

ISL-ch4 Lab and Exrcises.Rmd

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0.9000 --
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     vforcats 0.5.0

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

smarket.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/Smarket.csv", header=T, stringsAsFactors=F)

smarket.df = as_tibble(smarket.df)

str(attributes(smarket.df))

## List of 3
## $ names    : chr [1:10] "X" "Year" "Lag1" "Lag2" ...
## $ row.names: int [1:1250] 1 2 3 4 5 6 7 8 9 10 ...
## $ class    : chr [1:3] "tbl_df" "tbl" "data.frame"
str(smarket.df)

## tibble [1,250 x 10] (S3:tbl_df/tbl/data.frame)
## $ X        : int [1:1250] 1 2 3 4 5 6 7 8 9 10 ...
## $ Year     : int [1:1250] 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 ...
## $ Lag1     : num [1:1250] 0.381 0.959 1.032 -0.623 0.614 ...
## $ Lag2     : num [1:1250] -0.192 0.381 0.959 1.032 -0.623 ...
## $ Lag3     : num [1:1250] -2.624 -0.192 0.381 0.959 1.032 ...
## $ Lag4     : num [1:1250] -1.055 -2.624 -0.192 0.381 0.959 ...
## $ Lag5     : num [1:1250] 5.01 -1.055 -2.624 -0.192 0.381 ...
## $ Volume   : num [1:1250] 1.19 1.3 1.41 1.28 1.21 ...
## $ Today    : num [1:1250] 0.959 1.032 -0.623 0.614 0.213 ...
## $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...

cor(smarket.df[, -10])

##          X      Year      Lag1      Lag2      Lag3
## X 1.00000000 0.97977289 0.035414677 0.036022487 0.038988767
## Year 0.97977289 1.00000000 0.029699649 0.030596422 0.033194581
## Lag1 0.03541468 0.02969965 1.000000000 -0.026294328 -0.010803402
## Lag2 0.03602249 0.03059642 -0.026294328 1.000000000 -0.025896670
## Lag3 0.03898877 0.03319458 -0.010803402 -0.025896670 1.000000000
## Lag4 0.04143714 0.03568872 -0.002985911 -0.010853533 -0.024051036
## Lag5 0.03502515 0.02978799 -0.005674606 -0.003557949 -0.018808338
## Volume 0.54634793 0.53900647 0.040909908 -0.043383215 -0.041823686
## Today 0.03527333 0.03009523 -0.026155045 -0.010250033 -0.002447647
```

```

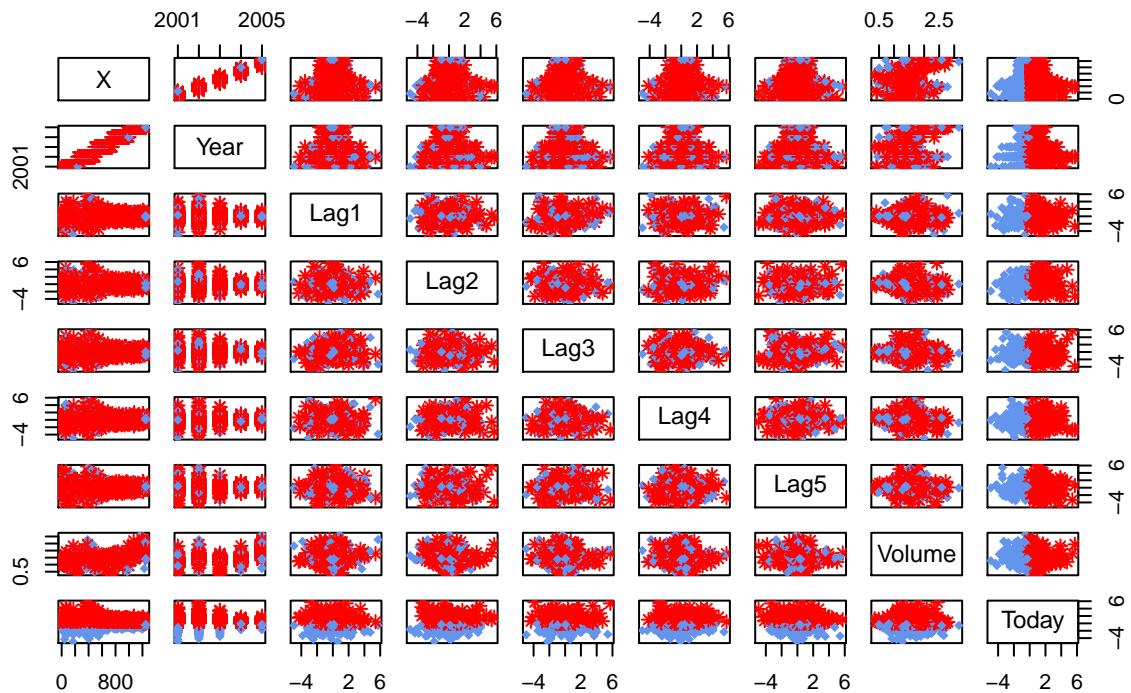
##          Lag4      Lag5      Volume      Today
## X      0.041437137  0.035025152  0.54634793  0.035273325
## Year   0.035688718  0.029787995  0.53900647  0.030095229
## Lag1   -0.002985911 -0.005674606  0.04090991 -0.026155045
## Lag2   -0.010853533 -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.024051036 -0.018808338 -0.04182369 -0.002447647
## Lag4    1.000000000 -0.027083641 -0.04841425 -0.006899527
## Lag5   -0.027083641  1.000000000 -0.02200231 -0.034860083
## Volume -0.048414246 -0.022002315  1.000000000  0.014591823
## Today   -0.006899527 -0.034860083  0.01459182  1.0000000000

group <- NA
group[smarket.df$Direction == "Up"] <- 1
group[smarket.df$Direction == "Down"] <- 2
group[is.na(smarket.df$Direction)] <- 3

pairs(smarket.df[, -10],
      col = c("red", "cornflowerblue", "purple")[group], # Change color by group
      pch = c(8, 18, 1)[group], # Change points by group
      main = "This is an even nicer pairs plot in R")

```

This is an even nicer pairs plot in R



```

# let's do the logistic regression
glm.fit <- glm (Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = smarket.df, family = binomial)
summary(glm.fit)

```

```

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = smarket.df)
##
## Deviance Residuals:

```

```

##      Min       1Q   Median     3Q      Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000  0.240736 -0.523   0.601
## Lag1        -0.073074  0.050167 -1.457   0.145
## Lag2        -0.042301  0.050086 -0.845   0.398
## Lag3         0.011085  0.049939  0.222   0.824
## Lag4         0.009359  0.049974  0.187   0.851
## Lag5         0.010313  0.049511  0.208   0.835
## Volume       0.135441  0.158360  0.855   0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1731.2 on 1249 degrees of freedom
## Residual deviance: 1727.6 on 1243 degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
# contrats shows Direction is set to dummy variable 1 for "Up"
contrasts(smarket.df$Direction)

##      Up
## Down  0
## Up    1
# This basically returns P(Y=1 | X) which means P(Y="Up" | X)
glm.probs <- predict(glm.fit, type="response")

# Since no data is given to predict , model is applied against the training data (i.e smarket.df)
# Thus the order of elements in the vector is the same as order of elements in smarket.df dataframe

# Let's convert probabilities to Up and down
Direction.pred <- rep("Down", nrow(smarket.df))
Direction.pred[glm.probs > 0.5] <- "Up"

#
# now use table to create a confusion matrix using predidted Direction vs training data Direction
(confusionMatrix <- table(Direction.pred, smarket.df$Direction))

##
## Direction.pred  Down   Up
##                 Down 145 141
##                 Up   457 507
# Now we can calculate training error rate
(trainingErrorRate <- (confusionMatrix[1,1] + confusionMatrix[2,2])/nrow(smarket.df))

## [1] 0.5216
(trainingErrorRate2 <- nrow(smarket.df[smarket.df$Direction == Direction.pred, ]) / nrow(smarket.df))

## [1] 0.5216
stopifnot(trainingErrorRate == trainingErrorRate2)

```

```

# We really need to test the model against test data
# Thus we need to train it on some part of the data and test it against some other part

# lets train model only on observations of year >= 2000
smarket.df.2005 <- smarket.df[smarket.df$Year >= 2005, ]
dim(smarket.df.2005) # 252 observations have year >= 2005

## [1] 252 10

# side note
stopifnot(smarket.df.2005$Direction == smarket.df$Direction[smarket.df$Year >= 2005])

Direction.2005 <- smarket.df.2005$Direction

# now we fit the model (using subset) only to observations of year < 2005
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                 data = smarket.df, family = binomial, subset = smarket.df$Year < 2005)

summary(glm.fit)

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = smarket.df, subset = smarket.df$Year <
##     2005)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q      Max
## -1.302   -1.190   1.079   1.160   1.350
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.191213  0.333690  0.573   0.567
## Lag1        -0.054178  0.051785 -1.046   0.295
## Lag2        -0.045805  0.051797 -0.884   0.377
## Lag3         0.007200  0.051644  0.139   0.889
## Lag4         0.006441  0.051706  0.125   0.901
## Lag5        -0.004223  0.051138 -0.083   0.934
## Volume      -0.116257  0.239618 -0.485   0.628
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 1383.3 on 997 degrees of freedom
## Residual deviance: 1381.1 on 991 degrees of freedom
## AIC: 1395.1
##
## Number of Fisher Scoring iterations: 3

# Run the model on test data (i.e observations for which Year >= 2005)
glm.probs <- predict(glm.fit, smarket.df.2005, type="response")
stopifnot(length(glm.probs) == length(Direction.2005))
# Convert thr probabilities into Up and Down using threshold 0.5
glm.pred <- rep("Down", nrow(smarket.df.2005))
glm.pred [glm.probs > .5] = "Up"

```

```



```

```

##      Down   35  35
##      Up     76 106
mean(glm preds == Direction.2005)

## [1] 0.5595238
# test error rate
mean(glm preds != Direction.2005)

## [1] 0.4404762
(FP_rate <- conf_matrix[2,1]/(conf_matrix[2,1] + conf_matrix[1,1]))

## [1] 0.6846847
(TP_rate <- conf_matrix[2,2]/(conf_matrix[2,2] + conf_matrix[1,2]))

## [1] 0.751773
# Now calculate Area under the curve
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##      cov, smooth, var
roc_obj <- roc(Direction.2005, glm.probs)

