty=1)
```

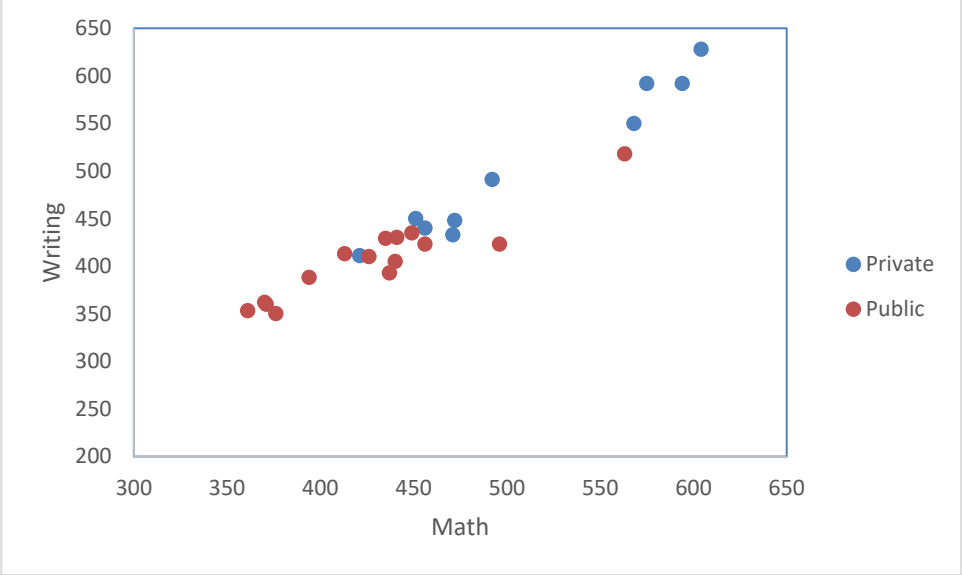

39.

a.



The bubble plot shows a positive linear relationship between the math and writing scores. There does not seem to be a relationship between the math score and the size of the school.

b.



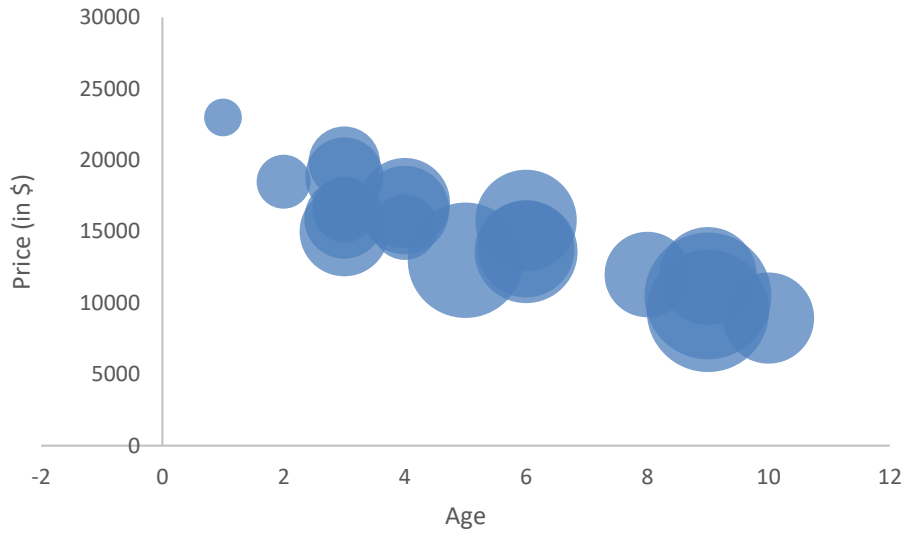
The scatterplot shows a positive linear relationship between the math and writing scores, and this relationship holds true for both public and private schools. The scatterplot suggests that private high schools tend to perform better than public high schools in terms of both math and writing scores.

The R Code

```
# Import the data into a data frame (table) and label it myData.
#obtaining bubble plot
plot(myData$Writing ~ myData$Math, type="n")
symbols(myData$Writing ~ myData$Math, circles=myData$Test_Takers, inches=0.5,
  bg="blue", main="A bubble plot of Writing, Math, and Test_Takers", xlab = "Math
  Score", ylab ="Writing Score")
# obtaining scatterplot with categorical variable
plot(myData$Writing ~ myData$Math,pch=16, col=ifelse(myData$Type==”Private”, 20
  , 26))
legend(“bottomright”, legend=c(“Private”,”Public”), pch=16, col=c(20,26))
```

40.

a.

