Midterm

2023-06-20

```
my_corrplot = function(x) {
    corrplot(x, method = "circle", type = "lower",diag = FALSE,
         tl.cex = 1,  # text label size
         cl.cex = 1,  # color legend text size
         number.cex = 1  # size of correlation coefficients
    )
}
```

Chapter 4 - applied

13. This question should be answered using the Weekly data set, which is part of the ISLR2 package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

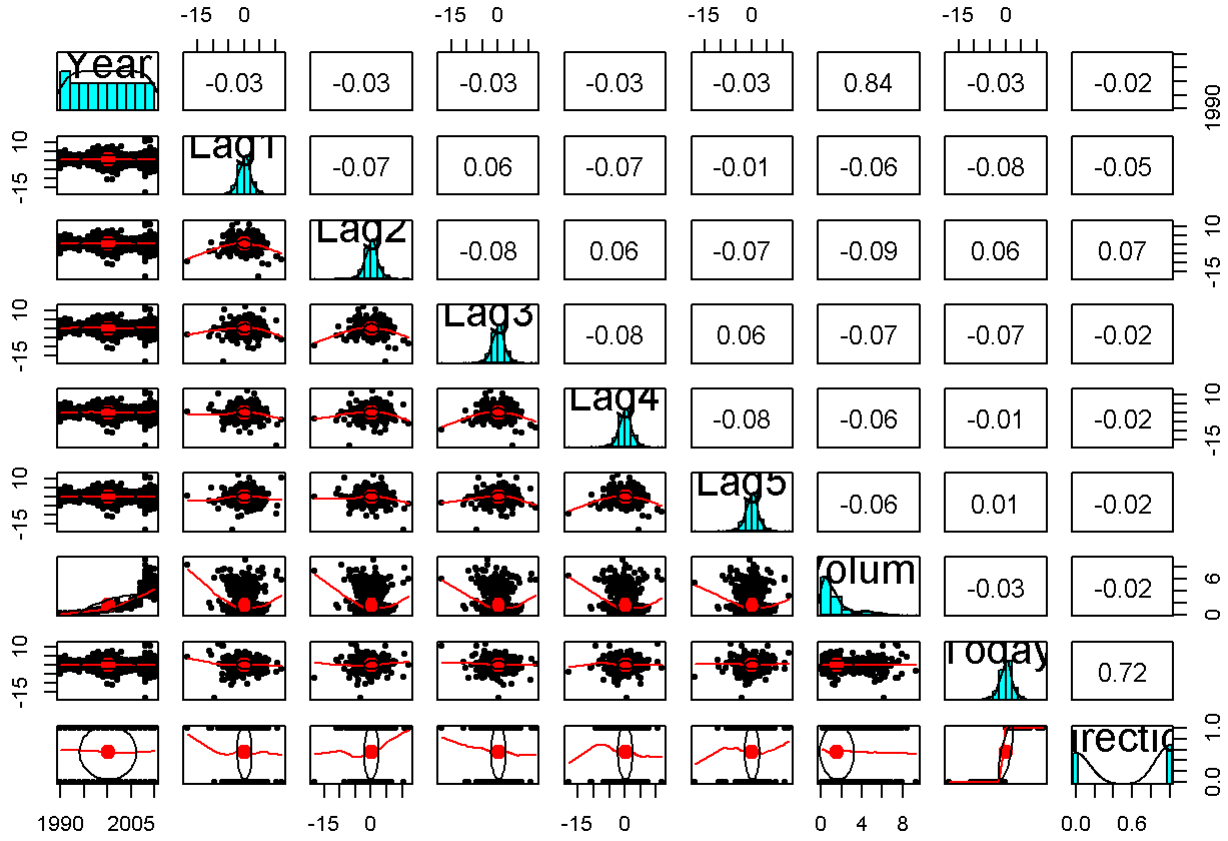
```
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

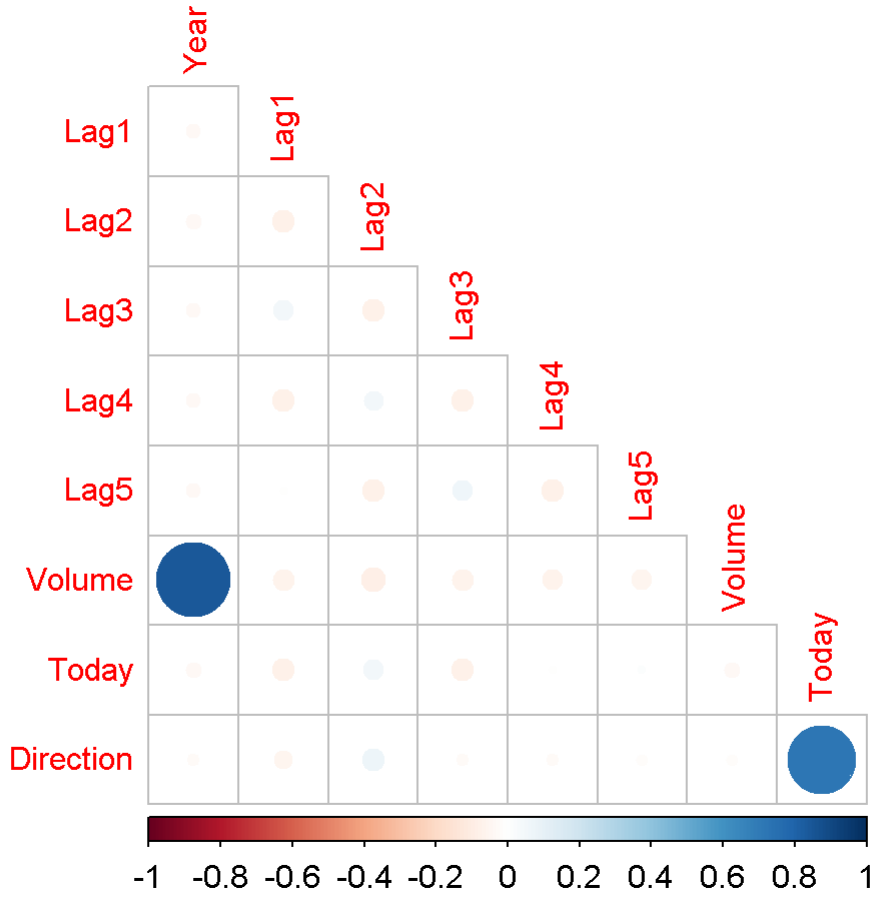
```
Weekly$Direction = ifelse(Weekly$Direction == "Up", 1, 0)
summary(Weekly)
```