## Setting levels: control = Down, case = Up
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.5584
# For some reason roc_obj$specificities has extra row
# roc_df <- tibble(
#   TPR=rev(roc_obj$sensitivities),
#   FPR=rev(1 - roc_obj$specificities),
#   labels=roc_obj$response,
#   scores=roc_obj$predictor)

# we can predict based on specefic values of Lag1 nd Lag2
(glm.probs <- predict(glm.fit,tibble(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)) , type = "response"))

##      1        2
## 0.4791462 0.4960939
library(tidyverse)

smarket.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/Smarket.csv", header=T, stringsAsFactors=F)

smarket.df = as_tibble(smarket.df)

```

```

# split to test train
smarket.df.2005 <- smarket.df[smarket.df$Year >= 2005, ]
Direction.2005 <- smarket.df.2005$Direction

# Note we use only Lag1 and Lag2 due to others predictors are even lower in statistically significance
( lda.fit <- MASS::lda(Direction ~ Lag1 + Lag2, data = smarket.df, family = binomial, subset = smarket.df$Year < 2005)

## Call:
## lda(Direction ~ Lag1 + Lag2, data = smarket.df, family = binomial,
##       subset = smarket.df$Year < 2005)
##
## Prior probabilities of groups:
##     Down      Up
## 0.491984 0.508016
##
## Group means:
##           Lag1      Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##           LD1
## Lag1 -0.6420190
## Lag2 -0.5135293

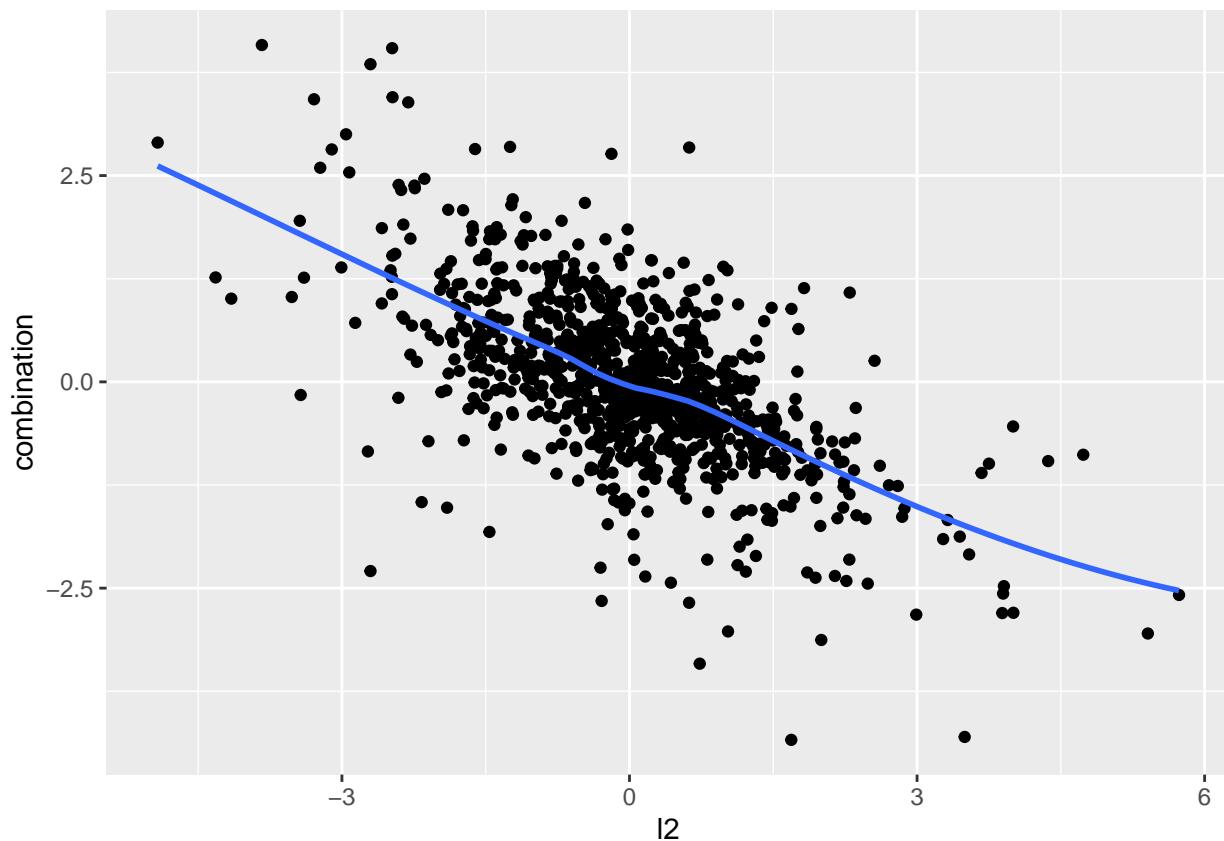
contrasts(smarket.df$Direction)

##     Up
## Down 0
## Up   1

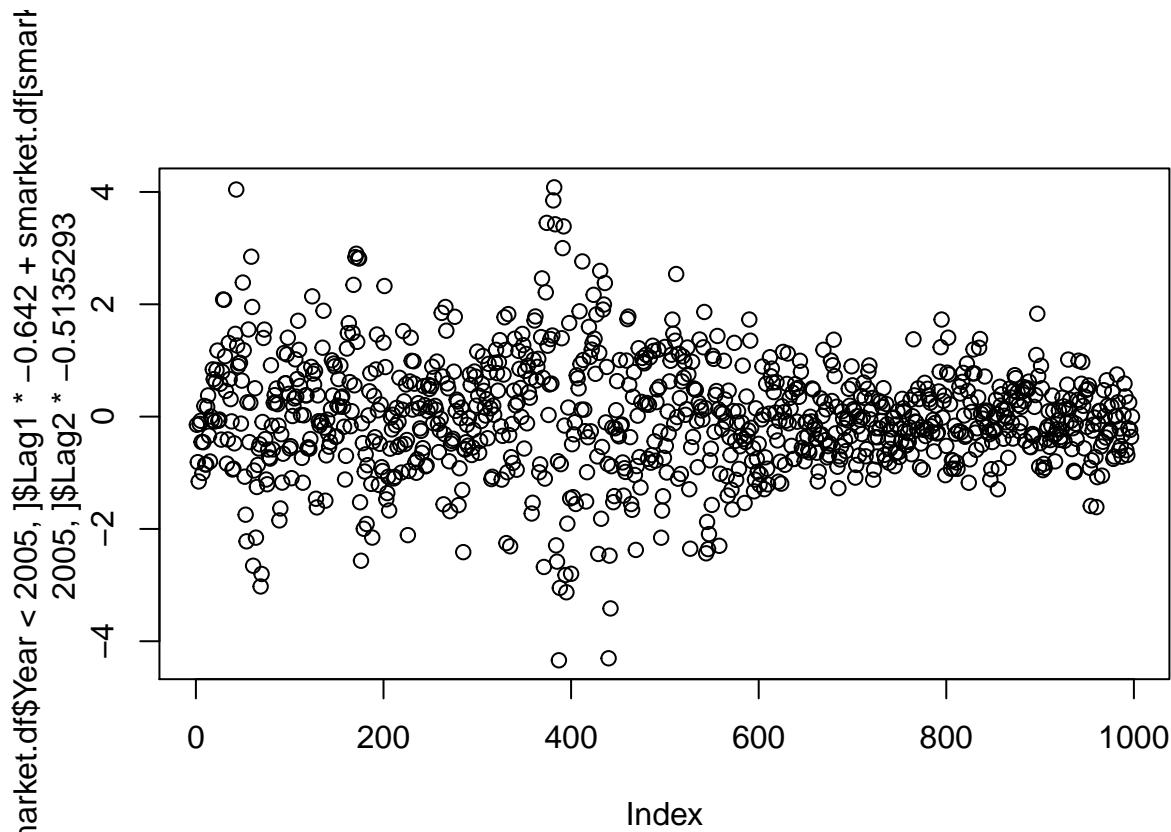
# if Lag1 * -0.642 + Lag2 * -0.5135293 is large then LDA predicts market increases otherwise it decreases
tibble(l1 = smarket.df[smarket.df$Year < 2005, ]$Lag1, l2 = smarket.df[smarket.df$Year < 2005, ]$Lag2, combination = l1 * -0.642 + l2 * -0.5135293) %>%
  ggplot(mapping = aes(x= l2, y = combination)) +
  geom_point() +
  geom_smooth(se=F)

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

```



```
plot(smarket.df[smarket.df$Year < 2005, ]$Lag1 * -0.642 + smarket.df[smarket.df$Year < 2005, ]$Lag2 * -0.001)
```



```

[smarket.df$Year < 2005,]$Lag1 * -0.642 + smarket.df$smart

# lets predict on test data using the model:
lda.pred <- predict(lda.fit, smarket.df.2005)
names(lda.pred)

## [1] "class"      "posterior"   "x"

# first get class ("Up", or "Down") from lda.pred
lda.class <- lda.pred$class
lda.posterior.Down = lda.pred$posterior[,1]
lda.posterior.Up = lda.pred$posterior[,2]

table(lda.class, Direction.2005)

##          Direction.2005
## lda.class Down Up
##       Down    35 35
##       Up     76 106

mean(lda.class == Direction.2005)

## [1] 0.5595238

# recreate the predictions contained in lda.pred$class.
sum(lda.pred$posterior[,1]>=.5)

## [1] 70

sum(lda.pred$posterior[,1]<.5)

## [1] 182

```

```

# posterior probability output by the model corresponds to "Down" probability :
# lda.pred$posterior[1:20,1]
# lda.class[1:20]

(class_prob <- tibble(probability = lda.pred$posterior[,1], cluster = lda.class))