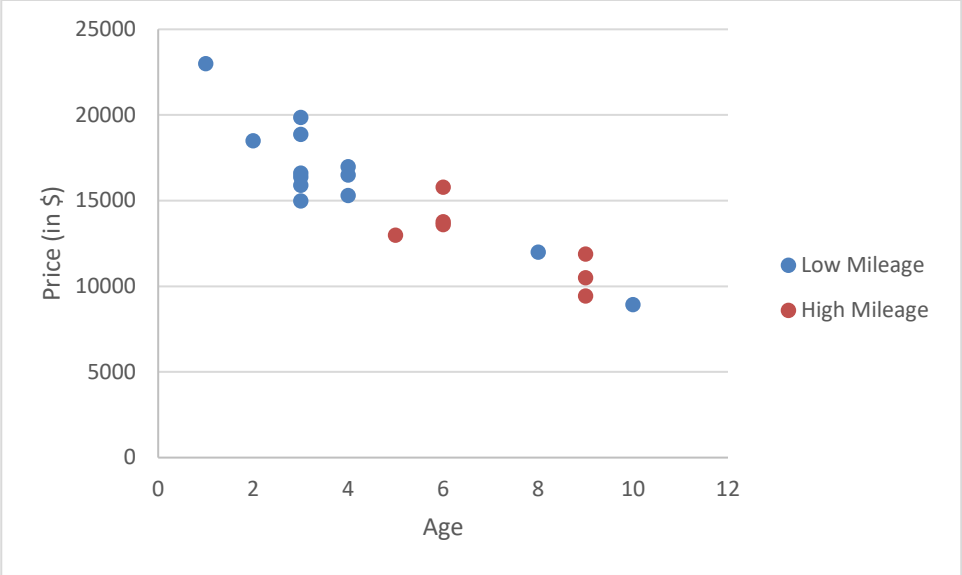


Price seems to be negatively correlated with Age and Mileage. Age and Mileage seem to be positively correlated.

b.

Seven cars have mileage greater than 50,000.

c.



The negative relationship between age and price is consistent for cars of both mileage categories.

The R Code

```
# Import the data into a data frame (table) and label it myData.
#obtaining bubble plot
plot(myData$Price ~ myData$Age, type="n")
symbols(myData$Price ~ myData$Age, circles=myData$Mileage, inches=0.5,
bg="blue", main="A bubble plot of Price, Age, and Mileage", xlab = "Age", ylab
="Price (in $)")
#converting Mileage to category variable
myData$Mileage_Category <-
ifelse(myData$Mileage<50000,"Low_Mileage","High_Mileage")
myData
table(myData$Mileage_Category)
# obtaining scatterplot with categorical variable
plot(myData$Price ~ myData$Age, pch=16,
col=ifelse(myData$Mileage_Category=="Low_Mileage", 20, 26), xlab="Age",
ylab="Price (in $)")
legend("bottomright", legend=c("Low_Mileage","HIgh_Mileage"), pch=16,
col=c(20,26))
```

41.

a.

	Size					
Color	XS	S	M	L	XL	Total
Blue	80	113	111	88	107	499
Green	95	76	122	98	97	488

Purple	91	119	63	91	81	445
Red	102	108	109	88	92	499
Yellow	114	112	107	85	102	520
Total	**482**	**528**	**512**	**450**	**479**	**2451**

109 M red t-shirts were sold; 81 XL purple t-shirts were sold

b. Excel heatmap using red-yellow-green in ascending order of quantity sold

	Size					
Color	**XS**	**S**	**M**	**L**	**XL**	**Total**
Blue	80	113	111	88	107	499
Green	95	76	122	98	97	488
Purple	91	119	63	91	81	445
Red	102	108	109	88	92	499
Yellow	114	112	107	85	102	520
Total	**482**	**528**	**512**	**450**	**479**	**2451**

M Green and S Purple are the two most popular combinations. M Purple and S Green are the least popular combinations.

42.

a.

	Location				
Crime	HOTEL/MOTEL	RESIDENCE	SIDEWALK	STREET	Total
ARSON		3		2	5
ASSAULT	4	191	114	175	484
BATTERY	14	742	439	380	1575
BURGLARY	1	511		4	516
SEXUAL ASSAULT	5	16	4	4	29
CRIMINAL DAMAGE	9	419	20	692	1140
CRIMINAL TRESPASS	4	46	11	21	82
DECEPTIVE PRACTICE	10	207	26	55	298

GAMBLING			1		1
HOMICIDE				22	22
INTERFERENCE WITH PUBLIC OFFICER		3	25	51	79
INTIMIDATION		3	1	1	5
KIDNAPPING		4	5	2	11
LIQUOR LAW VIOLATION	1	1	3	4	9
MOTOR VEHICLE THEFT		8	4	817	829
NARCOTICS		51	823	643	1517
NON-CRIMINAL		1			1
OBSCENITY		1			1
OFFENSE INVOLVING CHILDREN		106	2	5	113
OTHER OFFENSE	6	491	45	173	715
PROSTITUTION	3		5	29	37
PUBLIC INDECENCY	1				1
PUBLIC PEACE VIOLATION	2	10	34	54	100
ROBBERY	1	22	409	276	708
SEX OFFENSE		10	2	3	15
STALKING		4	1	2	7
THEFT	38	545	169	1200	1952
WEAPONS VIOLATION	1	32	44	56	133
Total	100	3427	2187	4671	10385

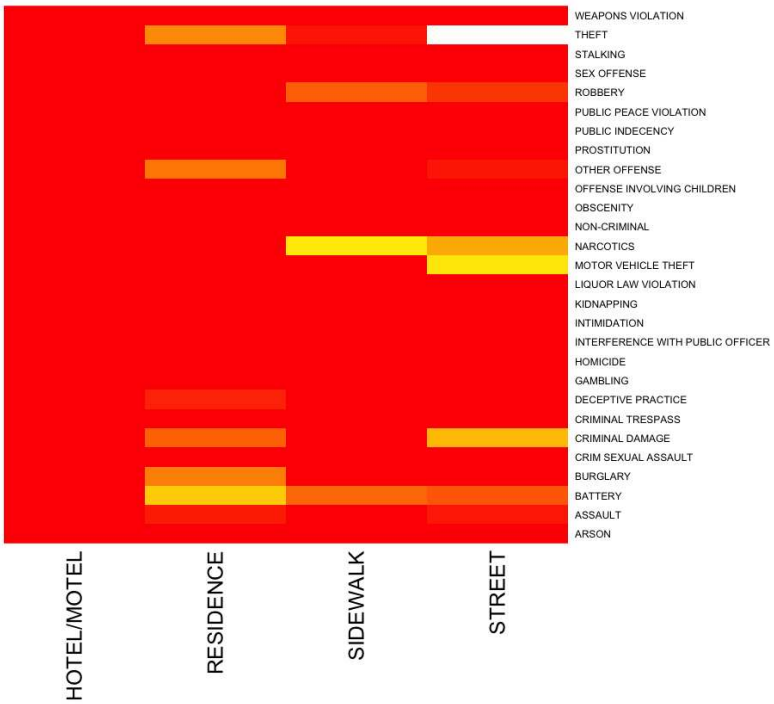
There are 511 cases of a burglary in a residence.