```
##       Year           Lag1                Lag2                Lag3          
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4                Lag5              Volume            Today          
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
##    Direction 
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :1.0000  
##  Mean   :0.5556  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

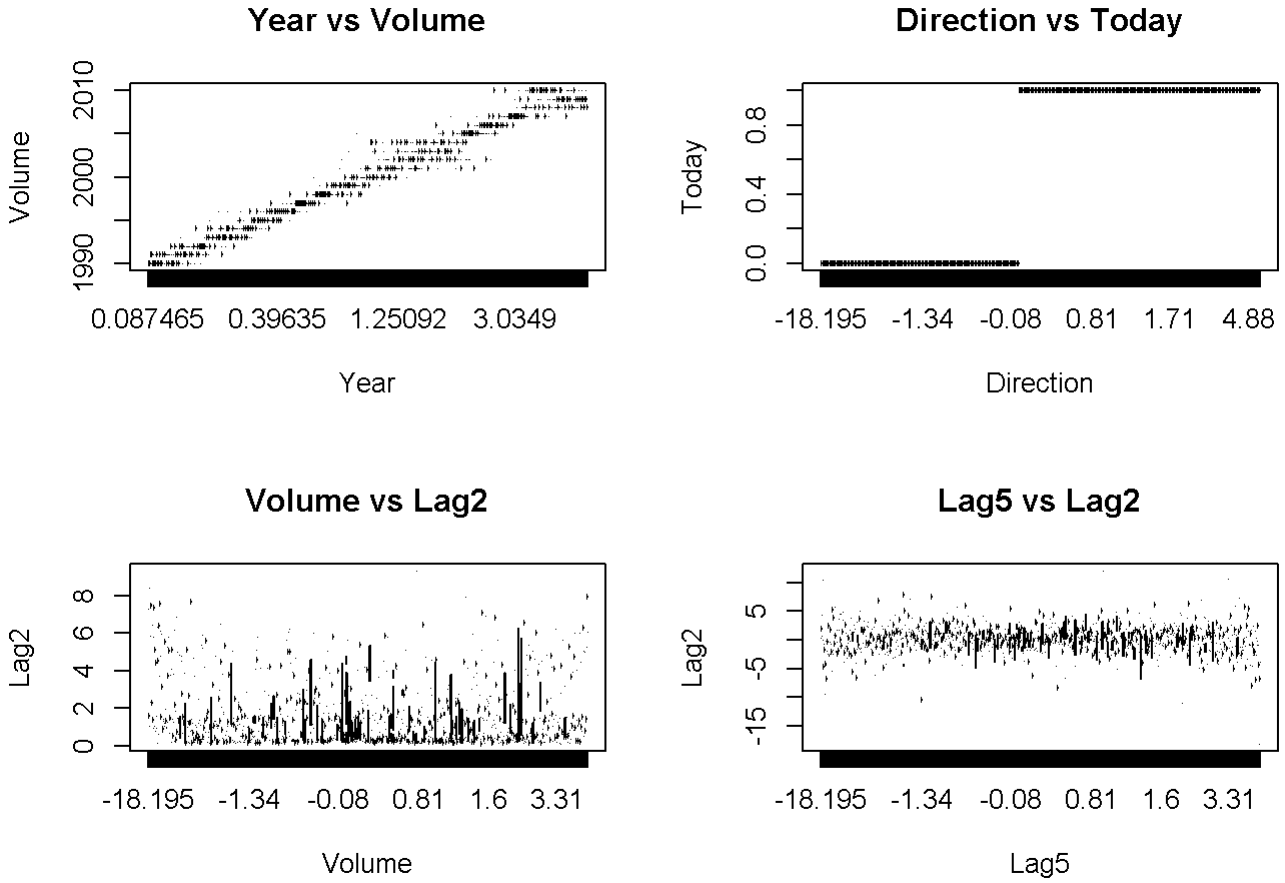
```
pairs.panels(Weekly, cex.labels = 2)
```



```
Weekly_cor = cor(Weekly)
my_corrplot(Weekly_cor)
```




```
par(mfrow = c(2, 2))
boxplot(Weekly$Year ~ Weekly$Volume,
        main = "Year vs Volume",
        xlab = "Year",
        ylab = "Volume")
boxplot(Weekly$Direction ~ Weekly$Today,
        main = "Direction vs Today",
        xlab = "Direction",
        ylab = "Today")
boxplot(Weekly$Volume ~ Weekly$Lag2,
        main = "Volume vs Lag2",
        xlab = "Volume",
        ylab = "Lag2")
boxplot(Weekly$Lag5 ~ Weekly$Lag2,
        main = "Lag5 vs Lag2",
        xlab = "Lag5",
        ylab = "Lag2")
```



As this is stock market data, most values have weak correlations; however, there are two strong correlations: Volume and Year, and Direction and Today. Volume of shares traded per day has, on average, risen in each subsequent year. It has a strong correlation of 0.84. Direction and Today are also strongly correlated at 0.72. Upon inspection, it occurs that Direction and Today are highly overlapped. Direction is a binary variable measuring whether the market has increased or decreased over the preceding week, while today measures the return over the previous week. While Lag1 through Lag6 and volume have far weaker correlations, It would be worthwhile to investigate their performance as combined variables.