## # A tibble: 252 x 2
##   probability cluster
##       <dbl> <fct>
## 1      0.490 Up
## 2      0.479 Up
## 3      0.467 Up
## 4      0.474 Up
## 5      0.493 Up
## 6      0.494 Up
## 7      0.495 Up
## 8      0.487 Up
## 9      0.491 Up
## 10     0.484 Up
## # ... with 242 more rows

library(pROC)
roc_obj <- roc(Direction.2005, lda.posterior.Up)

## Setting levels: control = Down, case = Up
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.5584

# roc_df <- tibble(
#   TPR=rev(roc_obj$sensitivities),
#   FPR=rev(1 - roc_obj$specificities),
#   labels=roc_obj$response,
#   scores=roc_obj$predictor)

library(tidyverse)

smarket.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/Smarket.csv", header=T, stringsAsFactors=F)

smarket.df = as_tibble(smarket.df)

# split to test train
smarket.df.2005 <- smarket.df[smarket.df$Year >= 2005, ]
Direction.2005 <- smarket.df.2005$Direction

# Note we use only Lag1 and Lag2 due to others predictors are even lower in statistically significance
(qda.fit <- MASS::qda(Direction ~ Lag1 + Lag2, data = smarket.df, family = binomial, subset = smarket.df.2005))

## Call:
## qda(Direction ~ Lag1 + Lag2, data = smarket.df, family = binomial,
##       subset = smarket.df$Year < 2005)
##
## Prior probabilities of groups:
##       Down        Up

```

```

## 0.491984 0.508016
##
## Group means:
##          Lag1      Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
qda.fit

## Call:
## qda(Direction ~ Lag1 + Lag2, data = smarket.df, family = binomial,
##       subset = smarket.df$Year < 2005)
##
## Prior probabilities of groups:
##     Down      Up
## 0.491984 0.508016
##
## Group means:
##          Lag1      Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
contrasts(smarket.df$Direction)

##      Up
## Down  0
## Up    1
qda.pred <- predict(qda.fit, smarket.df.2005)
names(qda.pred)

## [1] "class"      "posterior"
# first get class ("Up", or "Down") from lda.pred
qda.class <- qda.pred$class
qda.posterior.Down = qda.pred$posterior[,1]
qda.posterior.Up = qda.pred$posterior[,2]

table(qda.class, Direction.2005)

##          Direction.2005
## qda.class Down Up
##        Down 30 20
##        Up   81 121
mean(qda.class == Direction.2005)

## [1] 0.5992063
# recreate the predictions contained in lda.pred$class.
sum(qda.pred$posterior[,1]>=.5)

## [1] 50
sum(qda.pred$posterior[,1]<.5)

## [1] 202
(class_prob <- tibble(probability = qda.pred$posterior[,1], cluster = qda.class))

```

```

## # A tibble: 252 x 2
##   probability cluster
##   <dbl> <fct>
## 1 0.487 Up
## 2 0.476 Up
## 3 0.464 Up
## 4 0.474 Up
## 5 0.490 Up
## 6 0.491 Up
## 7 0.492 Up
## 8 0.485 Up
## 9 0.489 Up
## 10 0.482 Up
## # ... with 242 more rows

library(pROC)
roc_obj <- roc(Direction.2005, qda.posterior.Up)

## Setting levels: control = Down, case = Up
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.562

# QDA predictions are accurate almost 60% of the time
# This level of accuracy is quite impressive for stock market data, which is known to be quite hard to predict.
# However, we recommend evaluating this method's performance on a larger test set before betting that it is good.

# knn() forms predictions using a single command
library(class)
library(tidyverse)

smarket.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/Smarket.csv", header=T, stringsAsFactors=F)

smarket.df = as_tibble(smarket.df)

smarket.df.2005 <- smarket.df[smarket.df$Year >= 2005, ]
Direction.2005 <- smarket.df.2005$Direction

# first create train and test data
train.X <- cbind(smarket.df$Lag1,smarket.df$Lag2)[smarket.df$Year < 2005, ]
test.X <- cbind(smarket.df$Lag1,smarket.df$Lag2)[smarket.df$Year >= 2005, ]
train.Direction =smarket.df$Direction [smarket.df$Year < 2005]

# a seed must be set in order to ensure reproducibility of results.
set.seed (1)

# arg1 : matrix containing the predictors associated with the training data
# arg2 : matrix containing the predictors associated with the test data
# arg3 : A vector containing the class labels for the training observations
# arg4 : number of nearest neighbors
knn.pred=knn(train.X,test.X,train.Direction ,k=3)
str(attributes(knn.pred))

## List of 2
## $ levels: chr [1:2] "Down" "Up"

```

```

## $ class : chr "factor"


```

```

train.Y <- caravan.df$Purchase[-test]
test.Y <- caravan.df$Purchase[test]

# now do the KNN
set.seed(1)
knn.pred <- class::knn(train.X, test.X, train.Y, k = 2)
mean(test.Y != "No")

## [1] 0.059
(conf_matrix <- table(knn.pred, test.Y))

##          test.Y
## knn.pred  No Yes
##      No 877 48
##      Yes 64 11
# precision is high and that is what is important in this use case
(precision <- conf_matrix[2,1]/(conf_matrix[2,1] + conf_matrix[2,2]))

## [1] 0.8533333

# Use logistic regression to compare
# Note for Logistic regression we do not need to standardize the predictors

dim(caravan.df[test, ])

## [1] 1000   87

glm.fit <- glm(Purchase ~ MGEMLEEF + PPERSAUT + PLEVEN + PBRAND + PBRAND + ALEVEN + APLEZIER , data = caravan.df)
summary(glm.fit)

##
## Call:
## glm(formula = Purchase ~ MGEMLEEF + PPERSAUT + PLEVEN + PBRAND +
##       PBRAND + ALEVEN + APLEZIER, family = binomial, data = caravan.df,
##       subset = -test)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -1.2472  -0.3909  -0.2658  -0.1912   2.8998
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.38590   0.28110 -15.603 < 2e-16 ***
## MGEMLEEF     0.09825   0.07664   1.282   0.1999
## PPERSAUT     0.24328   0.02614   9.306 < 2e-16 ***
## PLEVEN      -0.20454   0.11759  -1.739   0.0820 .
## PBRAND       0.16694   0.03250   5.137 2.79e-07 ***
## ALEVEN       0.56724   0.23568   2.407   0.0161 *
## APLEZIER     1.92991   0.43595   4.427 9.56e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2187.1  on 4821  degrees of freedom

```

```

## Residual deviance: 2010.2 on 4815 degrees of freedom
## AIC: 2024.2
##
## Number of Fisher Scoring iterations: 6
glm.probs <- predict(glm.fit, caravan.df[test, ], type = "response")
stopifnot(length(glm.probs) == length(test.Y))
glm.pred <- ifelse(glm.probs > 0.05, "Yes", "No")
table(glm.pred, test.Y)

##          test.Y
## glm.pred  No Yes
##      No 500 14
##      Yes 441 45

library(tidyverse)
library(class)
weekly.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/datasets/Weekly.csv",
                     header=T, stringsAsFactors = T, na.strings = "?")
weekly.df = tibble(weekly.df)
#names(weekly.df)
dim(weekly.df)

## [1] 1089     9
#str(weekly.df)

# a) Let's take some summary
summary(weekly.df)

##      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
##      Down:484
##      Up  :605
##
##      #
##      #
##      #
##      #

cor(weekly.df[, -9])

##          Year        Lag1        Lag2        Lag3        Lag4
##  Year   1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
##  Lag1  -0.032289274  1.000000000 -0.07485305  0.05863568 -0.071273876

```

```

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

# percentage of up and down classification:
table(weekly.df$Direction)/sum(table(weekly.df$Direction))

##
## Down Up
## 0.4444444 0.5555556

# The only correlation is between Year and Volume which is 0.84194162
group <- NA
group[weekly.df$Direction == "Up"] <- 1
str(group)

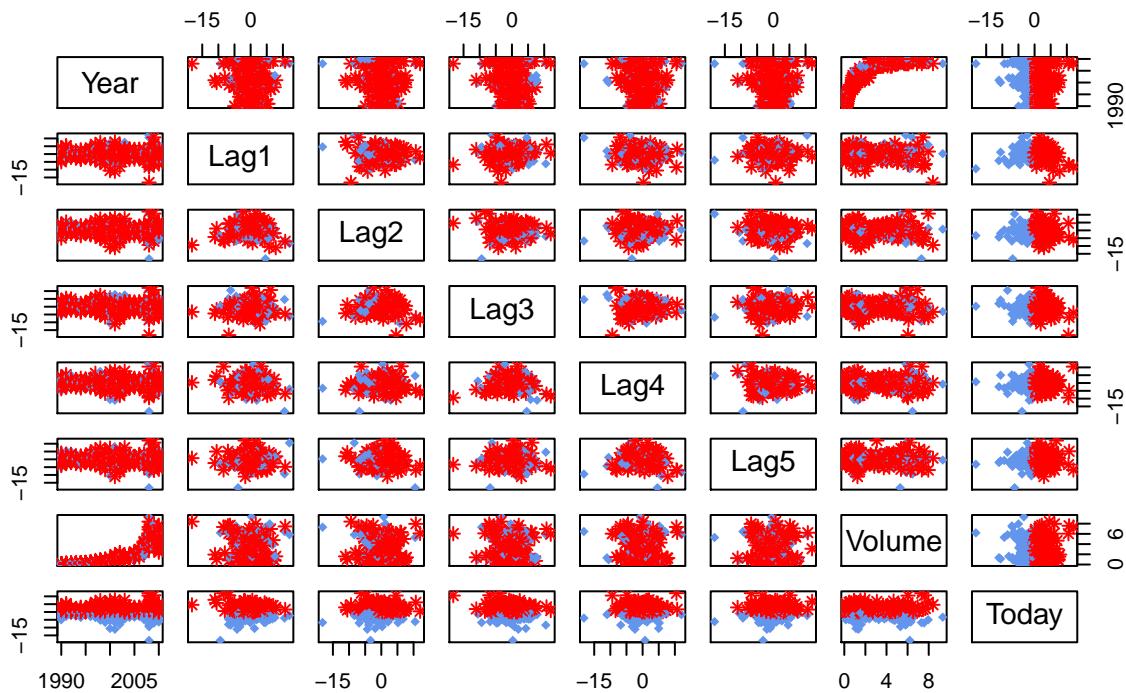
## num [1:1089] NA NA 1 1 1 NA 1 1 1 NA ...
group[weekly.df$Direction == "Down"] <- 2
str(group)