b.

Excel

Row Labels	HOTEL/MOTEL	RESIDENCE	SIDEWALK	STREET
ARSON		3		2
ASSAULT	4	191	114	175
BATTERY	14	742	439	380
BURGLARY	1	511		4
CRIM SEXUAL ASSAULT	5	16	4	4
CRIMINAL DAMAGE	9	419	20	692
CRIMINAL TRESPASS	4	46	11	21
DECEPTIVE PRACTICE	10	207	26	55
GAMBLING			1	
HOMICIDE				22
INTERFERENCE WITH PUBLIC OFFICER		3	25	51
INTIMIDATION		3	1	1
KIDNAPPING		4	5	2
LIQUOR LAW VIOLATION	1	1	3	4
MOTOR VEHICLE THEFT		8	4	817
NARCOTICS		51	823	643
NON-CRIMINAL		1		
OBSCENITY		1		
OFFENSE INVOLVING CHILDREN		106	2	5
OTHER OFFENSE	6	491	45	173
PROSTITUTION	3		5	29
PUBLIC INDECENCY	1			
PUBLIC PEACE VIOLATION	2	10	34	54
ROBBERY	1	22	409	276
SEX OFFENSE		10	2	3
STALKING		4	1	2
THEFT	38	545	169	1200
WEAPONS VIOLATION	1	32	44	56

R



Theft on the street is the most common crime, followed by narcotics on a sidewalk, then motor vehicle theft on the street.

The R Code

