Homework 3



#for nicer table output  
library(knitr)

# Conceptual Problems

* Book Problem 4

1. 10% of the data will be used as the distribution is uniform and our range accounts for 10% of the total range of [0, 1]. (Not asked for, but we can show this via simulation.)

set.seed(10)  
unif\_values <- runif(100000)  
mean(unif\_values < 0.65 & unif\_values > 0.55)



[1] 0.10049

1. Since we have 0.1 probability of X1 being in the range and, separately/independently, a 0.1 probability of X2 being in the range, we have 0.1\*0.1 = 0.01 probability of a value falling in the range. So 1% of our observations would fall in the range.



unif\_values\_2 <- runif(100000)  
mean((unif\_values< 0.65 & unif\_values > 0.55) & (unif\_values\_2 > 0.3 & unif\_values\_2 < 0.4))

[1] 0.01098

1. Here only of the observations fall in the region! Almost no chance of a value falling in the interval of choice.



1. We can see that, if we want to use the same ‘size’ region, as grows we will have fewer and fewer observations. That means we either need to have way more observations or we need to make our ‘near’ range larger as grows.



1. Let be the length.
   1. For , 10% of the observations fall in any area of length L = 0.1 in [0,1].
   2. For , we want . This means



* 1. Generically, for we want or . For a few values of we can see the lengths below:

p <- 1:30  
L <- (0.1)^(1/p)  
L <- round(L, 3)  
data.frame(p, L) |>  
 knitr::kable()

| p | L |
| --- | --- |
| 1 | 0.100 |
| 2 | 0.316 |
| 3 | 0.464 |
| 4 | 0.562 |
| 5 | 0.631 |
| 6 | 0.681 |
| 7 | 0.720 |
| 8 | 0.750 |
| 9 | 0.774 |
| 10 | 0.794 |
| 11 | 0.811 |
| 12 | 0.825 |
| 13 | 0.838 |
| 14 | 0.848 |
| 15 | 0.858 |
| 16 | 0.866 |
| 17 | 0.873 |
| 18 | 0.880 |
| 19 | 0.886 |
| 20 | 0.891 |
| 21 | 0.896 |
| 22 | 0.901 |
| 23 | 0.905 |
| 24 | 0.909 |
| 25 | 0.912 |
| 26 | 0.915 |
| 27 | 0.918 |
| 28 | 0.921 |
| 29 | 0.924 |
| 30 | 0.926 |

* Book Problem 5:



* 1. If the Bayes decision boundary is truly linear, then we would expect LDA to do better on the test set as QDA would be finding relationships that don’t truly exist in the data and, therefore, should not exist in the test set. As QDA is able to better fit data, we would expect QDA to perform better on the training set.



* 1. If the Bayes decision boundary is truly non-linear, we expect QDA to perform better on both the training and test set. QDA generally fits more closely to the training data due to being a more complicated of a model. With a non-linear decision boundary, we’d also expect QDA to translate better to the test set.
  2. We would generally assume that QDA’s performance would improve relative to LDA as we’d have more data to fit the more complicated model better. If the true boundary is linear (the equal covariance assumption being reasonable), then LDA may continue to outpeform QDA. QDA should get closer and closer to the LDA fit in that situation as well.
  3. False - the performance of QDA may approach that of LDA as the sample size increases but QDA is generally going to be fitting too complicated of a relationship and, thus, won’t perform as well on the test set.
* Book Problem 6 Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficients given by intercept = -6, slope on hours studied = 0.05, and slope on undergraduate GPA of 1.
  1. Our (esitmated) log-odds of success are equal to . Our probability of success is = 0.3775407



* 1. To have a 50% probability we would also have a log-odds of 0. Setting our log-odds equation to 0 with GPA equal to 3.5 we get which implies we need to have hours = 50.



* Book Problem 9
  1. Let be the probability of success. If the odds are 0.37, then . Solving for this gives 0.27. So 27% of people will default.



* 1. The odds are so here we have odds of 0.1905 of defaulting.



# Implementation Problems

* Book Problem 14

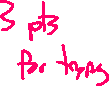
1. The binary variable can be created in a lot of ways. We’ll use mutate() and if\_else() from the dplyr package.