## num [1:1089] 2 2 1 1 1 2 1 1 1 2 ...
group[is.na(weekly.df$Direction)] <- 3

pairs(weekly.df[, -9],
      col = c("red", "cornflowerblue", "purple")[group], # Change color by group
      pch = c(8, 18, 1)[group], # Change points by group
      main = "correlation between oairs of predictors in weekly stock")

```

correlation between pairs of predictors in weekly stock



b) Logistic regression using full data set:

```
glm.fit <- glm(Direction ~ . -Year -Today, data = weekly.df, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ . - Year - Today, family = binomial,
##      data = weekly.df)
##
## 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 is the only predictor that is statistically significant

# How well our model fits depends on the difference between the model and the observed data.
# One approach for binary data is to implement a Hosmer Lemeshow goodness of fit test:

library(ResourceSelection)

## ResourceSelection 0.3-5 2019-07-22
hoslem.test(weekly.df$Direction, fitted(glm.fit))

## Warning in Ops.factor(1, y): '-' not meaningful for factors

##
## Hosmer and Lemeshow goodness of fit (GOF) test
##
## data: weekly.df$Direction, fitted(glm.fit)
## X-squared = 1089, df = 8, p-value < 2.2e-16
# c) confusion matrix

# first lets see the contrasts of Direction to know what is assigned to 1 and which is 2
contrasts(weekly.df$Direction)

##          Up
## Down    0
## Up      1
# Contrasts shows Down is 0 and Up is 1
# Since Posterior is P(Y=1/X) Tuse if posterior > 0.5 it should be "Up"

glm.probs <- predict(glm.fit, weekly.df, type = "response")
glm.predict <- ifelse(glm.probs > 0.5, "Up", "Down")
(confusion_table <- table(glm.predict, weekly.df$Direction))

##
## glm.predict Down Up
##           Down   54  48
##           Up     430 557
# overall fraction of correct predictions:
sprintf("overall fraction of correct predictions: %s", mean(glm.predict == weekly.df$Direction))

## [1] "overall fraction of correct predictions: 0.561065197428834"
# Null classifier:
sprintf("Null classifier : %s ", (48 + 557)/(48 + 557 + 430 + 54) )

## [1] "Null classifier : 0.5555555555555556 "
# FP rate:
sprintf("FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,2]))

## [1] "FP rate (TypeI error, 1 - specificity) : 0.888429752066116"
# TP rate:
sprintf("TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+confusion_table[1,1]))

```

```

## [1] "TP rate (1-TypeII error, power, sensetivity, recall) : 0.920661157024793"
# precision:
sprintf("precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "precision: 0.564336372847011"
# specificity 1-FP/N:
sprintf("specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[1,1]))

## [1] "specificity 1-FP/N: 0.111570247933884"
# d)

train <- (weekly.df$Year >= 1990 & weekly.df$Year <= 2008)
test.Y <- weekly.df[!train,]$Direction
test.X <- weekly.df[!train,]
train.Y <- weekly.df[train, ]$Direction

glm.fit <- glm(Direction ~ Lag2, data = weekly.df, family = binomial, subset = train)
sprintf("summary of logistic regression: ")

## [1] "summary of logistic regression: "
summary(glm.fit)

##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = weekly.df,
##      subset = train)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -1.536   -1.264    1.021    1.091    1.368
##
## 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

glm.probs <- predict(glm.fit, test.X, type = "response")
# again since contrasts(weekly.df$Direction) shows dummy variable 1 assigned to Up
# and since P(y=1/x) is glm.probs what we get is posterior of probability of Up scenario
glm.pred <- ifelse(glm.probs > 0.5, "Up", "Down")

stopifnot(length(glm.pred) == length(test.Y))

```

```

(confusion_table <- table(glm.pred, test.Y))

##           test.Y
## glm.pred Down Up
##      Down    9   5
##      Up     34  56
# Null classifier:
sprintf("Logistic Regression : Null classifier : %s ", (5 + 56)/(5 + 56 + 9 + 34) )

## [1] "Logistic Regression : Null classifier : 0.586538461538462 "
# overall fraction of correct predictions:
sprintf("Logistic Regression : overall fraction of correct predictions: %s", mean(glm.pred == test.Y))

## [1] "Logistic Regression : overall fraction of correct predictions: 0.625"
# FP rate:
sprintf("Logistic Regression : FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,2]))

## [1] "Logistic Regression : FP rate (TypeI error, 1 - specificity) : 0.790697674418605"
# TP rate:
sprintf("Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[1,1]/(confusion_table[1,1]+confusion_table[2,2]))

## [1] "Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : 0.918032786885246"
# precision:
sprintf("Logistic Regression : precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[1,1]))

## [1] "Logistic Regression : precision: 0.6222222222222222"
# specificity 1-FP/N:
sprintf("Logistic Regression : specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,2]))

## [1] "Logistic Regression : specificity 1-FP/N: 0.209302325581395"
library(pROC)
roc_obj <- roc(test.Y, glm.probs)

## Setting levels: control = Down, case = Up
## Setting direction: controls > cases
auc(roc_obj)

## Area under the curve: 0.4537
# -----
# e) repeat part (d) using LDA

lda.fit <- MASS::lda(Direction ~ Lag2, data = weekly.df,family = binomial, subset = train)

# contrasts shows 1 is adssigned to Up
contrasts(weekly.df$Direction)

##           Up
## Down    0
## Up     1

```

```

# lets predict on test data using the model:
lda.pred <- predict(lda.fit, test.X, type = "response")
names(lda.pred)

## [1] "class"      "posterior" "x"

stopifnot(length (lda.pred$class) == length(test.Y))
(confusion_table <- table(lda.pred$class, test.Y))

##          test.Y
##          Down Up
##    Down    9  5
##    Up     34 56

# Null classifier:
sprintf("LDA: Null classifier : %s ", (5 + 56)/(5 + 56 + 9 + 34) )

## [1] "LDA: Null classifier : 0.586538461538462 "

# overall fraction of correct predictions:
sprintf("LDA: overall fraction of correct predictions: %s", mean(lda.pred$class == test.Y))

## [1] "LDA: overall fraction of correct predictions: 0.625"

# FP rate:
sprintf("LDA: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "LDA: FP rate (TypeI error, 1 - specificity) : 0.790697674418605"

# TP rate:
sprintf("LDA: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "LDA: TP rate (1-TypeII error, power, sensetivity, recall) : 0.918032786885246"

# precision:
sprintf("LDA: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1])))

## [1] "LDA: precision: 0.6222222222222222"

# specificity 1-FP/N:
sprintf("LDA: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[2,1]))

## [1] "LDA: specificity 1-FP/N: 0.209302325581395"

# Below is the corresponding posterior probabilities of tests
colnames(lda.pred$posterior)

## [1] "Down" "Up"

lda.pred$posterior[1:10, "Up"]

##          1         2         3         4         5         6         7         8
## 0.5263445 0.6441383 0.4867140 0.4857052 0.5200273 0.5402414 0.6228883 0.4815276
##          9        10
## 0.4519603 0.4853882

lda.pred$posterior[1:10, "Down"]

##          1         2         3         4         5         6         7         8
## 0.4736555 0.3558617 0.5132860 0.5142948 0.4799727 0.4597586 0.3771117 0.5184724
##          9        10
## 0.5480397 0.5146118

```

```

library(pROC)
roc_obj <- roc(test.Y, lda.pred$posterior[, "Up"])

## Setting levels: control = Down, case = Up
## Setting direction: controls > cases
auc(roc_obj)

## Area under the curve: 0.4537
# -----
# f) repeat part (d) using QDA

qda.fit <- MASS::qda(Direction ~ Lag2, data = weekly.df,family = binomial, subset = train)

# contrasts shows 1 is assigned to Up
contrasts(weekly.df$Direction)

##      Up
## Down  0
## Up    1
# lets predict on test data using the model:
qda.pred <- predict(qda.fit, test.X, type = "response")
names(qda.pred)

## [1] "class"      "posterior"
stopifnot(length (qda.pred$class) == length(test.Y))
(confusion_table <- table(qda.pred$class, test.Y))

##      test.Y
##      Down Up
##      Down   0  0
##      Up     43 61
# Null classifier:
sprintf("QDA: Null classifier : %s ", 61/(61+43) )

## [1] "QDA: Null classifier : 0.586538461538462 "
# overall fraction of correct predictions:
sprintf("QDA: overall fraction of correct predictions: %s", mean(qda.pred$class == test.Y))

## [1] "QDA: overall fraction of correct predictions: 0.586538461538462"
# FP rate:
sprintf("QDA: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "QDA: FP rate (TypeI error, 1 - specificity) : 1"
# TP rate:
sprintf("QDA: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "QDA: TP rate (1-TypeII error, power, sensetivity, recall) : 1"
# precision:
sprintf("QDA: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "QDA: precision: 0.586538461538462"

```

```

# specificity 1-FP/N:
sprintf("QDA: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[1,1]))
## [1] "QDA: specificity 1-FP/N: 0"

# Below is the corresponding posterior probabilities of tests
colnames(qda.pred$posterior)

## [1] "Down" "Up"

qda.pred$posterior[1:10, "Up"]

##          1         2         3         4         5         6         7         8
## 0.5215370 0.7306048 0.5264584 0.5270882 0.5197265 0.5290087 0.6714698 0.5299634
##          9        10
## 0.5625713 0.5272914

qda.pred$posterior[1:10, "Down"]

##          1         2         3         4         5         6         7         8
## 0.4784630 0.2693952 0.4735416 0.4729118 0.4802735 0.4709913 0.3285302 0.4700366
##          9        10
## 0.4374287 0.4727086

library(pROC)
roc_obj <- roc(test.Y, qda.pred$posterior[, "Up"])