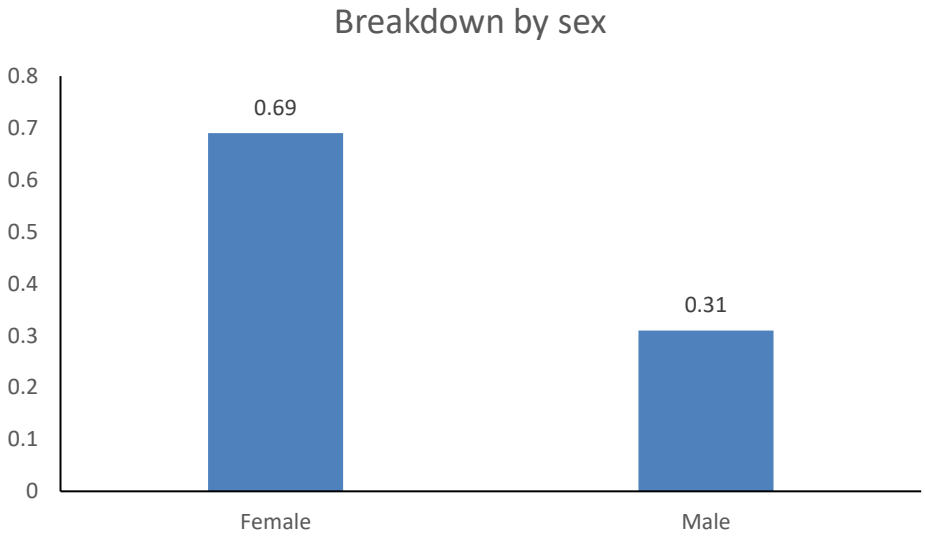
```
# Import the data into a data frame (table) and label it myData.
#constructing the contingency table
myTable <- table(myData$CrimeType, myData$Location)
myTable
#contructing heat map

myData.matrix <- as.matrix(myTable)
heatmap(myData.matrix, col = heat.colors(256), scale ="none",
      Rowv= NA, Colv = NA)
```

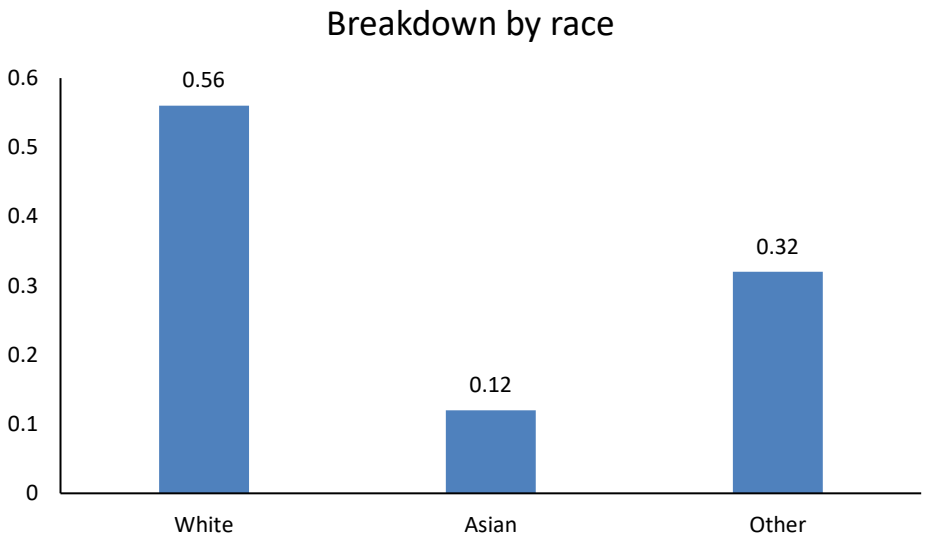
Possible Output for Suggested Case Studies

There are many ways to approach these case studies. We provide tabular and graphical guidance for one possible scenario. Instructors may also be interested in Connect multiple-choice exercises (algorithmic) that we have created using the **TechSales_Rep** data. These exercises span multiple chapters, including this one.

Report 4.1
First subset the **College_Admissions** data by applicants to the School of Arts & Letters. You should obtain 6,964 observations.

- Demographics of the applicant pool by sex and race:

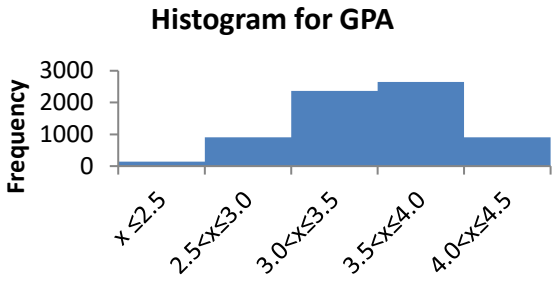


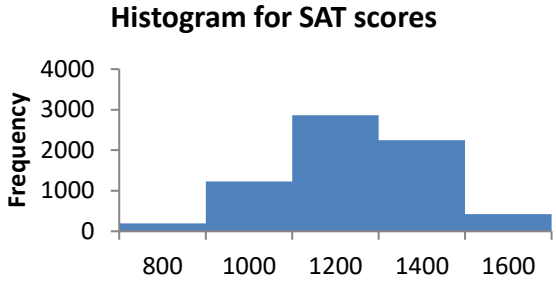


- Frequency distributions for GPA and SAT scores:

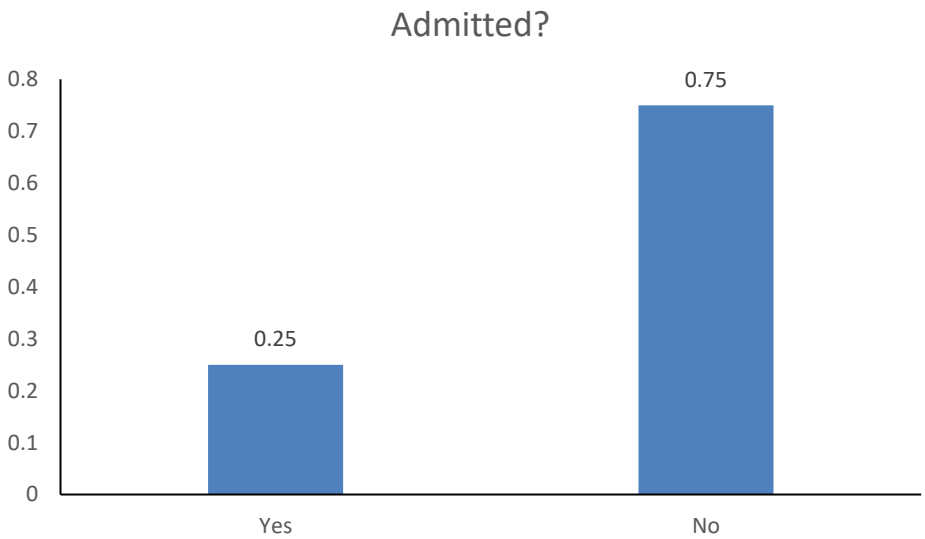
GPA	Frequency	Relative Frequency	SAT	Frequency	Relative Frequency
x ≤2.5	144	0.02	x ≤800	800	0.11
2.5<x≤3.0	902	0.13	800<x≤1000	1000	0.14
3.0<x≤3.5	2365	0.34	1000<x≤1200	1200	0.17
3.5<x≤4.0	2652	0.38	1200<x≤1400	1400	0.20
4.0<x≤4.5	901	0.13	1400<x≤1600	1600	0.23

- Histograms for GPA and SAT scores





- Breakdown of Acceptance Rate



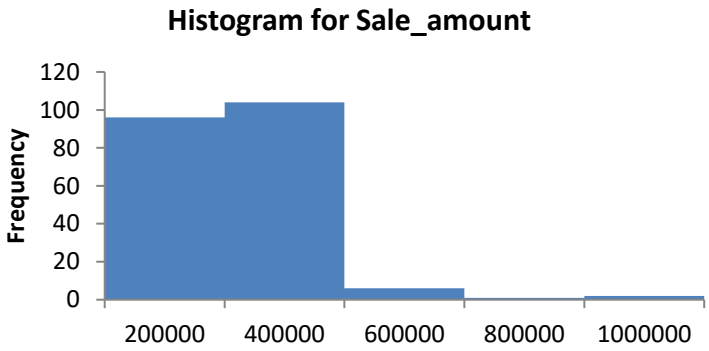
Report 4.2

First subset the **House_price** data by the college town of Ames, Iowa (Iowa State University). Then make sure that you only include houses with at least two bedrooms and where the sale price is less than $1,000,000. You should obtain 209 observations.