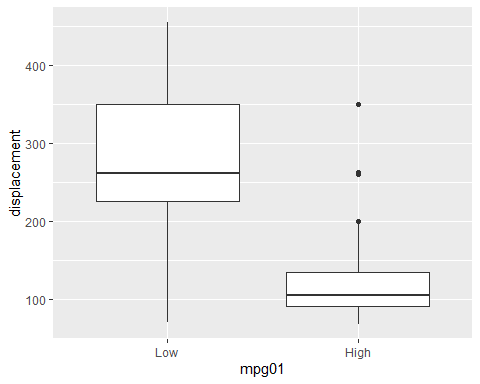
library(tidyverse)  
library(ISLR2)  
Auto <- Auto  
Auto <- Auto |>  
 mutate(mpg01 = factor(if\_else(mpg > median(mpg), 1, 0),   
 levels = c(0, 1),   
 labels = c("Low", "High")))



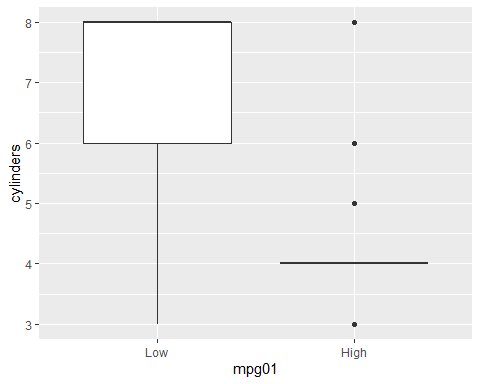
1. Answers will vary. A couple of plots are made below.

ggplot(Auto, aes(y = displacement)) +  
 geom\_boxplot(aes(x = mpg01))



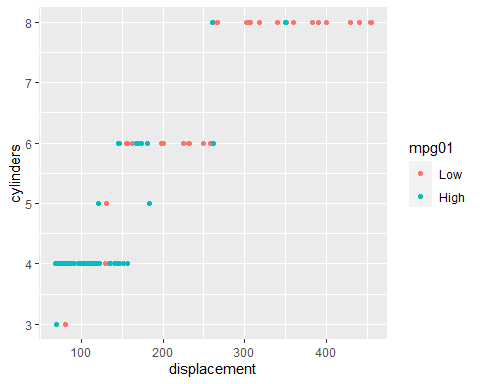


ggplot(Auto, aes(y = cylinders)) +  
 geom\_boxplot(aes(x = mpg01))



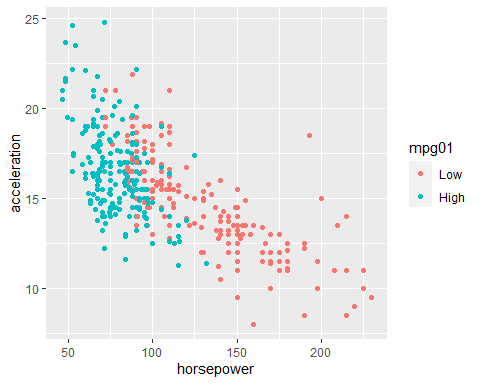
The boxplots show that the lower MPG cars tend to have higher displacement. Similarly, the lower MPG cars tend to have a higher number of cylinders.

ggplot(Auto, aes(x = displacement, y = cylinders, color = mpg01)) +  
 geom\_point()



This scatter plot shows a generally increasing relationship between displacement and cylinders. The points also show that the MPG are generally lower for a higher displacement, higher number of cylinder car.

ggplot(Auto, aes(x = horsepower, y = acceleration, color = mpg01)) +  
 geom\_point()



We see a strong trend that a higher horsepower is associated with lower acceleration. Most of the higher horsepower and lower acceleration points also happen to have lower gas mileage.

(More plots and exploration could be done!)

1. Split the data into a training set and a test set.

library(caret)  
set.seed(11)  
split <- createDataPartition(Auto$mpg01, p = 0.7, list = FALSE)  
train <- Auto[split, ]  
test <- Auto[-split, ]  
tc <- trainControl(method = "cv", number = 10)



1. 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?

