ISLR Chapter 4 Applied Solution

Abhirup Sen

03/06/2021

#10. APPLIED: The Weekly Dataset (Logistic, LDA, QDA, KNN)#

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(ISLR) # 'Weekly' data
library(caret) # train(), confusionMatrix()
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(MASS) # lda(), qda(), `Boston` data
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
select <- dplyr::select # MASS 'select' clashing with dplyr
library(class) # knn()
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

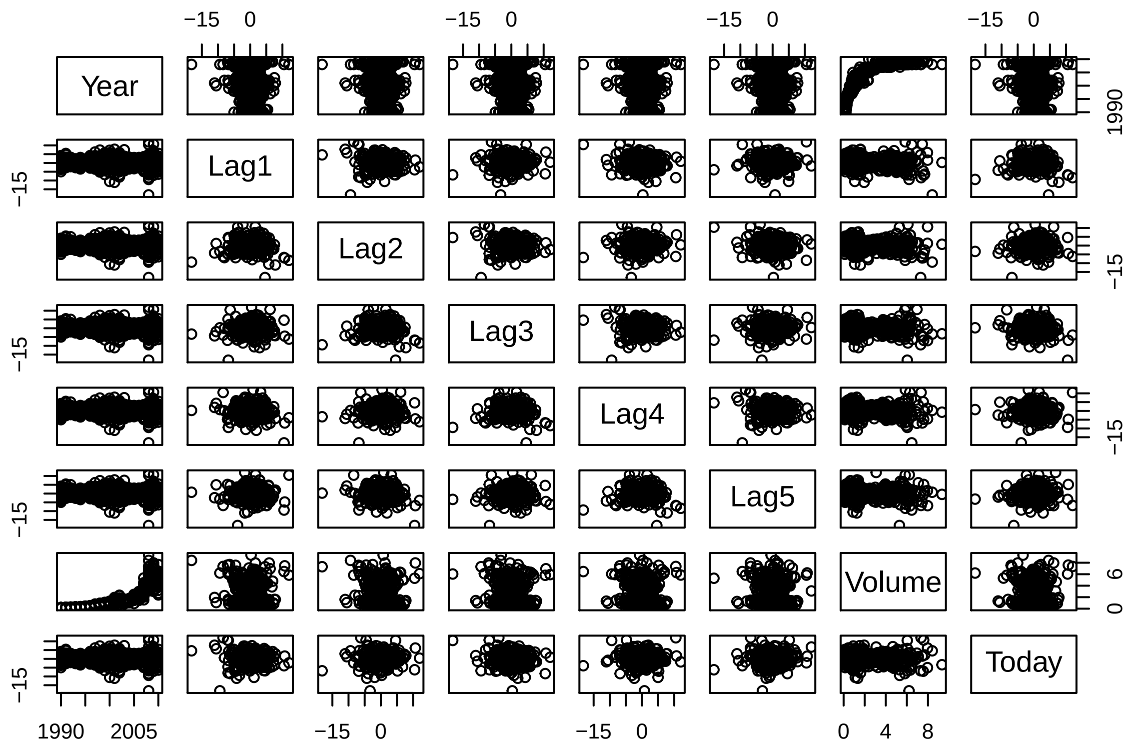
This question should be answered using the *Weekly* data set, which is part of the ISLR package. This data is similar in nature to the *Smarket* data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from tglimpse(Weekly)he beginning of 1990 to the end of 2010.

```r
glimpse(Weekly)
```

```
## Rows: 1,089
## Columns: 9
## $ Year      <dbl> 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, ~
## $ Lag1      <dbl> 0.816, -0.270, -2.576, 3.514, 0.712, 1.178, -1.372, 0.807, 0~
## $ Lag2      <dbl> 1.572, 0.816, -0.270, -2.576, 3.514, 0.712, 1.178, -1.372, 0~
## $ Lag3      <dbl> -3.936, 1.572, 0.816, -0.270, -2.576, 3.514, 0.712, 1.178, -~
## $ Lag4      <dbl> -0.229, -3.936, 1.572, 0.816, -0.270, -2.576, 3.514, 0.712, ~
## $ Lag5      <dbl> -3.484, -0.229, -3.936, 1.572, 0.816, -0.270, -2.576, 3.514,~
## $ Volume    <dbl> 0.1549760, 0.1485740, 0.1598375, 0.1616300, 0.1537280, 0.154~
## $ Today     <dbl> -0.270, -2.576, 3.514, 0.712, 1.178, -1.372, 0.807, 0.041, 1~
## $ Direction <fct> Down, Down, Up, Up, Up, Down, Up, Up, Up, Down, Down, Up, Up~
```

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

```r
pairs(Weekly[ ,-9])
```




```r
abs(cor(Weekly[ ,-9]))
```

```
##              Year        Lag1       Lag2       Lag3        Lag4        Lag5
## Year   1.00000000 0.032289274 0.03339001 0.03000649 0.031127923 0.030519101
## Lag1   0.03228927 1.000000000 0.07485305 0.05863568 0.071273876 0.008183096
## Lag2   0.03339001 0.074853051 1.00000000 0.07572091 0.058381535 0.072499482
## Lag3   0.03000649 0.058635682 0.07572091 1.00000000 0.075395865 0.060657175
## Lag4   0.03112792 0.071273876 0.05838153 0.07539587 1.000000000 0.075675027
## Lag5   0.03051910 0.008183096 0.07249948 0.06065717 0.075675027 1.000000000
## Volume 0.84194162 0.064951313 0.08551314 0.06928771 0.061074617 0.058517414
## Today  0.03245989 0.075031842 0.05916672 0.07124364 0.007825873 0.011012698
##            Volume       Today
## Year   0.84194162 0.032459894
## Lag1   0.06495131 0.075031842
## Lag2   0.08551314 0.059166717
## Lag3   0.06928771 0.071243639
## Lag4   0.06107462 0.007825873
## Lag5   0.05851741 0.011012698
## Volume 1.00000000 0.033077783
## Today  0.03307778 1.000000000
```

As we would expect with stock market data, there are no obvious strong relationships between the Lag variables. However, there do appear to be some interesting trends over time. We create the Week variable below, allowing for easier plotting of trends, since there is a chronology to the rows that is not shown fully through the Year variable.

```r
Weekly %>%
  filter(lead(Lag1) != Today) %>%
  nrow()
```

```
## [1] 0
```

