HW 4

Trey Davidson

2024-09-19

# Problem 6

A.) Approx 0.3775

B.) 50 hours

# Problem 9

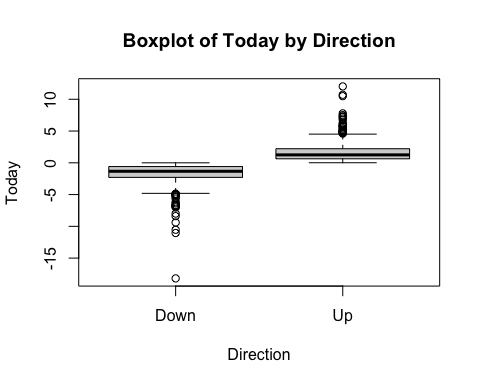
A.) 37/137

B.) 0.19

# Problem 13

A.)

library(ISLR2)  
  
attach(Weekly)  
  
boxplot(Today ~ Direction,   
 xlab = "Direction", ylab = "Today",   
 main = "Boxplot of Today by Direction")



#This plot tells us that if today is extermly postive or extermly neagtive then the weekly will direction will likely pull in the direction. The usefulness of this is questioanble though as if the market goes down by 10% on Monday, then it will be very unlikely that it will be up 11.1111% required to be up by Friday so the usefulness of this predicatior is uknown.

B.)

model <- glm(Direction ~ Today + Lag1 + Lag2 + Volume, data = Weekly, family = binomial)

## Warning: glm.fit: algorithm did not converge

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

summary(model)

##   
## Call:  
## glm(formula = Direction ~ Today + Lag1 + Lag2 + Volume, family = binomial,   
## data = Weekly)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -14.080 383.957 -0.037 0.971  
## Today 981.188 16703.915 0.059 0.953  
## Lag1 -1.328 188.512 -0.007 0.994  
## Lag2 4.011 220.087 0.018 0.985  
## Volume 4.572 511.516 0.009 0.993  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1.4962e+03 on 1088 degrees of freedom  
## Residual deviance: 1.4882e-05 on 1084 degrees of freedom  
## AIC: 10  
##   
## Number of Fisher Scoring iterations: 25

# None of these seem to be statistically significant

C.)

predicted\_probs <- predict(model, type = "response")  
  
prob\_assoicate\_up <- 0.5  
  
# Creates a vector that will deterime if it meets the threshold to be up  
meets\_prob\_requirment <- (predicted\_probs > prob\_assoicate\_up)  
  
  
# Converts True -> Up and False -> down  
predicted\_direction <- ifelse(meets\_prob\_requirment, "Up", "Down")  
  
confusion\_matrix <- table(Actual = Direction, Predicted = predicted\_direction)  
  
confusion\_matrix

## Predicted  
## Actual Down Up  
## Down 484 0  
## Up 0 605

# The confusion matrix is indicating nothing to be about the types of mistakes as its apperantly gettting everything correct

D.)

### Model LOG  
# Filter data for 1990 to 2008  
Weekly1990\_2008 <- Weekly[Weekly$Year >= 1990 & Weekly$Year <= 2008, ]  
  
# Filter data for 2009 to 2010  
Weekly2009\_2010 <- Weekly[Weekly$Year >= 2009 & Weekly$Year <= 2010, ]  
  
# Create logistic regression model  
model\_2 <- glm(Direction ~ Lag2, data = Weekly1990\_2008, family = binomial)  
  
# Predict results of model 2  
predicted\_probs <- predict(model\_2, newdata = Weekly2009\_2010, type = "response")  
  
# Classify predictions based on threshold  
  
threshold\_model\_2 <- 0.5  
predicted\_direction <- ifelse(predicted\_probs > threshold\_model\_2, "Up", "Down")  
  
# Create confusion matrix using the actual values from the Weekly2009\_2010 dataset  
confusion\_matrix <- table(Actual = Weekly2009\_2010$Direction, Predicted = predicted\_direction)  
  
# Print the confusion matrix  
confusion\_matrix

## Predicted  
## Actual Down Up  
## Down 9 34  
## Up 5 56

### Model LDA  
  
# Load the MASS package  
library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':  
##   
## Boston

# Create the LDA model  
LDA\_model <- lda(Direction ~ Lag2, data = Weekly1990\_2008)  
  
# Make predictions (class labels)  
predicted\_probs <- predict(LDA\_model, newdata = Weekly2009\_2010)  
  
# Get predicted class labels  
predicted\_direction\_LDA <- predicted\_probs$class  
  
# Create the confusion matrix using the actual values from the Weekly2009\_2010 dataset  
confusion\_matrix\_LDA <- table(Actual = Weekly2009\_2010$Direction, Predicted = predicted\_direction\_LDA)  
  