#your variables may vary  
LDA\_fit <- train(mpg01 ~ cylinders + weight + acceleration,  
 data = train,   
 method="lda",  
 trControl = tc)  
LDA\_fit$results |>  
 kable()



| parameter | Accuracy | Kappa | AccuracySD | KappaSD |
| --- | --- | --- | --- | --- |
| none | 0.8873016 | 0.7744666 | 0.0614342 | 0.1230138 |

LDA\_conf <- confusionMatrix(test$mpg01, predict(LDA\_fit, test))  
LDA\_conf$table |>  
 kable()

|  | Low | High |
| --- | --- | --- |
| Low | 52 | 6 |
| High | 4 | 54 |

LDA\_conf$overall |>  
 kable()

|  | x |
| --- | --- |
| Accuracy | 0.9137931 |
| Kappa | 0.8275862 |
| AccuracyLower | 0.8471738 |
| AccuracyUpper | 0.9578841 |
| AccuracyNull | 0.5172414 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 0.7518296 |

1. 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?

#your variables may vary  
QDA\_fit <- train(mpg01 ~ cylinders + weight + acceleration,  
 data = train,   
 method="qda",  
 trControl = tc)  
QDA\_fit$results |>  
 kable()



| parameter | Accuracy | Kappa | AccuracySD | KappaSD |
| --- | --- | --- | --- | --- |
| none | 0.8915039 | 0.7828851 | 0.0408137 | 0.081513 |

QDA\_conf <- confusionMatrix(test$mpg01, predict(QDA\_fit, test))  
QDA\_conf$table |>  
 kable()

|  | Low | High |
| --- | --- | --- |
| Low | 53 | 5 |
| High | 6 | 52 |

QDA\_conf$overall |>  
 kable()

|  | x |
| --- | --- |
| Accuracy | 0.9051724 |
| Kappa | 0.8103448 |
| AccuracyLower | 0.8366605 |
| AccuracyUpper | 0.9517059 |
| AccuracyNull | 0.5086207 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 1.0000000 |

1. 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?



#your variables may vary  
LR\_fit <- train(mpg01 ~ (cylinders + weight + acceleration)^2,  
 data = train,   
 method="glm",  
 family = "binomial",  
 trControl = tc)  
LR\_fit$results |>  
 kable()

| parameter | Accuracy | Kappa | AccuracySD | KappaSD |
| --- | --- | --- | --- | --- |
| none | 0.8945462 | 0.7885174 | 0.0440112 | 0.0886492 |

LR\_fit$finalModel$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 42.0576666 |
| cylinders | -2.3340203 |
| weight | -0.0158887 |
| acceleration | -1.0927545 |
| cylinders:weight | 0.0008168 |
| cylinders:acceleration | -0.0267983 |
| weight:acceleration | 0.0004922 |

LR\_conf <- confusionMatrix(test$mpg01, predict(LR\_fit, test))  
LR\_conf$table |>  
 kable()

|  | Low | High |
| --- | --- | --- |
| Low | 54 | 4 |
| High | 8 | 50 |

LR\_conf$overall |>  
 kable()

|  | x |
| --- | --- |
| Accuracy | 0.8965517 |
| Kappa | 0.7931034 |
| AccuracyLower | 0.8262682 |
| AccuracyUpper | 0.9453932 |
| AccuracyNull | 0.5344828 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 0.3864762 |

1. Perform naive Bayes 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?



#your variables may vary  
NB\_fit <- train(mpg01 ~ cylinders + weight + acceleration,  
 data = train,   
 method="nb",  
 trControl = tc)



Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
observation 20

#this warning can safely be ignored  
NB\_fit$results |>  
 kable()

| usekernel | fL | adjust | Accuracy | Kappa | AccuracySD | KappaSD |
| --- | --- | --- | --- | --- | --- | --- |
| FALSE | 0 | 1 | 0.8953500 | 0.7906999 | 0.0539442 | 0.1078419 |
| TRUE | 0 | 1 | 0.8845035 | 0.7690170 | 0.0595697 | 0.1191262 |

NB\_conf <- confusionMatrix(test$mpg01, predict(NB\_fit, test))  
NB\_conf$table |>  
 kable()

|  | Low | High |
| --- | --- | --- |
| Low | 53 | 5 |
| High | 5 | 53 |

NB\_conf$overall |>  
 kable()

|  | x |
| --- | --- |
| Accuracy | 0.9137931 |
| Kappa | 0.8275862 |
| AccuracyLower | 0.8471738 |
| AccuracyUpper | 0.9578841 |
| AccuracyNull | 0.5000000 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 1.0000000 |

1. 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?