- Frequency distribution for Sale_amount:

Sale_amount	Frequency	Relative Frequency
x ≤ 200,000	96	0.46
200,000 < x ≤400,000	104	0.50
400,000 < x ≤600,000	6	0.03
600,000 < x ≤800,000	1	0.00
800,000 <x ≤ 1,000,000	2	0.01

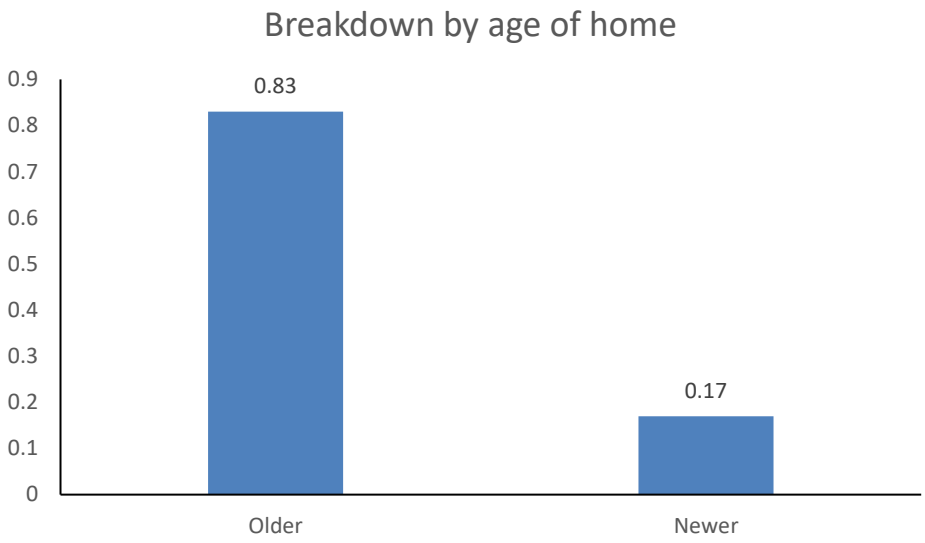
- Histogram for Sale_amount:



- Frequency distribution for the number of bedrooms:

Number of Bedrooms	Frequency	Relative Frequency
2	30	0.14
3	103	0.49
4	54	0.26
5+	22	0.11

- Breakdown by age of home (Older: built before 2000 versus Newer: built 2000+)

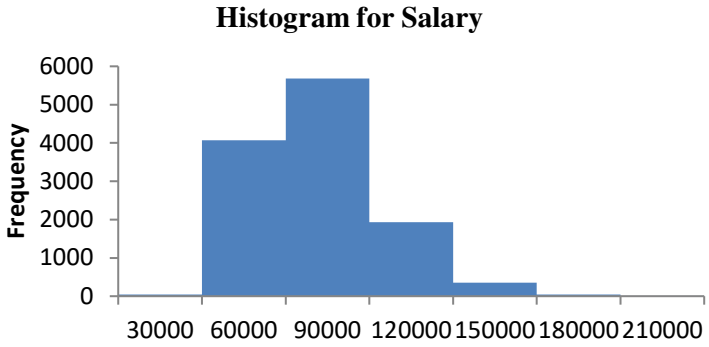


Report 4.3
First subset the **TechSales_Reps** data by the Software business. After doing this you should obtain 12,130 observations.