## Setting levels: control = Down, case = Up
## Setting direction: controls > cases
auc(roc_obj)

## Area under the curve: 0.4914
# -----
# g) repeat part (d) using KNN where K = 1

# First remove NAs

# Now find a subset of records that have at least one NA
# dfSubsetWithNa <- weekly.df[rowSums(is.na(weekly.df)) > 0,]
# head(dfSubsetWithNa)

# # Check a particular column that has NA and include the record
# dfSubsetWithNaInOneCol <- weekly.df[is.na(weekly.df$Directions), ]
# head(dfSubsetWithNaInOneCol)

# get a subset of records in dataframe with no NA in any column:
# dfSubsetWithNoNa <- weekly.df[rowSums(is.na(weekly.df)) == 0, ]
# head(dfSubsetWithNoNa)

# To use KNN first we need to standardize the predictor Lag2

standardized.weekly <- scale(weekly.df[, c("Lag2")])
names(standardized.weekly)

## NULL

```

```

# now split it into to train and test
# First make a dataframe
helperDf <- tibble(Year = weekly.df$Year, Direction = weekly.df$Direction
                    , Lag2 = standardized.weekly[,1])
head(helperDf)

## # A tibble: 6 x 3
##   Year Direction Lag2
##   <int>    <fct>   <dbl>
## 1 1990 Down     0.603
## 2 1990 Down     0.282
## 3 1990 Up      -0.179
## 4 1990 Up      -1.16
## 5 1990 Up      1.43
## 6 1990 Down     0.238

# Let's break helper into test and train

train.X <- helperDf[train, "Lag2"]
train.Y <- helperDf[train, ]$Direction
test.X <- helperDf[!train, "Lag2"]
test.Y <- helperDf[!train, ]$Direction

# now do the KNN
set.seed(1)
knn.pred <- class::knn(train.X, test.X, train.Y, k = 1)

(confusion_table <- table(knn.pred, test.Y))

##          test.Y
## knn.pred Down Up
##        Down 21 31
##        Up    22 30
stopifnot(length(knn.pred) == length(test.Y))

# Null classifier:
sprintf("KNN: Null classifier : %s ", (31+30)/(21+22+31+30) )

## [1] "KNN: Null classifier : 0.586538461538462 "
# overall fraction of correct predictions:
sprintf("KNN: overall fraction of correct predictions: %s", mean(knn.pred == test.Y))

## [1] "KNN: overall fraction of correct predictions: 0.490384615384615"
# FP rate:
sprintf("KNN: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "KNN: FP rate (TypeI error, 1 - specificity) : 0.511627906976744"
# TP rate:
sprintf("KNN: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "KNN: TP rate (1-TypeII error, power, sensetivity, recall) : 0.491803278688525"
# precision:
sprintf("KNN: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))
```

```

## [1] "KNN: precision: 0.576923076923077"
# specificity 1-FP/N:
sprintf("KNN: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[1,1]))
## [1] "KNN: specificity 1-FP/N: 0.488372093023256"
print ("LDA and QDA are best classifiers with 62% of overall correct prediction.")

## [1] "LDA and QDA are best classifiers with 62% of overall correct prediction."
library(tidyverse)
library(class)
car.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/datasets/Auto.csv",
                  header=T, stringsAsFactors = T, na.strings = "?")
car.df = tibble(car.df)
dim(car.df)

## [1] 392   9

# a)
car.df <- car.df %>%
  mutate(mpg01 = ifelse (mpg > median(mpg), 1, 0))
colnames(car.df)

## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"        "mpg01"

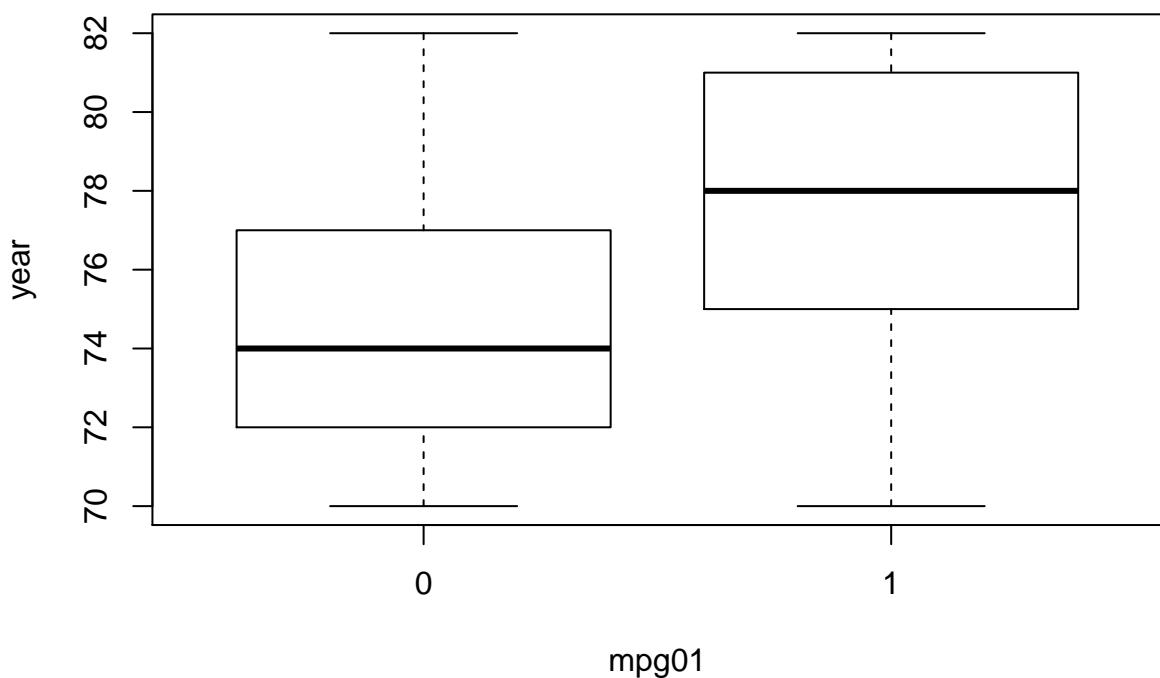
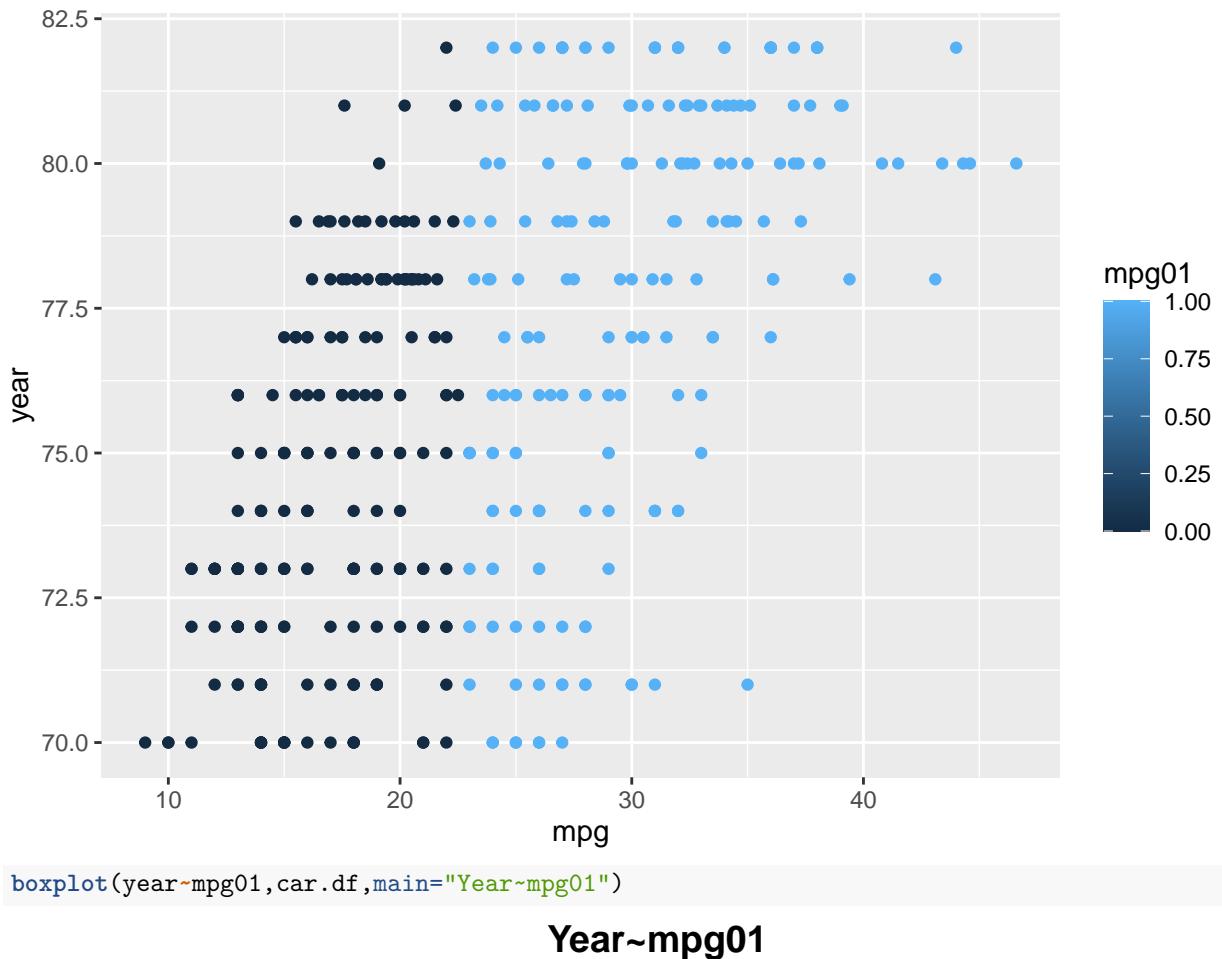
median(car.df$mpg)