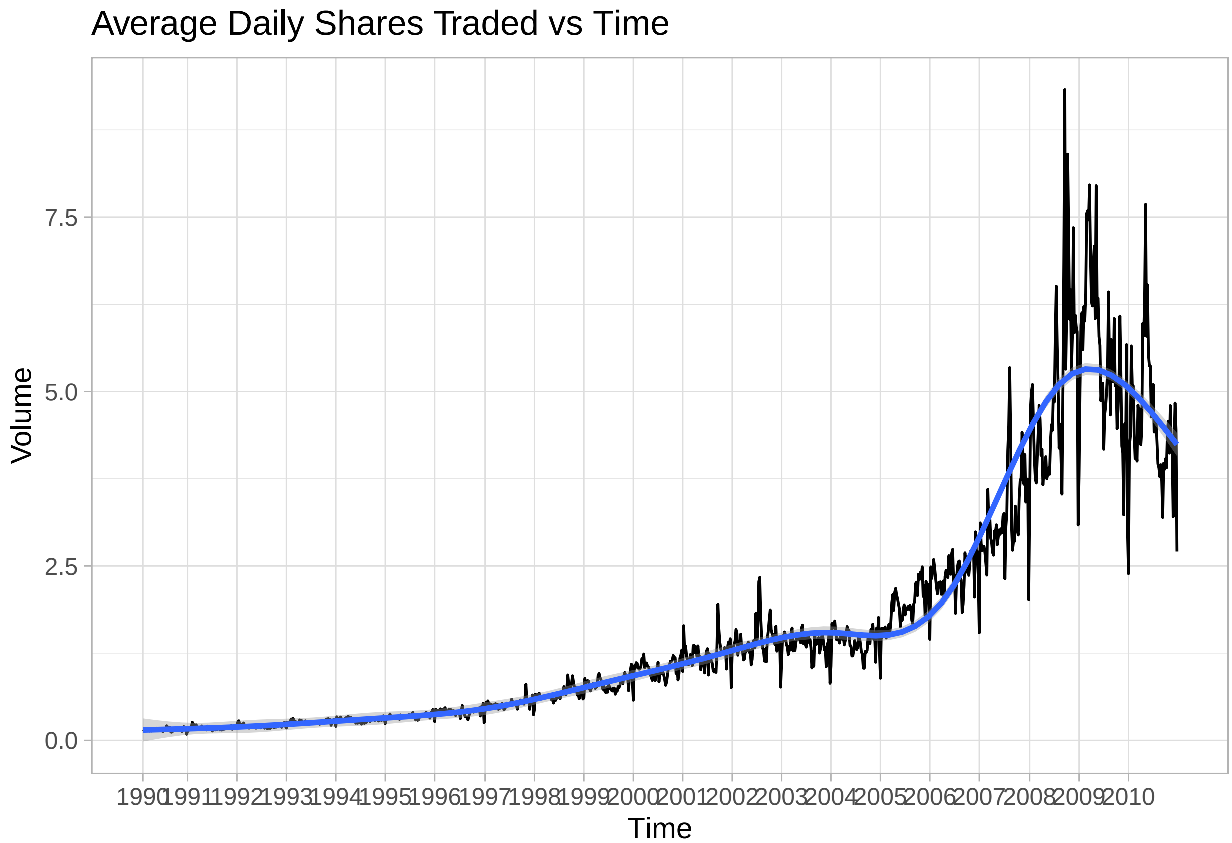
Since there are no rows out of order, the dataset appears to be correctly ordered in ascending weeks.

```r
Weekly$Week <- 1:nrow(Weekly)
```

Looking at Volume over time, there has been a significant increase in the volume of shares traded since the 90's. This appears to have peaked around 2009, starting to decrease in 2010. it would be interesting to see the S&P 500 stats since then.

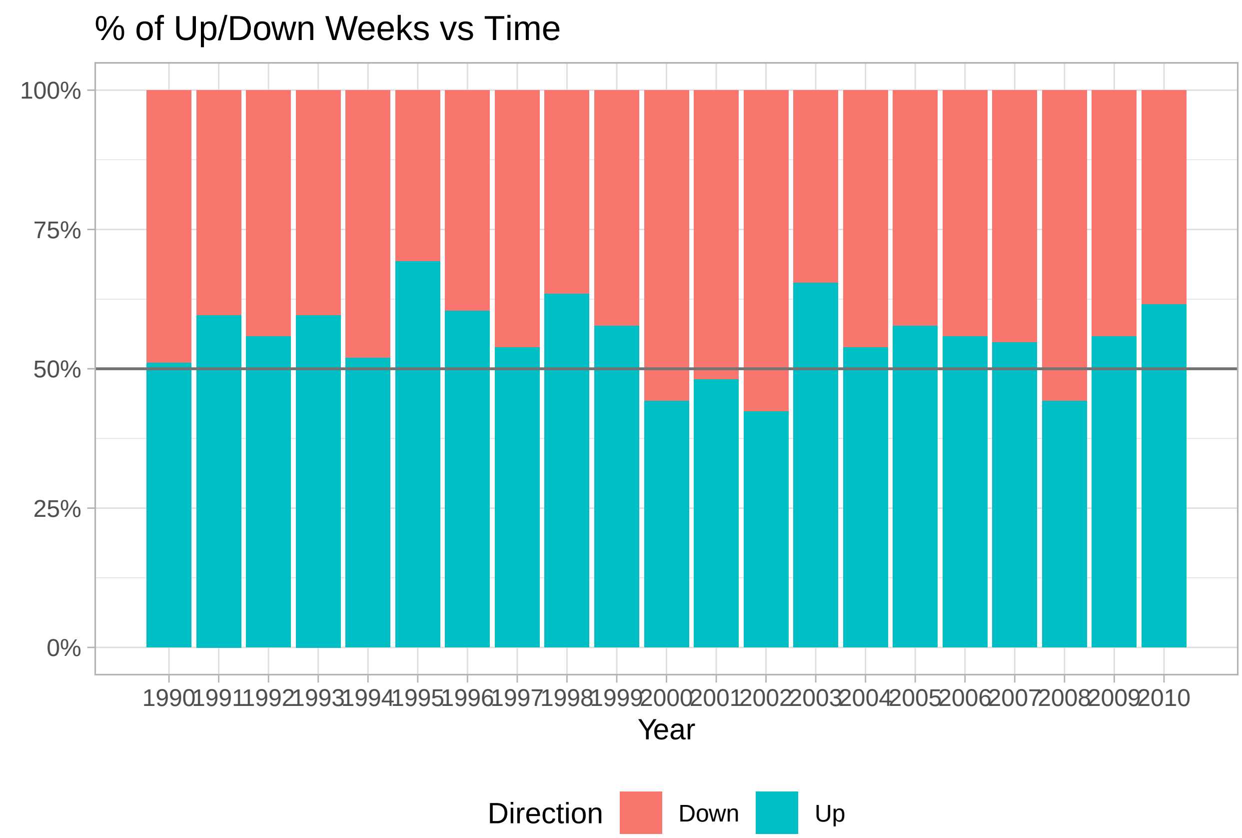
```r
year_breaks <- Weekly %>%
  group_by(Year) %>%
  summarize(Week = min(Week))

ggplot(Weekly, aes(x = Week, y = Volume)) +
  geom_line() +
  geom_smooth() +
  scale_x_continuous(breaks = year_breaks$Week, minor_breaks = NULL, labels = year_breaks$Year) +
  labs(title = "Average Daily Shares Traded vs Time",
       x = "Time") +
  theme_light()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Here is Direction over time, which is less interesting. There appear to only be 4 years in which >= 50% of the weeks didn't see a positive return (2000, 2001, 2002, 2008).