- Frequency distribution for Salary:

Salary	Frequency	Relative Frequency
$x \leq 30{,}000$	40	0.00
$30{,}000 < x \leq 60{,}000$	4066	0.34
$60{,}000 < x \leq 90{,}000$	5685	0.47
$90{,}000 < x \leq 120{,}000$	1928	0.16
$120{,}000 < x \leq 150{,}000$	356	0.03
$150{,}000 < x \leq 180{,}000$	47	0.00
$180{,}000 < x \leq 210{,}000$	8	0.00

- Histogram for Salary:

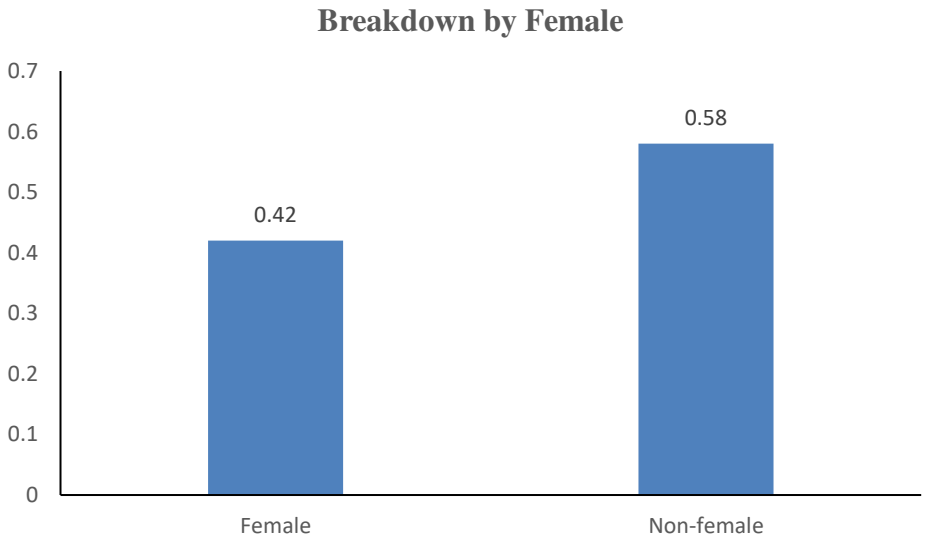


- Frequency distribution for NPS:

Salary	Frequency	Relative Frequency
$x \leq 2$	253	0.02
$2 < x \leq 4$	2702	0.22
$4 < x \leq 6$	3659	0.30
$6 < x \leq 8$	2984	0.25
$8 < x \leq 10$	2532	0.21

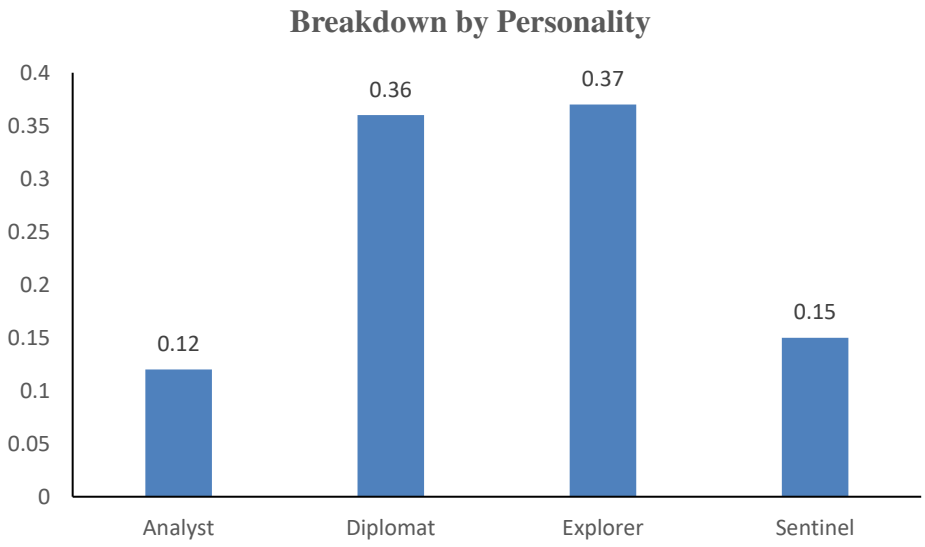
- Breakdown by Female variable:

	Frequency	Relative Frequency
Female	5087	0.42
Non-Female	7043	0.58