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
log_model = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = 'bi
nomial')
summary(log_model)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = Weekly)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Overall, this model barely outperforms a model with no predictors shown by the small difference between the null deviance and residual deviance. However, there are a few features that display statistical significance. The intercept and Lag2 have p-values of .0019 and .0296 respectively. Since these are less than 0.05, it indicates they are significant. All other feature failed to meet the 0.05 threshold.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
predicted_direction = ifelse(predict(log_model, Weekly, type = "response") > 0.5, "Up", "Down")
confusion_matrix = table(Actual = Weekly$Direction, Predicted = predicted_direction)
correct_predictions <- sum(diag(confusion_matrix))
total_predictions <- sum(confusion_matrix)
fraction_correct <- correct_predictions / total_predictions
confusion_matrix
```

```
##       Predicted
## Actual Down  Up
##      0   54 430
##      1   48 557
```

The confusion matrix was correct with an approximate probability of “, fraction_correct,”This confusion matrix demonstrates high recall, but low specificity. That is, it accurately predicts Up about 90% of the time, but it only correctly predicts down about 10% of the time. Overall, the precision of the model of 56%, while unsuitable for

many applications, could be useful in a high volume stock market situation."

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train_data = subset(Weekly, Year <= 2008)
train_data = subset(train_data, Year >= 1990)
X = glm(Direction ~ Lag2, data = train_data, family = "binomial")
summary(X)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = "binomial", data = train_data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```
test_data = subset(Weekly, Year >= 2009)
test_data = subset(test_data, Year <= 2010)
prediction = ifelse(predict(X, test_data, type = "response") > 0.5, "Up", "Down")
confusion_matrix_test = table(Actual = test_data$Direction, Predicted = prediction)
confusion_matrix_test
```

```
##       Predicted
## Actual Down Up
##      0    9 34
##      1    5 56
```

```
correct_predictions_test = sum(diag(confusion_matrix_test))
totals = sum(confusion_matrix_test)
fraction_correct = (correct_predictions_test / totals)
confusion_matrix_test
```

```
##       Predicted
## Actual Down Up
##      0    9 34
##      1    5 56
```

```
cat("The percentage of correct predictions for logistic regression is ", fraction_correct * 100)
```

```
## The percentage of correct predictions for logistic regression is  62.5
```

(e) Repeat (d) using LDA.

```
lda_model = lda(Direction ~ Lag2, data = train_data)
print(lda_model)
```

```
## Call:
## lda(Direction ~ Lag2, data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.4477157 0.5522843
##
## Group means:
##          Lag2
## 0 -0.03568254
## 1  0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
lda_predicton = predict(lda_model, test_data)$class
lda_confusion = table(Actual = test_data$Direction, Predicted = lda_predicton)
lda_confusion
```

```
##       Predicted
## Actual  0  1
##      0  9 34
##      1  5 56
```

```
lda_correct_preds = sum(diag(lda_confusion))
lda_totals = sum(lda_confusion)
lda_percentage = (lda_correct_preds / lda_totals) * 100
cat("The percentage of correct predictions for lda is ", lda_percentage)
```

```
## The percentage of correct predictions for lda is  62.5
```

(f) Repeat (d) using QDA.

```
qda_model = qda(Direction ~ Lag2, data = train_data)
print(qda_model)
```

```
## Call:
## qda(Direction ~ Lag2, data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.4477157 0.5522843
##
## Group means:
##          Lag2
## 0 -0.03568254
## 1  0.26036581
```

```
qda_predicton = predict(qda_model, test_data)$class
qda_confusion = table(Actual = test_data$Direction, Predicted = lda_predicton)
qda_confusion
```

```
##       Predicted
## Actual  0  1
##      0  9 34
##      1  5 56
```

```
qda_correct_preds = sum(diag(qda_confusion))
qda_totals = sum(qda_confusion)
qda_percentage = (qda_correct_preds / qda_totals) * 100
cat("The percentage of correct predictions for qda is ", qda_percentage)
```

```
## The percentage of correct predictions for qda is  62.5
```