```
ggplot(Weekly, aes(x = Year, fill = Direction)) +
  geom_bar(position = "fill") +
  geom_hline(yintercept = 0.5, col = "grey45") +
  scale_x_continuous(breaks = seq(1990, 2010), minor_breaks = NULL) +
  scale_y_continuous(labels = scales::percent_format()) +
  theme_light() +
  theme(axis.title.y = element_blank(),
        legend.position = "bottom") +
  ggtitle("% of Up/Down Weeks vs Time")
```

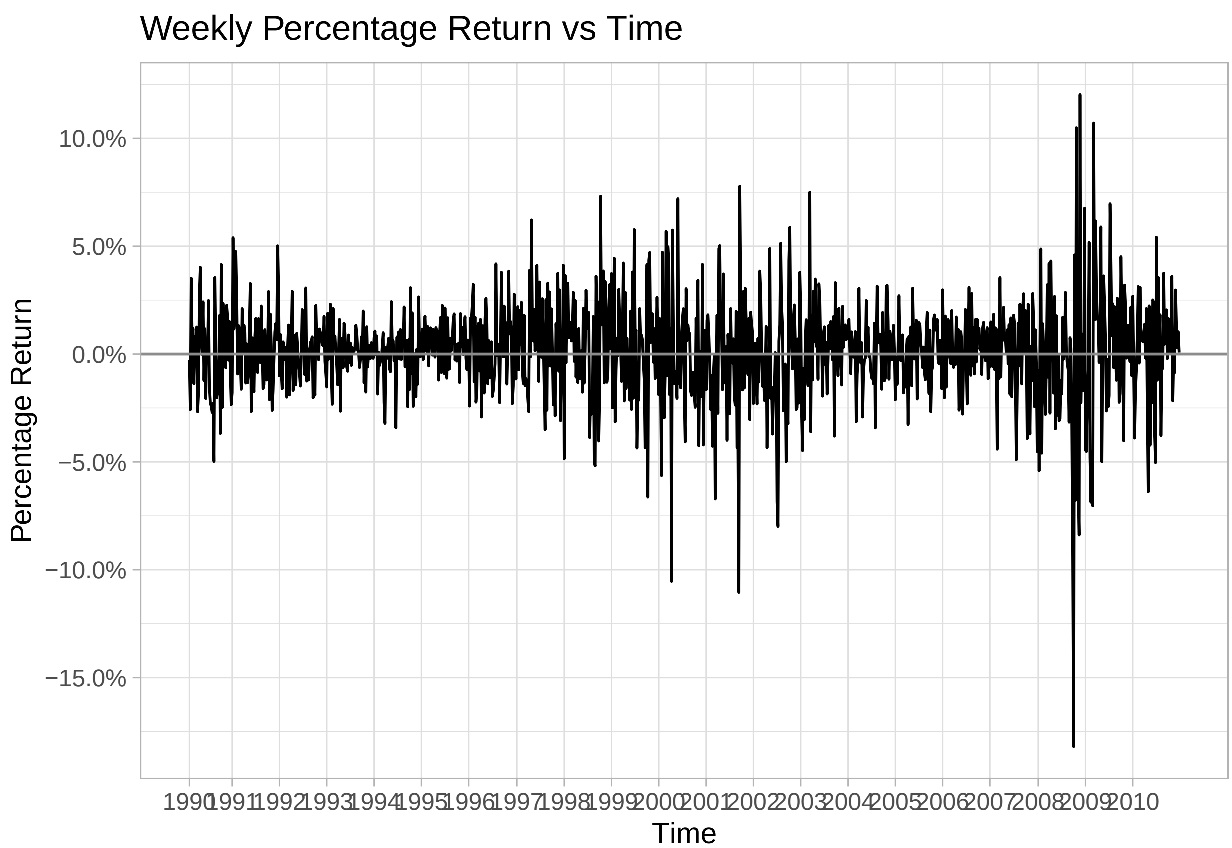


The split of the weeks into Down & Up can be seen in the table above

```
prop.table(table(Weekly$Direction))
```

```
##
##      Down        Up
## 0.4444444 0.5555556
```

```
ggplot(Weekly, aes(x = Week, y = Today / 100)) +
  geom_line() +
  scale_x_continuous(breaks = year_breaks$Week, minor_breaks = NULL, labels = year_breaks$Year) +
  scale_y_continuous(labels = scales::percent_format(), breaks = seq(-0.2, 0.2, 0.05)) +
  geom_hline(yintercept = 0, col = "grey55") +
  theme_light() +
  labs(title = "Weekly Percentage Return vs Time",
       x = "Time",
       y = "Percentage Return")
```



Q: 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?

```
glm_dir <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
               data = Weekly,
               family = "binomial")

summary(glm_dir)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## 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
```

Lag2 appears to be the only statistically significant predictor

Q: 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.

```r
predicted <- factor(ifelse(predict(glm_dir, type = "response") < 0.5, "Down", "Up"))

confusionMatrix(predicted, Weekly$Direction, positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down  Up
##       Down   54  48
##       Up    430 557
##
##                Accuracy : 0.5611
##                  95% CI : (0.531, 0.5908)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.369
##
##                   Kappa : 0.035
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9207
##             Specificity : 0.1116
##          Pos Pred Value : 0.5643
##          Neg Pred Value : 0.5294
##              Prevalence : 0.5556
##          Detection Rate : 0.5115
##    Detection Prevalence : 0.9063
##       Balanced Accuracy : 0.5161
##
##        'Positive' Class : Up
##
```

This is reflected in the very poor specificity (it does not predict the negative class well).

Q: 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).

```r
train <- Weekly[Weekly$Year <= 2008, ]
test <- Weekly[Weekly$Year > 2008, ]

glm_dir <- glm(Direction ~ Lag2,
               data = train,
               family = "binomial")

predicted <- factor(ifelse(predict(glm_dir, newdata = test, type = "response") < 0.5, "Down", "Up"))

confusionMatrix(predicted, test$Direction, positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    9  5
##       Up     34 56
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : Up
##
```

Here we get an Accuracy of *0.625.*

Q: Repeat (d) using LDA.

```r
lda_dir <- lda(Direction ~ Lag2, data = train)


predicted_lda <- predict(lda_dir, newdata = test)

confusionMatrix(data = predicted_lda$class,
                reference = test$Direction,
                positive = "Up")
```

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

```
##
##           Reference
## Prediction Down Up
##       Down    9  5
##       Up     34 56
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : Up
##
```

```r
identical(as.character(predicted_lda$class),
as.character(ifelse(predicted_lda$posterior[ ,2] < 0.5, "Down", "Up")))
```

```
## [1] TRUE
```

Q: Repeat (d) using QDA.

```r
qda_dir <- qda(Direction ~ Lag2, data = train)


predicted_qda <- predict(qda_dir, newdata = test)

confusionMatrix(data = predicted_qda$class,
                reference = test$Direction,
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    0  0
##       Up     43 61
##
##                Accuracy : 0.5865
##                  95% CI : (0.4858, 0.6823)
```

```
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.5419
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.504e-10
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5865
##          Neg Pred Value :    NaN
##              Prevalence : 0.5865
##          Detection Rate : 0.5865
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Up
##
```

Here we get an Accuracy of 0.5865.

Note that the QDA classifier just predicts Up for every test observation - it behaves identically to the naive classifier on this dataset, with a sensitivity of 1 and a specificity of 0.

Q: Repeat (d) using KNN with K = 1.