- Breakdown by Personality variable:

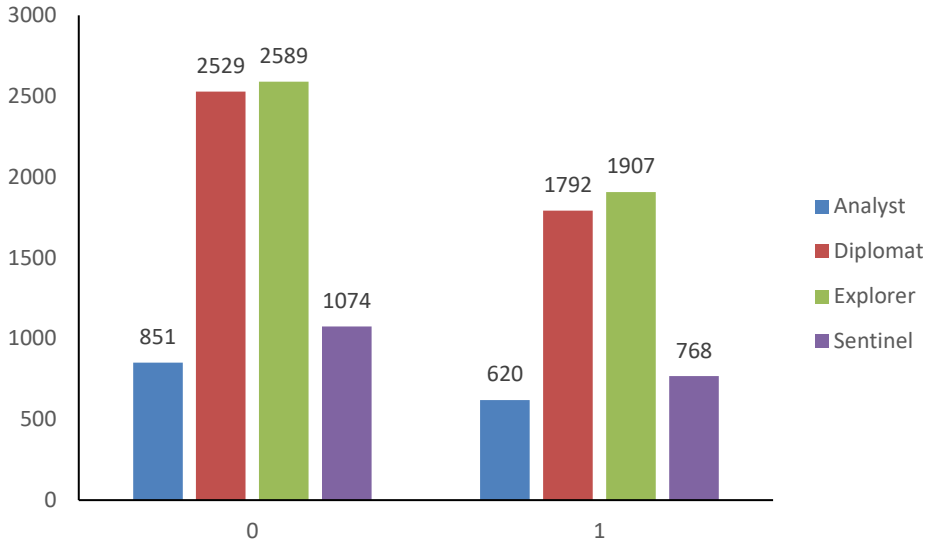
	Frequency	Relative Frequency
Analyst	1471	0.12
Diplomat	4321	0.36
Explorer	4496	0.37
Sentinel	1842	0.15



- A contingency table cross-classified by Female and Personality:

	Analyst	Diplomat	Explorer	Sentinel
Non-female (0)	851	2529	2589	1074
Female (1)	620	1792	1907	768

- Clustered Column Chart cross-classified by Female and Personality:

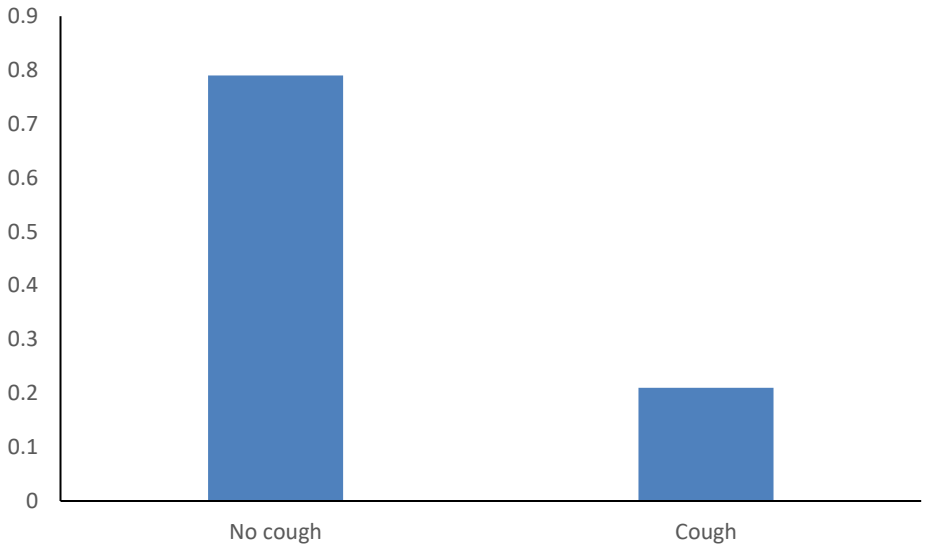


Report 4.4

First subset the **COVID_Testing** data by females who are over 60 years of age and tested positive. After doing this you should obtain 6,035 observations.

- Breakdown by Cough variable:

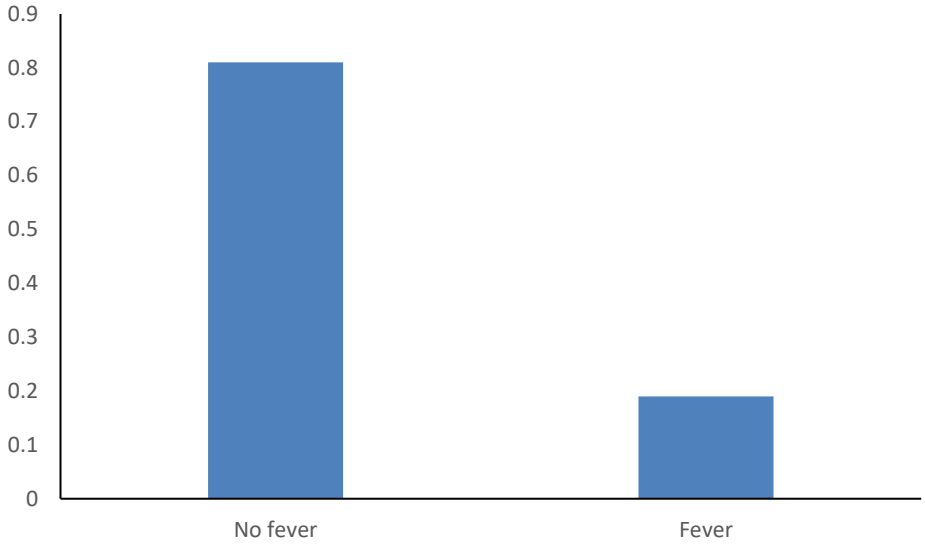
	Frequency	Relative Frequency