(g) Repeat (d) using KNN with K = 1.

```
X_train = as.matrix(train_data[ , "Lag2", drop = FALSE])
y_train = train_data$Direction
X_test = as.matrix(test_data[ , "Lag2", drop = FALSE])
knn_pred = knn(scale(X_train), scale(X_test), y_train, k=1)
print(knn_pred)
```

```
##   [1] 1 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 0
##  [38] 1 1 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  [75] 0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1
## Levels: 0 1
```

```
knn_confusion = table(Actual = test_data$Direction, Predicted = knn_pred)
knn_confusion
```

```
##       Predicted
## Actual  0  1
##      0 14 29
##      1 26 35
```

```
knn_correct_preds = sum(diag(knn_confusion))
knn_totals = sum(knn_confusion)
knn_percentage = (knn_correct_preds/knn_totals) * 100
cat("The percentage of correct predictions for knn with k = 1 is ", knn_percentage)
```

```
## The percentage of correct predictions for knn with k = 1 is  47.11538
```

(h) Repeat (d) using naive Bayes.

```
naive_Bayes_model = naiveBayes(Direction ~ Lag2, data = train_data)
print(naive_Bayes_model)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.4477157 0.5522843
##
## Conditional probabilities:
##    Lag2
## Y          [,1]     [,2]
##   0 -0.03568254 2.199504
##   1  0.26036581 2.317485
```

```
naive_bayes_pred = predict(naive_Bayes_model, newdata = test_data)
nB_confusion = table(Actual = test_data$Direction, Predicted = naive_bayes_pred)
nB_confusion
```

```
##       Predicted
## Actual  0  1
##      0  0 43
##      1  0 61
```

```
nB_correct_preds = sum(diag(nB_confusion))
nB_totals = sum(nB_confusion)
nB_percentage = (nB_correct_preds/nB_totals) * 100
cat("The percentage of correct predictions for naive Bayes is ", nB_percentage)
```

```
## The percentage of correct predictions for naive Bayes is  58.65385
```

(i) Which of these methods appears to provide the best results on this data?

Linear Regression, LDA, and QDA all performed about equally as well by fraction of correct predictions. We could try other comparisons like AuC-ROC, F1, etc.; however, LDA would be fitted best to data with a normal distibution for each class, and QDA assumes a normal distribution regardless of class. Absent other data, I would select logistic regression for its simplicity and interpretablity.

Chapter 5 - Applied

5. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that uses income and balance to predict default.

```
Default_log_model = glm(default ~ income + balance, data = Default, family = "binomial")
summary(Default_log_model)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

```
set.seed(42)
Default$default = ifelse(Default$default == "Yes" | Default$default == 1, 1, 0)
train_index = sample(1:nrow(Default), nrow(Default)/2)
X = Default[train_index, ]
y = Default[-train_index, ]
```

ii. Fit a multiple logistic regression model using only the training observations.

```
Default_seed42_model = glm(default ~ income + balance, data = X, family = 'binomial')
Default_seed42_preds = predict(Default_seed42_model, newdata = y, type = 'response')
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

```
Default_seed42_classes = ifelse(Default_seed42_preds > 0.5, 1, 0)
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
Default_seed42_error = mean(Default_seed42_classes != y$default)
cat("The estimated error of a logistic model using the validation set approach and a seed of 42 is
", Default_seed42_error)
```

```
## The estimated error of a logistic model using the validation set approach and a seed of 42 is
0.026
```