#your variables may vary  
kNN\_fit <- train(mpg01 ~ cylinders + weight + acceleration,  
 data = train,   
 method="knn",  
 tuneGrid = data.frame(k = 1:40),  
 trControl = tc)  
kNN\_fit$results |>  
 kable()



| k | Accuracy | Kappa | AccuracySD | KappaSD |
| --- | --- | --- | --- | --- |
| 1 | 0.8222222 | 0.6447771 | 0.0677068 | 0.1350817 |
| 2 | 0.8578042 | 0.7157444 | 0.1003399 | 0.2005956 |
| 3 | 0.8514550 | 0.7029465 | 0.0857559 | 0.1709447 |
| 4 | 0.8548942 | 0.7098756 | 0.0817757 | 0.1629886 |
| 5 | 0.8551587 | 0.7100985 | 0.0774064 | 0.1548165 |
| 6 | 0.8694444 | 0.7390391 | 0.0741739 | 0.1481367 |
| 7 | 0.8731481 | 0.7463954 | 0.0778527 | 0.1554635 |
| 8 | 0.8728836 | 0.7458463 | 0.0574596 | 0.1146439 |
| 9 | 0.8800265 | 0.7602516 | 0.0670659 | 0.1335080 |
| 10 | 0.8800265 | 0.7602516 | 0.0670659 | 0.1335080 |
| 11 | 0.8838624 | 0.7679052 | 0.0658866 | 0.1313358 |
| 12 | 0.8763228 | 0.7526534 | 0.0675431 | 0.1348083 |
| 13 | 0.8837302 | 0.7675275 | 0.0615911 | 0.1227743 |
| 14 | 0.8801587 | 0.7603847 | 0.0597087 | 0.1190011 |
| 15 | 0.8801587 | 0.7603847 | 0.0597087 | 0.1190011 |
| 16 | 0.8800265 | 0.7600500 | 0.0670659 | 0.1337242 |
| 17 | 0.8837302 | 0.7675275 | 0.0615911 | 0.1227743 |
| 18 | 0.8837302 | 0.7675275 | 0.0615911 | 0.1227743 |
| 19 | 0.8801587 | 0.7603847 | 0.0597087 | 0.1190011 |
| 20 | 0.8801587 | 0.7603847 | 0.0597087 | 0.1190011 |
| 21 | 0.8764550 | 0.7529071 | 0.0651162 | 0.1298122 |
| 22 | 0.8764550 | 0.7529071 | 0.0651162 | 0.1298122 |
| 23 | 0.8800265 | 0.7600500 | 0.0626973 | 0.1249583 |
| 24 | 0.8800265 | 0.7600500 | 0.0626973 | 0.1249583 |
| 25 | 0.8800265 | 0.7600500 | 0.0626973 | 0.1249583 |
| 26 | 0.8800265 | 0.7600500 | 0.0626973 | 0.1249583 |
| 27 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 28 | 0.8874339 | 0.7750057 | 0.0528823 | 0.1052712 |
| 29 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 30 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 31 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 32 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 33 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 34 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 35 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 36 | 0.8873016 | 0.7746710 | 0.0587192 | 0.1169615 |
| 37 | 0.8873016 | 0.7746710 | 0.0587192 | 0.1169615 |
| 38 | 0.8873016 | 0.7746710 | 0.0587192 | 0.1169615 |
| 39 | 0.8837302 | 0.7675281 | 0.0594256 | 0.1183846 |
| 40 | 0.8873016 | 0.7746710 | 0.0587192 | 0.1169615 |

kNN\_fit$bestTune

k  
28 28

kNN\_conf <- confusionMatrix(test$mpg01, predict(kNN\_fit, test))  
kNN\_conf$table |>  
 kable()

|  | Low | High |
| --- | --- | --- |
| Low | 51 | 7 |
| High | 8 | 50 |

kNN\_conf$overall |>  
 kable()

|  | x |
| --- | --- |
| Accuracy | 0.8706897 |
| Kappa | 0.7413793 |
| AccuracyLower | 0.7957043 |
| AccuracyUpper | 0.9257840 |
| AccuracyNull | 0.5086207 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 1.0000000 |