# Print the confusion matrix  
confusion\_matrix\_LDA

## Predicted  
## Actual Down Up  
## Down 9 34  
## Up 5 56

### Model QDA  
  
# Create the QDA model  
QDA\_model <- qda(Direction ~ Lag2, data = Weekly1990\_2008)  
  
# Make predictions (class labels)  
predicted\_probs <- predict(QDA\_model, newdata = Weekly2009\_2010)  
  
# Get predicted class labels  
predicted\_direction\_QDA <- predicted\_probs$class  
  
# Create the confusion matrix using the actual values from the Weekly2009\_2010 dataset  
confusion\_matrix\_QDA <- table(Actual = Weekly2009\_2010$Direction, Predicted = predicted\_direction\_QDA)  
  
# Print the confusion matrix  
confusion\_matrix\_QDA

## Predicted  
## Actual Down Up  
## Down 0 43  
## Up 0 61

### Model KNN with k = 1  
# Load the class package  
library(class)  
  
# Prepare the training and test data  
train\_data <- Weekly1990\_2008  
test\_data <- Weekly2009\_2010  
  
# Separate predictors and response for training  
train\_x <- as.matrix(train\_data[, "Lag2"]) # Convert to matrix  
train\_y <- train\_data$Direction  
  
# Separate predictors for testing  
test\_x <- as.matrix(test\_data[, "Lag2"]) # Convert to matrix  
  
# Perform k-NN with k = 1  
predicted\_direction\_KNN <- knn(train = train\_x, test = test\_x, cl = train\_y, k = 1)  
  
# Create the confusion matrix  
confusion\_matrix\_KNN <- table(Actual = test\_data$Direction, Predicted = predicted\_direction\_KNN)  
  
# Print the confusion matrix  
confusion\_matrix\_KNN

## Predicted  
## Actual Down Up  
## Down 21 22  
## Up 30 31

### Naive Bayes  
  
library(e1071)  
  
# Prepare the training and test data  
train\_data <- Weekly1990\_2008  
test\_data <- Weekly2009\_2010  
  
# Fit the Naive Bayes model  
naive\_bayes\_model <- naiveBayes(Direction ~ Lag2, data = train\_data)  
  
# Make predictions on the test data  
predicted\_direction\_Bayes <- predict(naive\_bayes\_model, newdata = test\_data)  
  
# Create the confusion matrix  
confusion\_matrix\_Bayes <- table(Actual = test\_data$Direction, Predicted = predicted\_direction\_Bayes)  
  
# Print the confusion matrix  
confusion\_matrix\_Bayes

## Predicted  
## Actual Down Up  
## Down 0 43  
## Up 0 61

i.) The model that did best by using the confusion table is LDA.

# Prepare the training and test data  
train\_data <- Weekly1990\_2008  
test\_data <- Weekly2009\_2010  
  
# Separate predictors and response for training  
train\_x <- as.matrix(train\_data[, c("Today")]) # Include Lag1, Lag2, and Lag3  
train\_y <- train\_data$Direction  
  
# Separate predictors for testing  
test\_x <- as.matrix(test\_data[, c("Today")]) # Include Lag1, Lag2, and Lag3  
  
# Perform k-NN with k = 27  
predicted\_direction\_KNN <- knn(train = train\_x, test = test\_x, cl = train\_y, k = 20)  
  
# Create the confusion matrix  
confusion\_matrix\_KNN <- table(Actual = test\_data$Direction, Predicted = predicted\_direction\_KNN)  
  
# Print the confusion matrix  
print(confusion\_matrix\_KNN)

## Predicted  
## Actual Down Up  
## Down 43 0  
## Up 0 61

Out of everything i did the one above performed the best.

### Problem 15

### A  
Power <- function()  
{  
 print(2 ^ 3)  
}  
  
Power

## function()  
## {  
## print(2 ^ 3)  
## }

### B.   
Power2 <- function(x, a)  
{  
 print(x ^ a)  
}  
  
  
Power2(3, 8)

## [1] 6561

# 3^8  
Power2(10, 3)

## [1] 1000

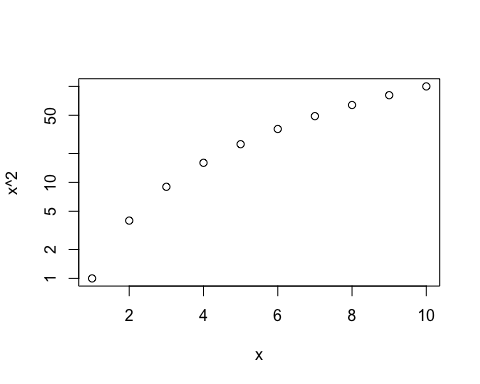
Power2(8, 17)

## [1] 2.2518e+15

Power2(131, 3)

## [1] 2248091

Power3 <- function(x, a)  
{  
 return(x^a)  
}  
  
x <- seq(1, 10, by = 1)  
  
plot(x, Power3(x, 2),  
 xlab = "x", ylab = "x^2",  
 log = "y")



PlotPower <- function(range, a)  
{  
 x <- range  
 plot(x, Power3(x, 2),  
 xlab = "x", ylab = "x^2",  
 log = "y")  
   
}