```r
test[100, "Lag2"]
```

```
## [1] 0.043
```

```r
train[c(10, 808), c("Lag2", "Direction")]
```

```
##      Lag2 Direction
## 10  0.041      Down
## 808 0.041        Up
```

```r
set.seed(1)
predicted_knn <- knn(train = data.frame(Lag2 = train$Lag2),
                     test = data.frame(Lag2 = test$Lag2),
                     cl = train$Direction,
                     k = 1,
                     prob = T)

attr(predicted_knn, "prob")[100]
```

```
## [1] 0.5
```

```r
predicted_knn[100]
```

```
## [1] Down
## Levels: Down Up
```

```r
confusionMatrix(data = predicted_knn,
                reference = test$Direction,
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   21 30
##       Up     22 31
##
##                Accuracy : 0.5
##                  95% CI : (0.4003, 0.5997)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9700
##
##                   Kappa : -0.0033
##
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.5082
##             Specificity : 0.4884
##          Pos Pred Value : 0.5849
##          Neg Pred Value : 0.4118
##              Prevalence : 0.5865
##          Detection Rate : 0.2981
##    Detection Prevalence : 0.5096
##       Balanced Accuracy : 0.4983
##
##        'Positive' Class : Up
##
```

Q: Which of these methods appears to provide the best results on this data? **LDA & Logistic Regression get the same test accuracy of 0.625**

Q: Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

KNN - selecting best K (using cross-validation):

```r
train$Today <- NULL

ctrl <- trainControl(method = "repeatedcv",
                     number = 5,
                     repeats = 5)

set.seed(111)

knn_train <- train(y = train$Direction,
                   x = train[ ,-8],
                   method = "knn",
                   metric = "Accuracy",
```

```r
                    preProcess = c("center", "scale"),
                    tuneGrid = expand.grid(k = seq(1, 50, 2)),
                    trControl = ctrl)

caret::varImp(knn_train)
```

```
## ROC curve variable importance
##
##        Importance
## Lag1      100.000
## Lag2       77.256
## Lag5       64.309
## Year       45.659
## Volume     43.735
## Week       42.513
## Lag4        4.578
## Lag3        0.000
```

```r
knn_train
```

```
## k-Nearest Neighbors
##
## 985 samples
##   8 predictor
##   2 classes: 'Down', 'Up'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 788, 788, 788, 788, 788, 787, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.4947996  -0.020756148
##    3  0.5232477   0.031373990
##    5  0.5230292   0.028769372
##    7  0.5372435   0.053353378
##    9  0.5372415   0.049612758
##   11  0.5343834   0.040819116
##   13  0.5346081   0.037598994
##   15  0.5368437   0.040363712
##   17  0.5317675   0.026161027
##   19  0.5301555   0.021608137
##   21  0.5370622   0.033488872
##   23  0.5330064   0.024861594
##   25  0.5401182   0.036317004
##   27  0.5376693   0.029998090
##   29  0.5321890   0.015868622
##   31  0.5338310   0.018687256
##   33  0.5360563   0.021431639
##   35  0.5322138   0.012669022
##   37  0.5283538   0.002535516
##   39  0.5330239   0.011432961
##   41  0.5289589   0.002400751
```

```
##   43  0.5295618   0.003081904
##   45  0.5299700   0.003587628
##   47  0.5303854   0.003745293
##   49  0.5320056   0.005552184
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 25.
```

```r
ggplot(knn_train) +
  geom_smooth() +
  theme_light() +
  scale_y_continuous(labels = scales::percent_format()) +
  ggtitle("KNN - 'K' Selection (5-repeated 5-fold cross-validation)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

caret automatically chooses the best value for k. Evaluating the performance of this new model on test:

```r
knn_pred <- predict(knn_train, newdata = test)

confusionMatrix(data = knn_pred,
                reference = test$Direction,
                positive = "Up")
```