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
for (this_seed in c(35, 40, 45)) {
  set.seed(this_seed)
  train_index = sample(1:nrow(Default), nrow(Default)/2)
  X = Default[train_index, ]
  y = Default[-train_index, ]
  Default_seed42_model = glm(default ~ income + balance, data = X, family = 'binomial')
  Default_seed42_preds = predict(Default_seed42_model, newdata = y, type = 'response')
  Default_seed42_classes = ifelse(Default_seed42_preds > 0.5, 1, 0)
  Default_seed42_error = mean(Default_seed42_classes != y$default)
  cat("The estimated error of a logistic model using the validation set approach and a seed of ",
this_seed, " is ", Default_seed42_error, "\n")
}
```

```
## The estimated error of a logistic model using the validation set approach and a seed of  35  is
0.0268
## The estimated error of a logistic model using the validation set approach and a seed of  40  is
0.026
## The estimated error of a logistic model using the validation set approach and a seed of  45  is
0.0256
```

As we varied the seed between 35, 40, 42, and 45, we saw our estimated error vary between 0.0256 and 0.0268. This highlights the variability of the model based on how it is divided into test and validation sets. While for this particular model the variability is rather small, it still reinforces the necessity of techniques like cross-validation for making better estimates of test error.

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using

the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```
set.seed(42)
Default$student = ifelse(Default$student == "Yes" | Default$student == 1, 1, 0)
train_index = sample(1:nrow(Default), nrow(Default)/2)
X = Default[train_index, ]
y = Default[-train_index, ]
Default_seed42_model = glm(default ~ income + balance + student, data = X, family = 'binomial')
Default_seed42_preds = predict(Default_seed42_model, newdata = y, type = 'response')
Default_seed42_classes = ifelse(Default_seed42_preds > 0.5, 1, 0)
Default_seed42_error = mean(Default_seed42_classes != y$default)
cat("The estimated error of a logistic model including a student dummy variable and a seed of 42 i
s ", Default_seed42_error, "\n")
```

```
## The estimated error of a logistic model including a student dummy variable and a seed of 42 is
0.0258
```

We do see a change when comparing the previous model with this one, 0.026 vs 0.0256 respectively. This difference is small, but could be significant when considering millions of dollars of investment across a large population. However, to determine if the inclusion of the dummy variable improves the model's error, we should perform other methods such as cross-validation to see if this improvement is consistent.

6. We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways:

1. using the bootstrap, and

2. using the standard formula for computing the standard errors in the glm() function.

Do not forget to set a random seed before beginning

your analysis.

(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
Default$default = ifelse(Default$default == "Yes" | Default$default == 1, 1, 0)
Default_model = glm(default ~ income + balance, data = Default, family = 'binomial')
summary(Default_model)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
boot.fn = function(data, index) {
  model = glm(default ~ income + balance, data = data[index, ], family = 'binomial')
  return(coef(model))
}
```

(c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients

for income and balance.

```
set.seed(42)
boot_coef = boot(Default, boot.fn, R=1000)
boot_coef
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01 -2.292405e-02 4.435269e-01
## t2*  2.080898e-05  2.737444e-08 5.073444e-06
## t3*  5.647103e-03  1.176249e-05 2.299133e-04
```

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

The estimated standard errors obtained by the two methods are fairly close. For glm() the standard error of the income and balance coefficients are 4.98e-06 and 2.27e-04, while for the boot() those standard errors are 5.07e-06 and 2.30e-04. The standard error of glm() assumes a binomial distribution, while the bootstrap doesn't make this assumption. However, bootstrap is significantly more computationally expensive. The preceding code takes as long to run as the rest of this document combined. This may preclude its use for larger data sets and more complex models.

9. We will now consider the Boston housing data set, from the ISLR2 library.

(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate ˆμ.