No cough	4786	0.79
Cough	1249	0.21



- Breakdown by Fever variable:

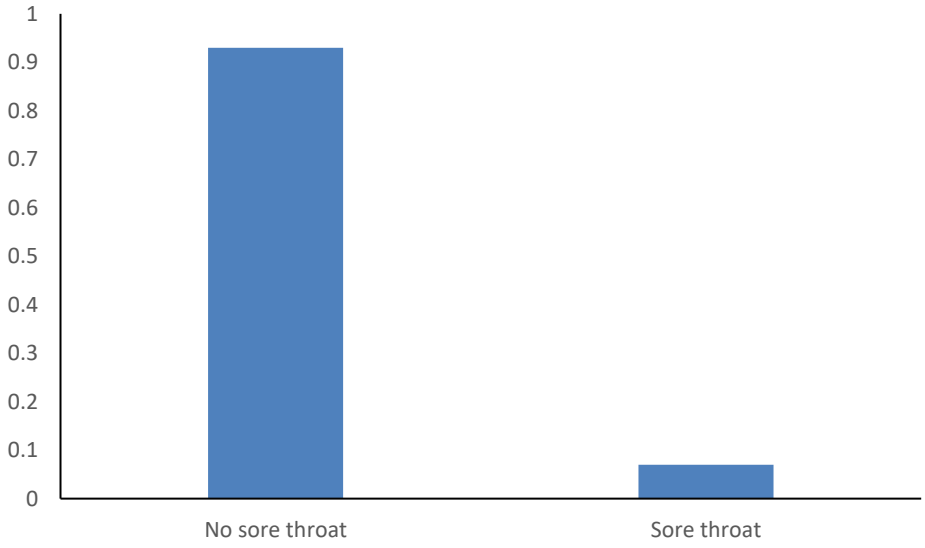
	Frequency	Relative Frequency

No fever	4883	0.81
Fever	1152	0.19



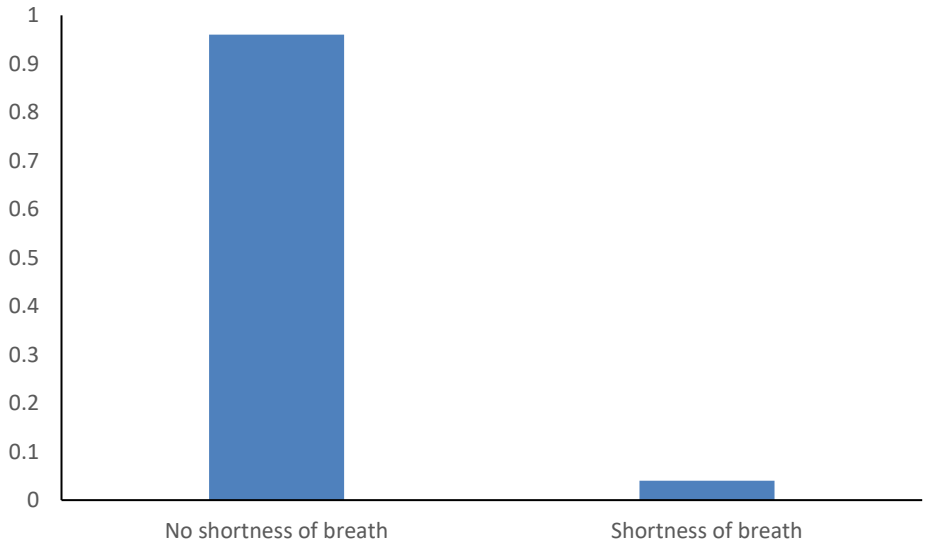
- Breakdown by Sore_throat variable:

	Frequency	Relative Frequency
No sore throat	5608	0.93
Sore throat	427	0.07



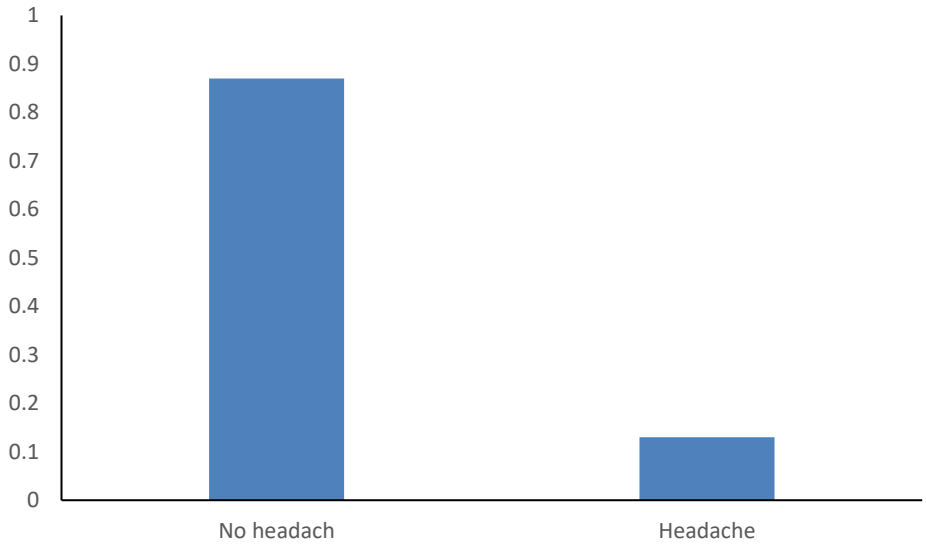
- Breakdown by Shortness_of_breath variable:

	Frequency	Relative Frequency
No shortness/breath	5816	0.96
Shortness of breath	219	0.04



- Breakdown by Headache variable:

	Frequency	Relative Frequency
No headache	5235	0.87
Headache	800	0.13



- Breakdown by Contact variable:

	Frequency	Relative Frequency
No contact	4031	0.67
Contact	2004	0.33