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

```
##
##           Reference
## Prediction Down Up
##       Down   19 20
##       Up     24 41
##
##                Accuracy : 0.5769
##                  95% CI : (0.4761, 0.6732)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.6193
##
##                   Kappa : 0.1156
##
##  Mcnemar's Test P-Value : 0.6511
##
##             Sensitivity : 0.6721
##             Specificity : 0.4419
##          Pos Pred Value : 0.6308
##          Neg Pred Value : 0.4872
##              Prevalence : 0.5865
##          Detection Rate : 0.3942
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.5570
##
##        'Positive' Class : Up
##
```

```
best_predictor <- function(dataframe, response) {

  if (sum(sapply(dataframe, function(x) {is.numeric(x) | is.factor(x)})) < ncol(dataframe)) {
    stop("Make sure that all variables are of class numeric/factor!")
  }

  # pre-allocate vectors
  varname <- c()
  vartype <- c()
  R2 <- c()
  R2_log <- c()
  R2_quad <- c()
  AIC <- c()
  AIC_log <- c()
  AIC_quad <- c()
  y <- dataframe[ ,response]

  # # # # # NUMERIC RESPONSE # # # # #
  if (is.numeric(y)) {

    for (i in 1:ncol(dataframe)) {

      x <- dataframe[ ,i]
      varname[i] <- names(dataframe)[i]

      if (class(x) %in% c("numeric", "integer")) {
        vartype[i] <- "numeric"
```

```
    } else {
      vartype[i] <- "categorical"
    }

    if (!identical(y, x)) {

      # linear: y ~ x
      R2[i] <- summary(lm(y ~ x))$r.squared

      # log-transform: y ~ log(x)
      if (is.numeric(x)) {
        if (min(x) <= 0) { # if y ~ log(x) for min(x) <= 0, do y ~ log(x + abs(min(x)) + 1)
          R2_log[i] <- summary(lm(y ~ log(x + abs(min(x)) + 1)))$r.squared
        } else {
          R2_log[i] <- summary(lm(y ~ log(x)))$r.squared
        }
      } else {
        R2_log[i] <- NA
      }

      # quadratic: y ~ x + x^2
      if (is.numeric(x)) {
        R2_quad[i] <- summary(lm(y ~ x + I(x^2)))$r.squared
      } else {
        R2_quad[i] <- NA
      }

    } else {
      R2[i] <- NA
      R2_log[i] <- NA
      R2_quad[i] <- NA
    }
  }

  print(paste("Response variable:", response))

  data.frame(varname,
             vartype,
             R2 = round(R2, 3),
             R2_log = round(R2_log, 3),
             R2_quad = round(R2_quad, 3)) %>%
    mutate(max_R2 = pmax(R2, R2_log, R2_quad, na.rm = T)) %>%
    arrange(desc(max_R2))


  # # # # # CATEGORICAL RESPONSE # # # # #
} else {

  for (i in 1:ncol(dataframe)) {

    x <- dataframe[ ,i]
    varname[i] <- names(dataframe)[i]
```

```r
      if (class(x) %in% c("numeric", "integer")) {
        vartype[i] <- "numeric"
      } else {
        vartype[i] <- "categorical"
      }

      if (!identical(y, x)) {

        # linear: y ~ x
        AIC[i] <- summary(glm(y ~ x, family = "binomial"))$aic

        # log-transform: y ~ log(x)
        if (is.numeric(x)) {
          if (min(x) <= 0) { # if y ~ log(x) for min(x) <= 0, do y ~ log(x + abs(min(x)) + 1)
            AIC_log[i] <- summary(glm(y ~ log(x + abs(min(x)) + 1), family = "binomial"))$aic
          } else {
            AIC_log[i] <- summary(glm(y ~ log(x), family = "binomial"))$aic
          }
        } else {
          AIC_log[i] <- NA
        }

        # quadratic: y ~ x + x^2
        if (is.numeric(x)) {
          AIC_quad[i] <- summary(glm(y ~ x + I(x^2), family = "binomial"))$aic
        } else {
          AIC_quad[i] <- NA
        }

      } else {
        AIC[i] <- NA
        AIC_log[i] <- NA
        AIC_quad[i] <- NA
      }
    }

    print(paste("Response variable:", response))

    data.frame(varname,
               vartype,
               AIC = round(AIC, 3),
               AIC_log = round(AIC_log, 3),
               AIC_quad = round(AIC_quad, 3)) %>%
      mutate(min_AIC = pmin(AIC, AIC_log, AIC_quad, na.rm = T)) %>%
      arrange(min_AIC)
  }
}
```

```r
train$junk_1 <- rnorm(nrow(train))
train$junk_2 <- runif(nrow(train))
train$junk_3 <- factor(as.numeric(rnorm(nrow(train)) > 0))
train$junk_4 <- rnorm(nrow(train))
train$junk_5 <- runif(nrow(train))
```

```r
train$junk_6 <- factor(as.numeric(rnorm(nrow(train)) > 0))
train$junk_7 <- rnorm(nrow(train))
train$junk_8 <- runif(nrow(train))
train$junk_9 <- factor(as.numeric(rnorm(nrow(train)) > 0))
train$junk_10 <- rnorm(nrow(train))

best_predictor(train, "Direction")
```

```
## [1] "Response variable: Direction"
```

```
##      varname     vartype      AIC  AIC_log AIC_quad  min_AIC
## 1       Lag1     numeric 1354.446 1354.199 1356.442 1354.199
## 2       Lag2     numeric 1354.543 1355.148 1355.435 1354.543
## 3     junk_6 categorical 1355.795       NA       NA 1355.795
## 4     junk_9 categorical 1356.462       NA       NA 1356.462
## 5     Volume     numeric 1356.838 1356.751 1358.833 1356.751
## 6     junk_4     numeric 1357.707 1358.426 1356.864 1356.864
## 7       Year     numeric 1357.111 1357.112 1358.772 1357.111
## 8       Week     numeric 1357.260 1358.273 1359.076 1357.260
## 9    junk_10     numeric 1357.319 1357.314 1359.128 1357.314
## 10    junk_2     numeric 1357.358 1357.721 1359.071 1357.358
## 11      Lag5     numeric 1357.365 1358.527 1358.188 1357.365
## 12    junk_7     numeric 1357.460 1357.617 1359.344 1357.460
## 13    junk_5     numeric 1358.566 1357.927 1359.136 1357.927
## 14    junk_1     numeric 1358.008 1357.938 1359.327 1357.938
## 15    junk_8     numeric 1357.945 1358.487 1359.603 1357.945
## 16      Lag3     numeric 1358.354 1358.038 1360.286 1358.038
## 17      Lag4     numeric 1358.497 1358.685 1359.007 1358.497
## 18    junk_3 categorical 1358.700       NA       NA 1358.700
## 19 Direction categorical       NA       NA       NA       NA
```

```r
train <- train %>%
  mutate(Lag_avg_abs = abs(Lag1) + abs(Lag2) + abs(Lag3) + abs(Lag4) + abs(Lag5),
         Lag_pos_cnt = (Lag1 > 0) + (Lag2 > 0) + (Lag3 > 0) + (Lag4 > 0) + (Lag5 > 0)) %>%
  group_by(Year) %>%
  mutate(Week_of_year = row_number()) %>%
  ungroup() %>%
  mutate(Week_of_year = case_when(Year == 1990 ~ as.numeric(Week_of_year + 5),
                                  TRUE ~ as.numeric(Week_of_year))) %>% # data appears to start 5wks in
  mutate(Quarter = factor(case_when(Week_of_year <= 13 ~ "Q1",
                                    Week_of_year <= 26 ~ "Q2",
                                    Week_of_year <= 39 ~ "Q3",
                                    TRUE ~ "Q4"))) %>%
  select(Direction, Lag1, Lag2, Lag_avg_abs, Lag_pos_cnt, Quarter)


test <- test %>%
  mutate(Lag_avg_abs = abs(Lag1) + abs(Lag2) + abs(Lag3) + abs(Lag4) + abs(Lag5),
         Lag_pos_cnt = (Lag1 > 0) + (Lag2 > 0) + (Lag3 > 0) + (Lag4 > 0) + (Lag5 > 0)) %>%
  group_by(Year) %>%
  mutate(Week_of_year = row_number()) %>%
  ungroup() %>%
  mutate(Quarter = factor(case_when(Week_of_year <= 13 ~ "Q1",
```

```r
                                    Week_of_year <= 26 ~ "Q2",
                                    Week_of_year <= 39 ~ "Q3",
                                    TRUE ~ "Q4"))) %>%
  select(Direction, Lag1, Lag2, Lag_avg_abs, Lag_pos_cnt, Quarter)


glimpse(train)
```

```
## Rows: 985
## Columns: 6
## $ Direction   <fct> Down, Down, Up, Up, Up, Down, Up, Up, Up, Down, Down, Up, ~
## $ Lag1        <dbl> 0.816, -0.270, -2.576, 3.514, 0.712, 1.178, -1.372, 0.807,~
## $ Lag2        <dbl> 1.572, 0.816, -0.270, -2.576, 3.514, 0.712, 1.178, -1.372,~
## $ Lag_avg_abs <dbl> 10.037, 6.823, 9.170, 8.748, 7.888, 8.250, 9.352, 7.583, 4~
## $ Lag_pos_cnt <int> 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 2~
## $ Quarter     <fct> Q1, Q1, Q1, Q1, Q1, Q1, Q1, Q1, Q2, Q2, Q2, Q2, Q2, Q2, Q2~
```

```r
glm_dir_2 <- glm(Direction ~ . + Lag1:Lag2,
             data = train,
             family = "binomial")


predicted <- factor(ifelse(predict(glm_dir_2, newdata = test, type = "response") < 0.5, "Down", "Up"))

confusionMatrix(predicted, test$Direction, positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   18 16
##       Up     25 45
##
##                Accuracy : 0.6058
##                  95% CI : (0.5051, 0.7002)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.3847
##
##                   Kappa : 0.1613
##
##  Mcnemar's Test P-Value : 0.2115
##
##             Sensitivity : 0.7377
##             Specificity : 0.4186
##          Pos Pred Value : 0.6429
##          Neg Pred Value : 0.5294
##              Prevalence : 0.5865
##          Detection Rate : 0.4327
##    Detection Prevalence : 0.6731
##       Balanced Accuracy : 0.5782
##
##        'Positive' Class : Up
##
```

The classifier performed better on the test data than the baseline approach with an accuracy of 60.58%, but this is slightly worse than the simple LDA & Logistic classifiers which scored 62.5%.