## [1] 22.75

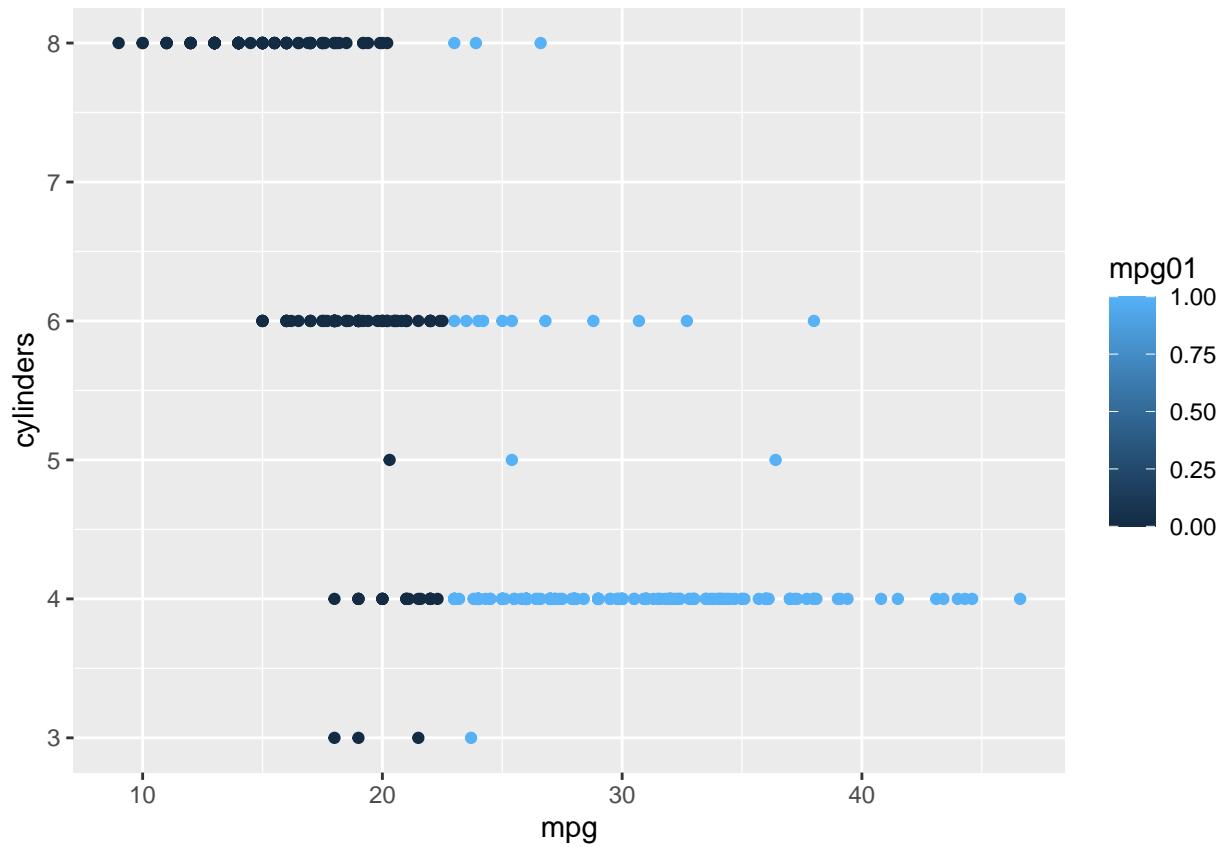
# b) Explore the data graphically

# group <- NA
# group[car.df$mpg01 == "0"] <- 1
# group[car.df$mpg01 == "1"] <- 2
#
# pairs(car.df[, c("mpg01", "year")],
#        col = c("red", "cornflowerblue", "purple")[group], # Change color by group
#        pch = c(8, 18, 1)[group], # Change points by group
#        main = "correlation between mpg01 and other predictors ")
#
# car.df %>%
#   ggplot() +
#   geom_point(mapping = aes(x = mpg, y = year, color = mpg01))

```

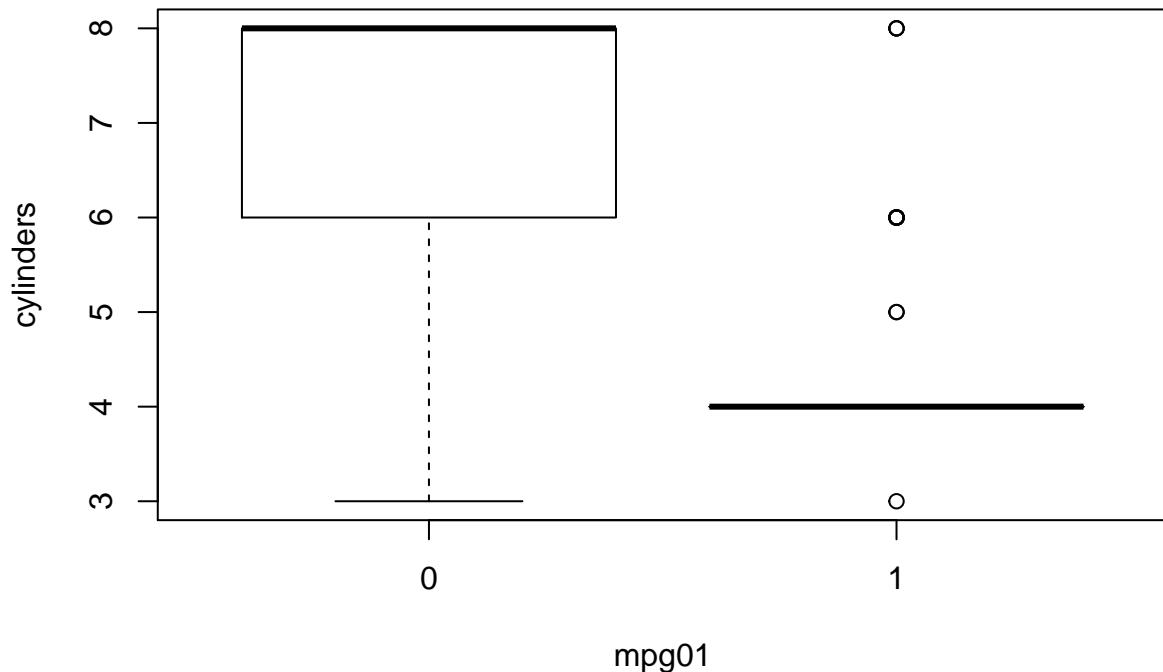


```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = cylinders, color = mpg01))
```

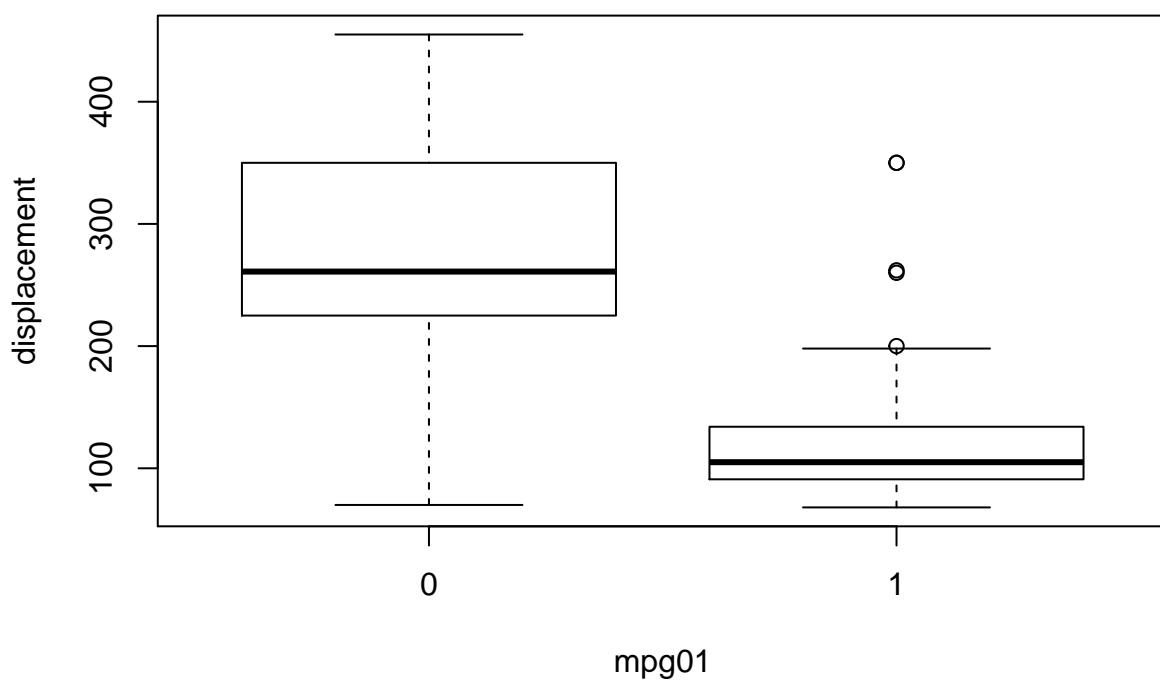
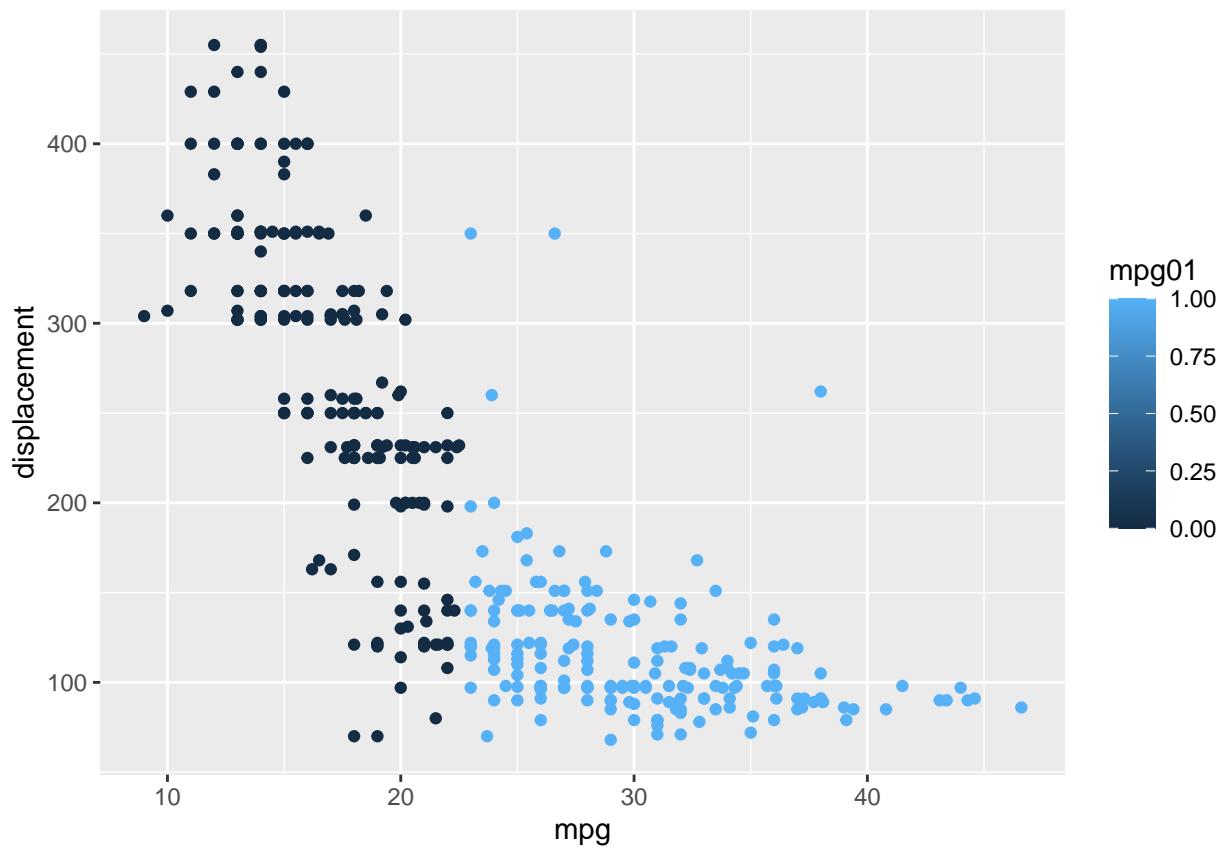