```
mu_hat = mean(Boston$medv)
cat("Our estimate for the population mean of medv, ˆμ, is ", mu_hat)
```

```
## Our estimate for the population mean of medv, ˆμ, is  22.53281
```

(b) Provide an estimate of the standard error of ˆμ. Interpret this result.

Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

```
se_mu_hat = sd(Boston$medv)/sqrt(length(Boston$medv))
cat("Our estimate of the standard error of ˆμ is ", se_mu_hat)
```

```
## Our estimate of the standard error of ˆμ is  0.4088611
```

In this case, with a standard error of 0.4088611, it implies that the estimate of the population mean based on the sample has a moderate amount of variability. because of the more moderate amount of variability, this means that the sample mean is expected to deviate by approximately 0.4088611 from the true population mean.

(c) Now estimate the standard error of ˆμ using the bootstrap. How does this compare to your answer from (b)?

```
boot.fn = function(data, index) mean(data[index])
boot_se = boot(Boston$medv, boot.fn, R=1000)
boot_se
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 -0.002375494   0.4185115
```

When we run the bootstrap formula, we receive a standard error of 0.4185115. This tells us a major thing. It tells us that the previous model in question b is going to be more reliable in estimating the population mean because its a smaller number. Now its not that much smaller with a difference of only about .01, but again, bootstrap will be less reliable with that larger standard error value. The smaller the standard error, the more accurate it will be when it comes to predicting the true population mean.

(d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).

Hint: You can approximate a 95 % confidence interval using the formula [ˆμ − 2SE(ˆμ), μˆ + 2SE(ˆμ)].

```
boot.ci(boot_se, type='bca')
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_se, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   (21.74, 23.36 )
## Calculations and Intervals on Original Scale
```

```
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

From the sets of ranges that we get using the bootstrap and the t test, we can see a minor difference in ranges. Comparing the two intervals, we can see they are very similar. Both will give us an estimate of the population mean and will have a similar range. Now the confidence level that we get from the t test is only slightly more accurate just because the values are minutely smaller. Overall they are both consistent in determining the confidence level but we would want to follow the t test model to get a more accurate prediction.

(e) Based on this data set, provide an estimate, ˆμmed, for the median value of medv in the population

```
mu_med_hat = median(Boston$medv)
cat("The median value of ˆμmed in the population is, ", mu_med_hat)
```

```
## The median value of ˆμmed in the population is,  21.2
```

(f) We now would like to estimate the standard error of ˆμmed. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error

of the median using the bootstrap. Comment on your findings.

```
boot.fn.median = function(data, index) median(data[index])
boot_med = boot(Boston$medv, boot.fn.median, R=1000)
boot_med
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn.median, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0137   0.3906596
```

Median Estiame shows us that its a value of 21.2. The Bias is a value of -.0137 because it is a negative value, it is estimated that the statistical median is slightly lower than the true population median. In other words, its a little bit understimated.For the standard error, its shows a value of .3906596. The size of this standard error is similar to the standard error of the mean. In this case, this is once again a moderate size of a standard error which might tell us that this might not be the most accurate way of predicting the true median.

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston census tracts. Call this quantity ˆμ0.1.

(You can use the quantile() function.)

```
mu_10pct_hat = quantile(Boston$medv, 0.1)
cat("Our estimate for ˆμ0.1 is, ", mu_10pct_hat)
```

```
## Our estimate for ˆμ0.1 is,  12.75
```

(h) Use the bootstrap to estimate the standard error of ˆμ0.1. Comment on your findings.

```
boot.fn.10pct = function(data, index) quantile(data[index], 0.1)
boot_10pct = boot(Boston$medv, boot.fn.10pct, R=1000)
cat("Out bootsrapped estimate of ˆμ0.1 is, ")
```

```
## Out bootsrapped estimate of ˆμ0.1 is,
```

```
boot_10pct
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn.10pct, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75  0.0201   0.4950819
```

The estimated value for the statistic of interest (10th percentile) in the original data is approximately 12.75. This means that 10% of the data points fall below this value.Additionally, the analysis indicates a small positive bias of 0.0201. This implies that, on average, the estimated 10th percentile is slightly higher than the true population value.The standard error of the estimate is approximately 0.4950819. This value reflects the variability or uncertainty associated with the estimation of the 10th percentile. A larger standard error suggests a greater amount of variability in the estimate.