11: In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

Q: Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

```
Auto$mpg01 <- factor(as.numeric(Auto$mpg > median(Auto$mpg)))

table(Auto$mpg01)
```

```
##
##   0   1
## 196 196
```

Q: Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

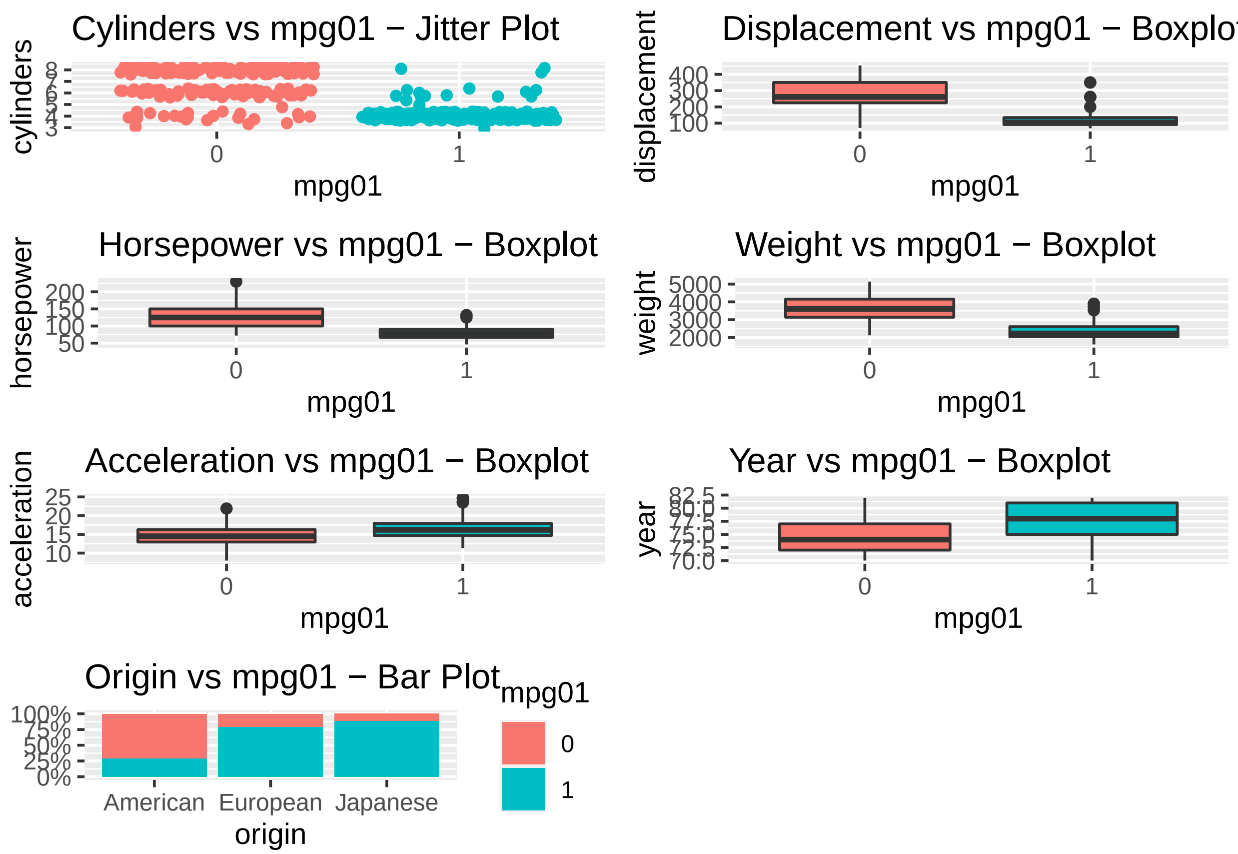
Excluding name (categorical with 301 unique values) and mpg (used to create mpg01), and converting origin to a factor:

```
Auto$name <- NULL
Auto$mpg <- NULL

Auto$origin <- factor(Auto$origin, labels = c("American", "European", "Japanese"))

glimpse(Auto)
```

```
## Rows: 392
## Columns: 8
## $ cylinders    <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 6, 6, 6, 4, ~
## $ displacement <dbl> 307, 350, 318, 304, 302, 429, 454, 440, 455, 390, 383, 34~
## $ horsepower   <dbl> 130, 165, 150, 150, 140, 198, 220, 215, 225, 190, 170, 16~
## $ weight       <dbl> 3504, 3693, 3436, 3433, 3449, 4341, 4354, 4312, 4425, 385~
## $ acceleration <dbl> 12.0, 11.5, 11.0, 12.0, 10.5, 10.0, 9.0, 8.5, 10.0, 8.5, ~
## $ year         <dbl> 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 7~
## $ origin       <fct> American, American, American, American, American, America~
## $ mpg01        <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, ~
```

```
g1 <- ggplot(Auto, aes(x = mpg01, y = cylinders, col = mpg01)) +
  geom_jitter() +
  theme(legend.position = "none") +
  ggtitle("Cylinders vs mpg01 - Jitter Plot")

g2 <- ggplot(Auto, aes(x = mpg01, y = displacement, fill = mpg01)) +
  geom_boxplot() +
```

```r
  theme(legend.position = "none") +
  ggtitle("Displacement vs mpg01 - Boxplot")

g3 <- ggplot(Auto, aes(x = mpg01, y = horsepower, fill = mpg01)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  ggtitle("Horsepower vs mpg01 - Boxplot")

g4 <- ggplot(Auto, aes(x = mpg01, y = weight, fill = mpg01)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  ggtitle("Weight vs mpg01 - Boxplot")

g5 <- ggplot(Auto, aes(x = mpg01, y = acceleration, fill = mpg01)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  ggtitle("Acceleration vs mpg01 - Boxplot")

g6 <- ggplot(Auto, aes(x = mpg01, y = year, fill = mpg01)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  ggtitle("Year vs mpg01 - Boxplot")

g7 <- ggplot(Auto, aes(x = origin, fill = mpg01)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  theme(axis.title.y = element_blank()) +
  ggtitle("Origin vs mpg01 - Bar Plot")


grid.arrange(g1, g2, g3, g4, g5, g6, g7,
             ncol = 2,
             nrow = 4)
```



Q: Split the data into a training set and a test set.

```
set.seed(444)
index <- createDataPartition(y = Auto$mpg01, p = 0.5, list = F)

train <- Auto[index, ]
test <- Auto[-index, ]

nrow(train) / nrow(Auto)
```

```
## [1] 0.5
```

Q: Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1)

lda_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "lda",
                 trControl = ctrl)
```

```
1 - lda_mpg$results$Accuracy # cv error
```

```
## [1] 0.09125731
```

test error:

```
predicted_lda <- predict(lda_mpg, newdata = test, type = "raw") # as opposed to type = "prob"

mean(predicted_lda != test$mpg01)
```

```
## [1] 0.122449
```

Q: Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
set.seed(2)

qda_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "qda",
                 trControl = ctrl)
```

train (cross-validation) error:

```
1 - qda_mpg$results$Accuracy
```

```
## [1] 0.08991228
```

test error:

```
predicted_qda <- predict(qda_mpg, newdata = test, type = "raw")

mean(predicted_qda != test$mpg01)
```

```
## [1] 0.1173469
```

In this case, QDA appears to perform better than LDA with respect to test error, and slightly better in terms of CV error.