```
boxplot(cylinders~mpg01, car.df, main="cylinders~mpg01")
```

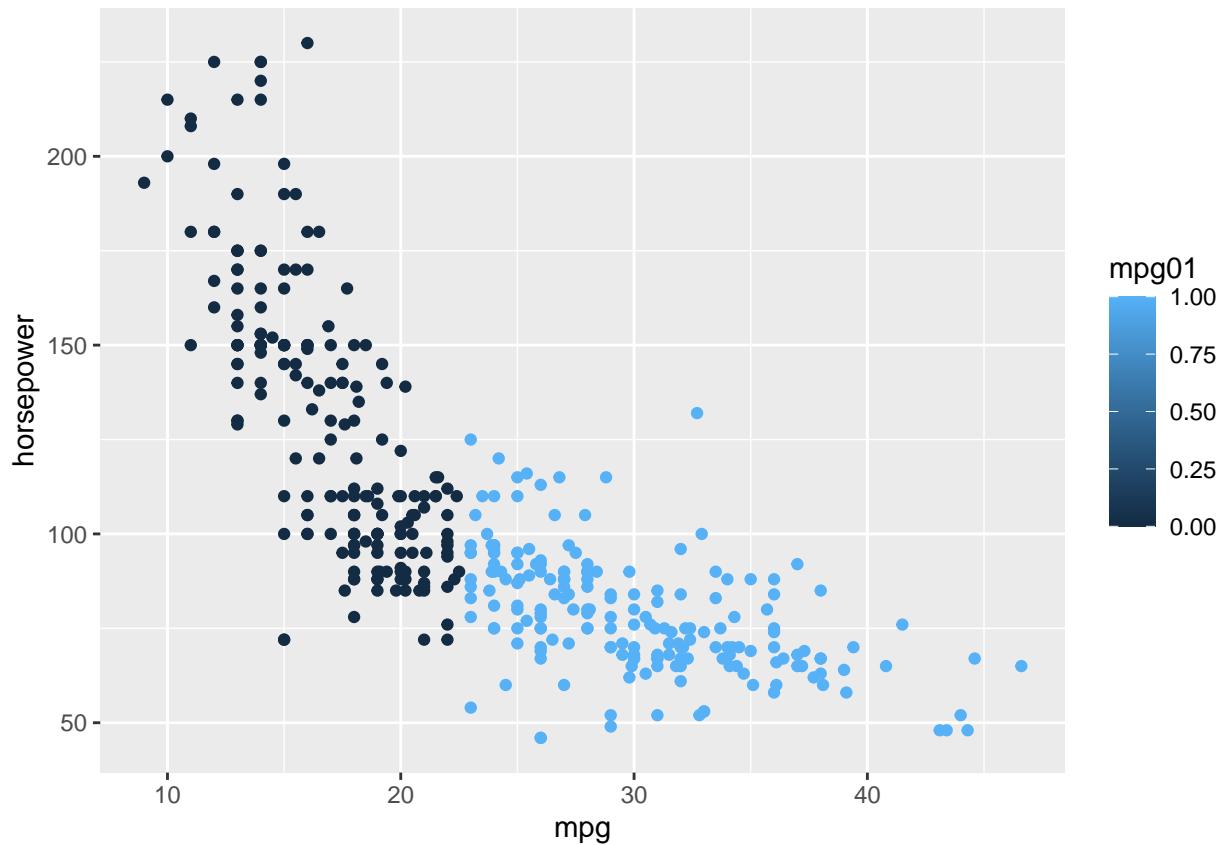
cylinders~mpg01



```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = displacement, color = mpg01))
```

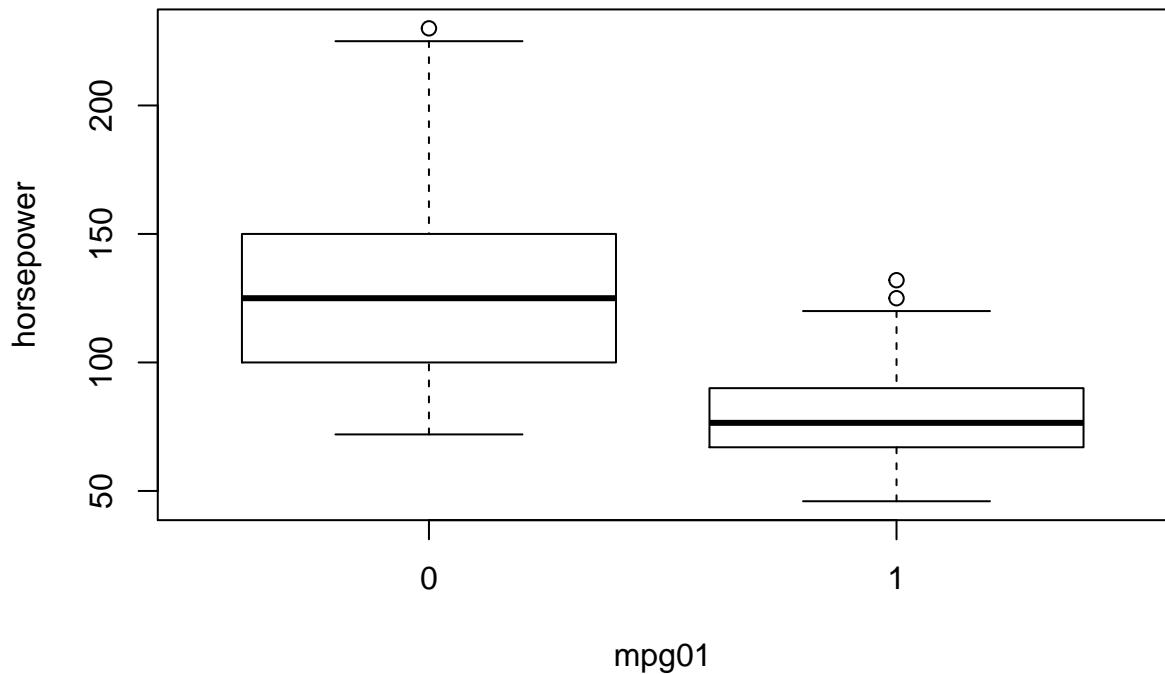


```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = horsepower, color = mpg01))
```

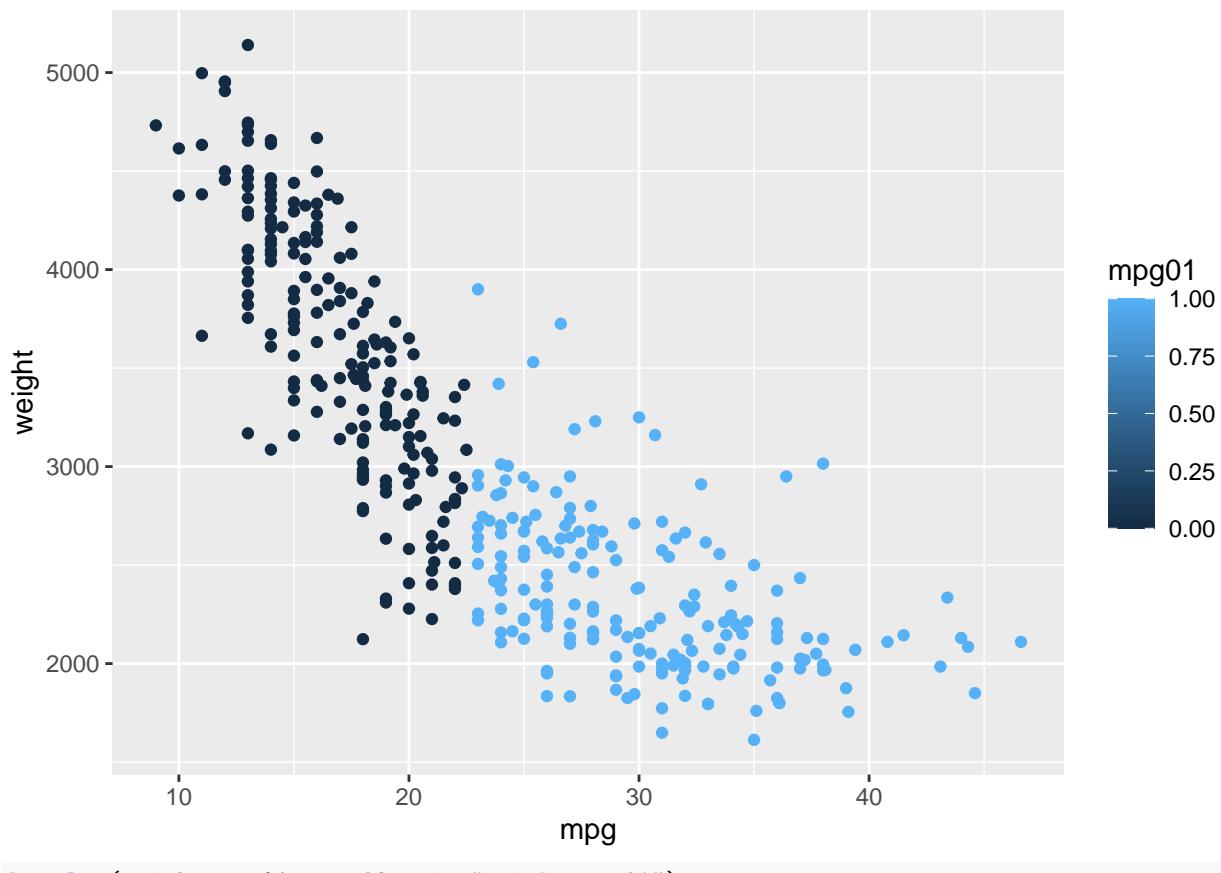