Q: Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
set.seed(3)

log_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "glm",
                 family = "binomial",
                 trControl = ctrl)
```

train (cross-validation) error:

```r
1 - log_mpg$results$Accuracy
```

```
## [1] 0.1084405
```

test error:

```r
predicted_log <- predict(log_mpg, newdata = test, type = "raw")

mean(predicted_log != test$mpg01)
```

```
## [1] 0.1173469
```

Logistic Regression performs better than LDA & QDA with respect to CV error & test error.

Q: Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```r
set.seed(4)

knn_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 trControl = ctrl,
                 tuneGrid = expand.grid(k = seq(1, 85, 3)))

knn_mpg
```

```
## k-Nearest Neighbors
##
## 196 samples
##   4 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (4), scaled (4)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 176, 177, 176, 176, 176, 176, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8779532  0.7554422
##    4  0.8999805  0.7995964
##    7  0.9033138  0.8062631
##   10  0.9033138  0.8062631
##   13  0.9033138  0.8062631
##   16  0.9033138  0.8062631
##   19  0.9033138  0.8062631
##   22  0.9033138  0.8062631
##   25  0.9033138  0.8062631
##   28  0.9033138  0.8062631
##   31  0.9033138  0.8062631
```

```
##   34  0.9033138  0.8062631
##   37  0.9033138  0.8062631
##   40  0.9033138  0.8062631
##   43  0.9033138  0.8062631
##   46  0.9033138  0.8062631
##   49  0.9033138  0.8062631
##   52  0.9033138  0.8062631
##   55  0.9033138  0.8062631
##   58  0.9033138  0.8062631
##   61  0.9033138  0.8062631
##   64  0.9049805  0.8095964
##   67  0.9049805  0.8095964
##   70  0.9066472  0.8129297
##   73  0.9100682  0.8197621
##   76  0.9083138  0.8162631
##   79  0.9084016  0.8164288
##   82  0.9067349  0.8130955
##   85  0.9034016  0.8065072
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 73.
```

train (cross-validation) error:

```r
1 - max(knn_mpg$results$Accuracy)
```

```
## [1] 0.08993177
```

test error:

```r
predicted_knn <- predict(knn_mpg, newdata = test, type = "raw")

mean(predicted_knn != test$mpg01)
```

```
## [1] 0.1122449
```

12 (a) Power() Function

```r
Power <- function() {
  2^3
}

Power()
```

```
## [1] 8
```

Q: Create a new function, Power2(), that allows you to pass any two numbers, x and a, and prints out the value of x^a. You can do this by beginning your function with the line Power2=function (x,a){. You should be able to call your function by entering, for instance, Power2(3,8) on the command line

```r
Power2 <- function(x, a) {
  x^a
}

Power2(3, 8)
```

```
## [1] 6561
```

Q: Using the Power2() function that you just wrote,

```r
Power2(10, 3)
```

```
## [1] 1000
```

```r
Power2(8, 17)
```

```
## [1] 2.2518e+15
```

```r
Power2(131, 3)
```

```
## [1] 2248091
```

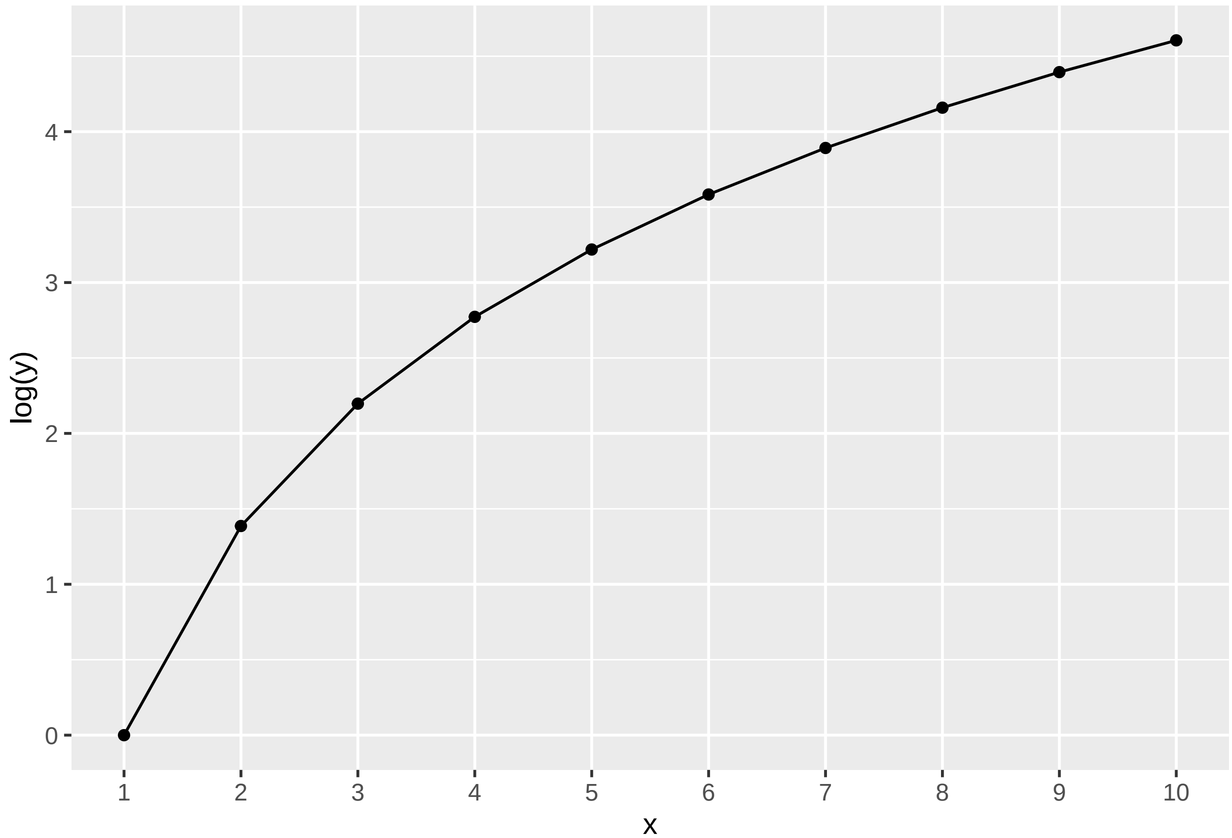
Q: Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply printing it to the screen. That is, if you store the value x^a in an object called result within your function, then you can simply return() this result, using the following line: return(result). This should be the last line in your function, before the } symbol.

```r
Power3 <- function(x, a) {
  result <- x^a
  result
}

Power3(3, 8)
```

```
## [1] 6561
```

Q: Now using the Power3() function, create a plot of [Math Processing Error]. The x-axis should display a range of integers from 1 to 10, and the y-axis should display [Math Processing Error]. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale.

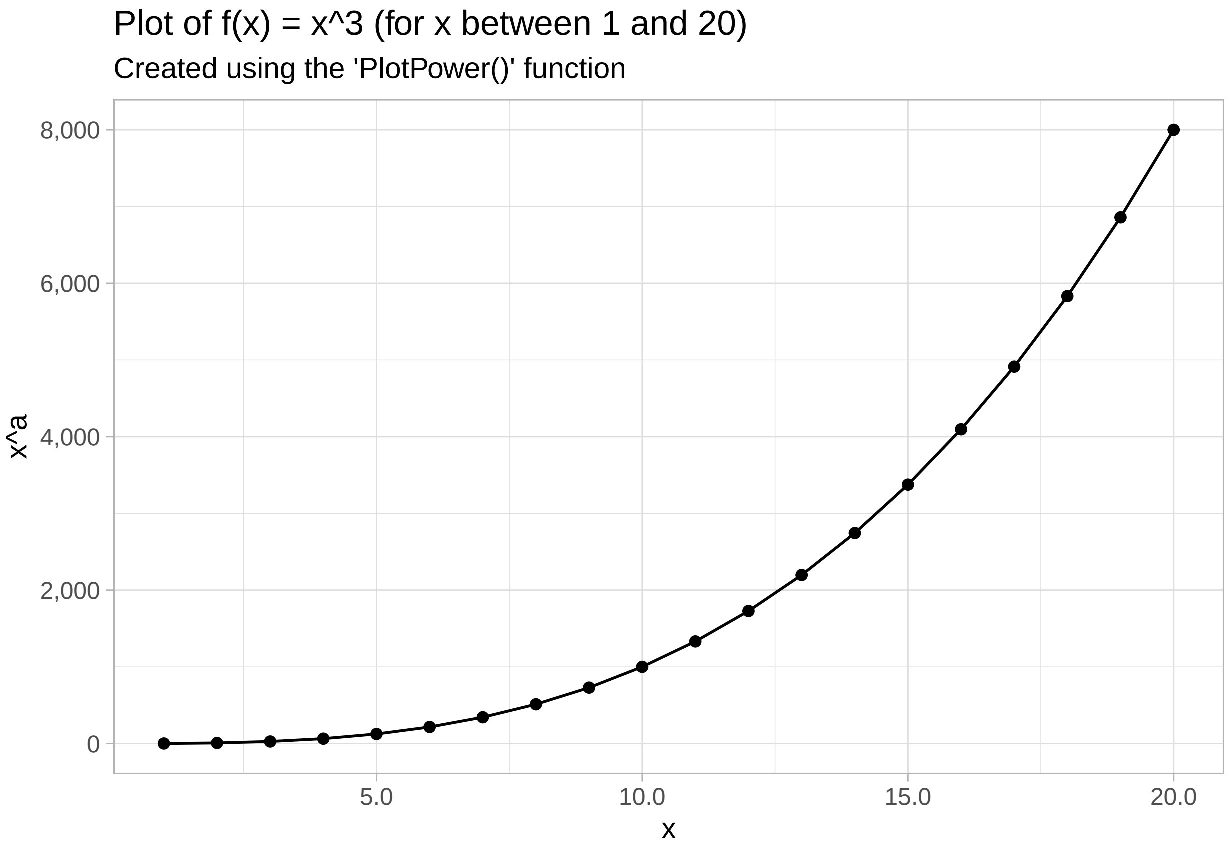
```r
data.frame(x = 1:10, y = Power3(1:10, 2)) %>%
  ggplot(aes(x = x, y = log(y))) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10, minor_breaks = NULL)
```



Q: Create a function, PlotPower(), that allows you to create a plot of x against x^a for a fixed a and for a range of values of x.

```
PlotPower <- function(x, a, col = "black") {
  ggplot(mapping = aes(x = x, y = x^a)) +
    geom_point(col = col) +
    geom_line(col = col) +
    scale_x_continuous(labels = scales::comma_format()) +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_light() +
    labs(title = paste0("Plot of f(x) = x^", as.character(a), " (for x between ", min(x), " and ", max(x
         subtitle = "Created using the 'PlotPower()' function")
}
```

```
PlotPower(1:20, 3)
```



#13.Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.#

```
Boston$crim <- factor(ifelse(Boston$crim > median(Boston$crim), 1, 0))

glimpse(Boston)
```

```
## Rows: 506
## Columns: 14
## $ crim    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ black   <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
```

```r
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 5)
```

Logistic Regression:

```r
set.seed(1)
log_crim <- train(crim ~ .,
                  data = Boston,
                  method = "glm",
                  family = "binomial",
                  trControl = ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
log_crim
```

```
## Generalized Linear Model
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 455, 456, 454, 456, 456, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9028549  0.8057423
```

LDA

```r
set.seed(2)
lda_crim <- train(crim ~ .,
                  data = Boston,
                  method = "lda",
                  trControl = ctrl)

lda_crim
```

```
## Linear Discriminant Analysis
##
```

```
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 455, 456, 456, 455, 454, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8498998  0.6996644
```

QDA

```
set.seed(3)
qda_crim <- train(crim ~ .,
                  data = Boston,
                  method = "qda",
                  trControl = ctrl)

qda_crim
```

```
## Quadratic Discriminant Analysis
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 456, 455, 455, 455, 456, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8894561  0.7788975
```

KNN

```
set.seed(4)
knn_crim <- train(crim ~ .,
                  data = Boston,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  trControl = ctrl,
                  tuneGrid = expand.grid(k = seq(1, 20, 2)))

knn_crim
```

```
## k-Nearest Neighbors
##
## 506 samples
##  13 predictor
```

```
##   2 classes: '0', '1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 455, 456, 456, 455, 455, 456, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9145970  0.8292052
##    3  0.9181104  0.8362439
##    5  0.9153345  0.8306748
##    7  0.9047128  0.8094000
##    9  0.8779164  0.7557756
##   11  0.8664404  0.7327928
##   13  0.8664323  0.7327720
##   15  0.8703940  0.7407136
##   17  0.8684169  0.7367859
##   19  0.8648404  0.7296460
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

KNN actually performed the best

```
varImp(knn_crim)
```

```
## ROC curve variable importance
##
##         Importance
## nox         100.00
## dis          86.05
## age          82.64
## indus        80.85
## tax          78.00
## rad          73.48
## lstat        59.43
## medv         50.63
## zn           47.35
## ptratio      45.59
## black        42.68
## rm           20.66
## chas          0.00
```

```
knn_best_to_worst <- as.data.frame(varImp(knn_crim)$importance) %>%
  rownames_to_column("var") %>%
  arrange(desc(X0)) %>%
  pull(var)

Boston <- Boston %>%
  dplyr::select(c(knn_best_to_worst, "crim"))
```

```
## Note: Using an external vector in selections is ambiguous.
```

```
## i Use `all_of(knn_best_to_worst)` instead of `knn_best_to_worst` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
cv_accuracy <- c()
chosen_k <- c()

set.seed(5)

for (i in 1:13) {
  knn_crim_temp <- as.data.frame(Boston[ ,c(1:i, 14)])
  knn_crim_temp <- train(crim ~ .,
                         data = knn_crim_temp,
                         method = "knn",
                         preProcess = c("center", "scale"),
                         trControl = ctrl,
                         tuneGrid = expand.grid(k = seq(1, 20, 2)))

  cv_accuracy[i] <- max(knn_crim_temp$results$Accuracy)
  chosen_k[i] <- as.numeric(knn_crim_temp$bestTune)
}

data.frame(pred_num = 1:13, cv_accuracy = cv_accuracy, chosen_k = chosen_k) %>%
  arrange(desc(cv_accuracy))
```

```
##    pred_num cv_accuracy chosen_k
## 1         1   0.9549303        1
## 2        10   0.9407704        5
## 3        11   0.9407192        3
## 4         6   0.9378703        1
## 5         5   0.9356350        5
## 6         4   0.9340434        3
## 7         7   0.9340048        1
## 8         9   0.9324929        1
## 9        12   0.9320217        5
## 10        8   0.9316078        3
## 11       13   0.9170664        5
## 12        2   0.8960546        1
## 13        3   0.8877502        7
```