```
boxplot(horsepower~mpg01, car.df, main="horsepower~mpg01")
```

horsepower~mpg01

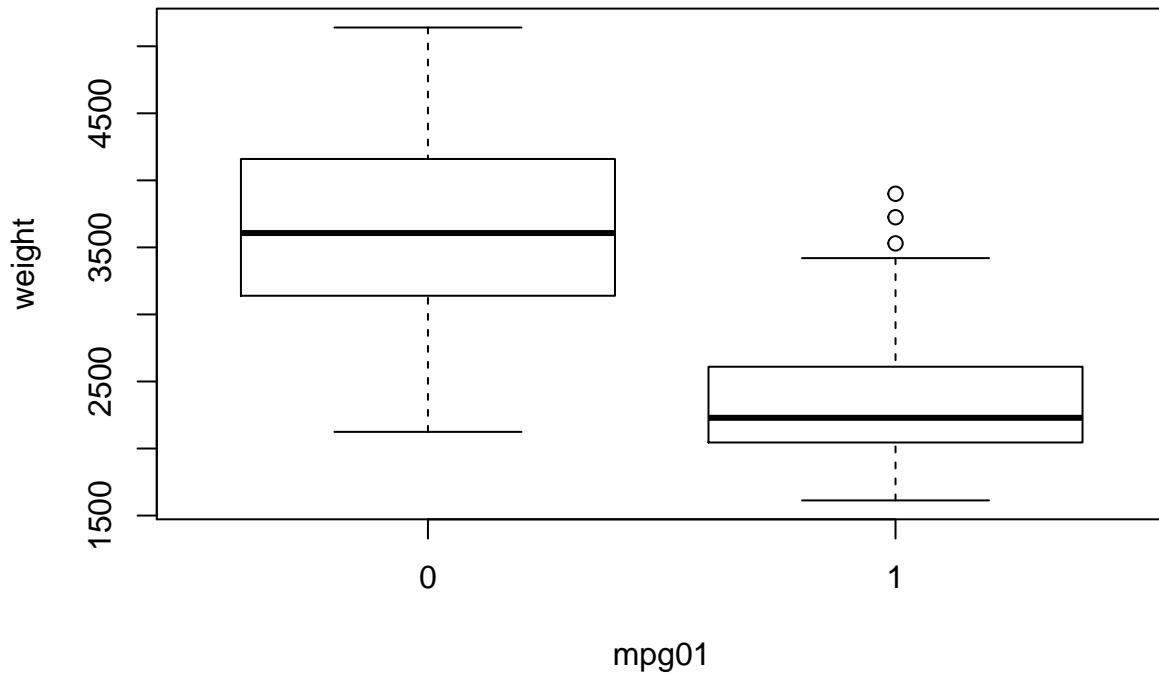


```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = weight, color = mpg01))
```

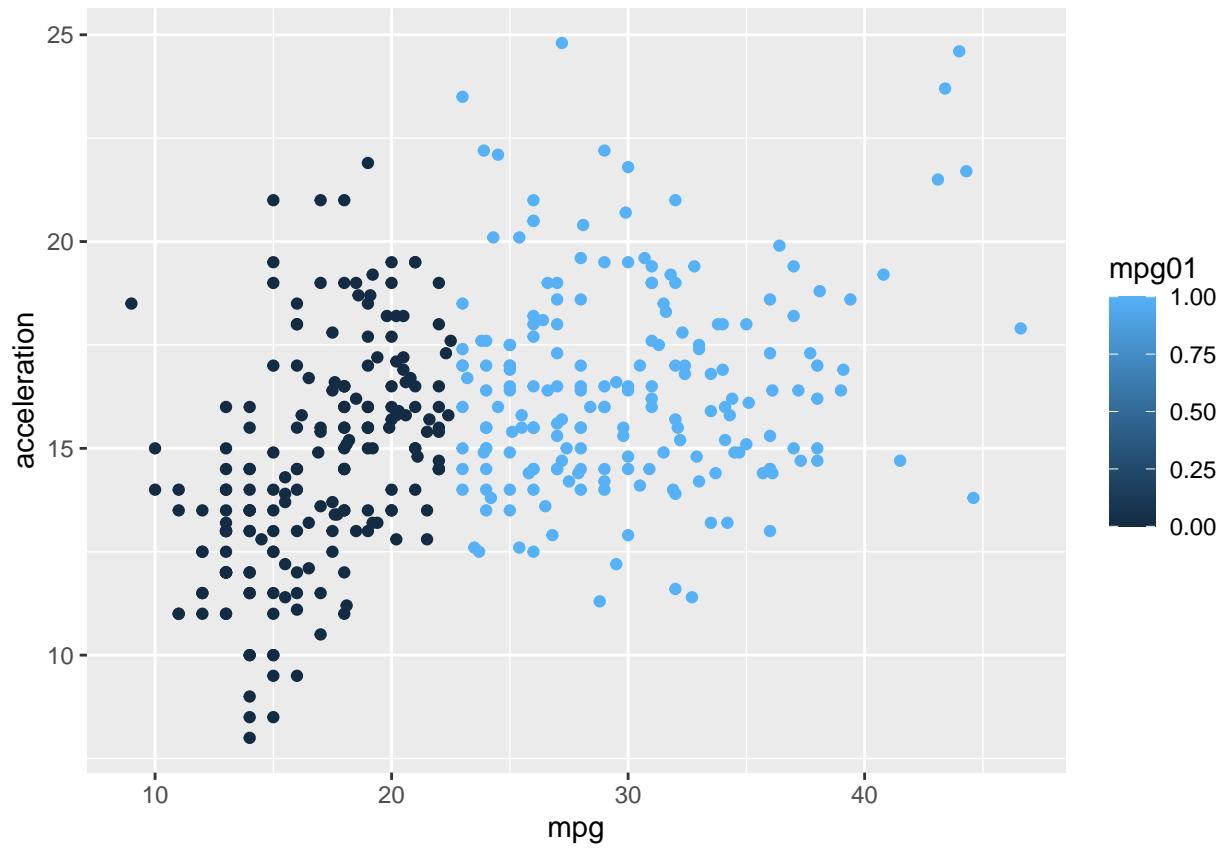


```
boxplot(weight~mpg01,car.df,main="weight~mpg01")
```

weight~mpg01

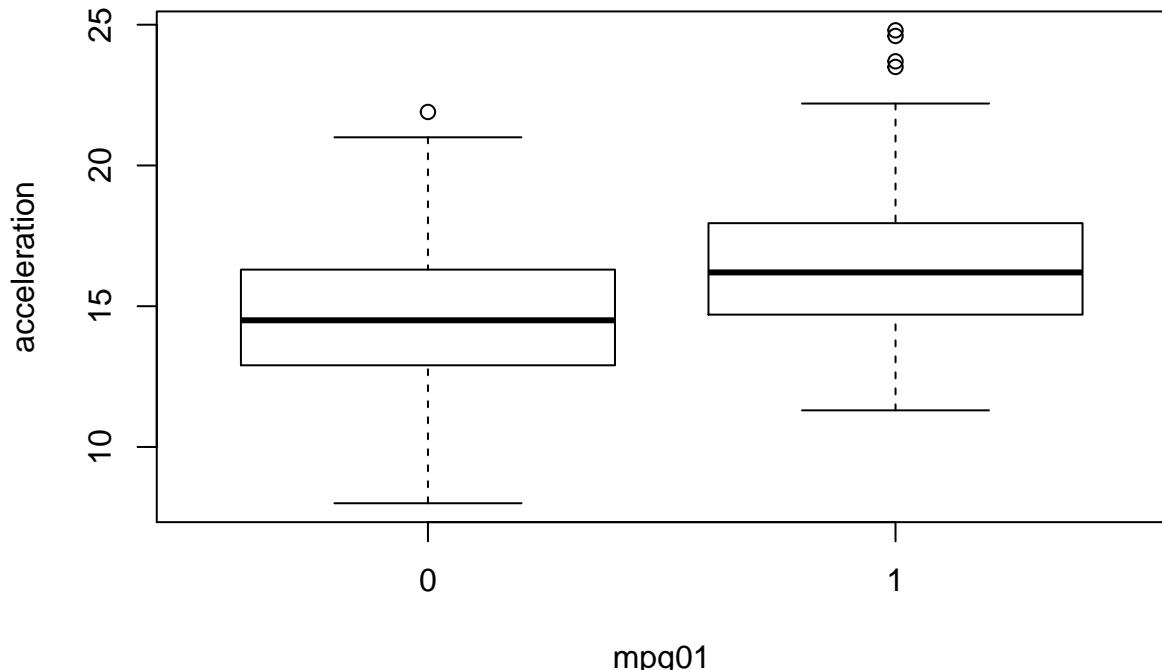


```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = acceleration, color = mpg01))
```

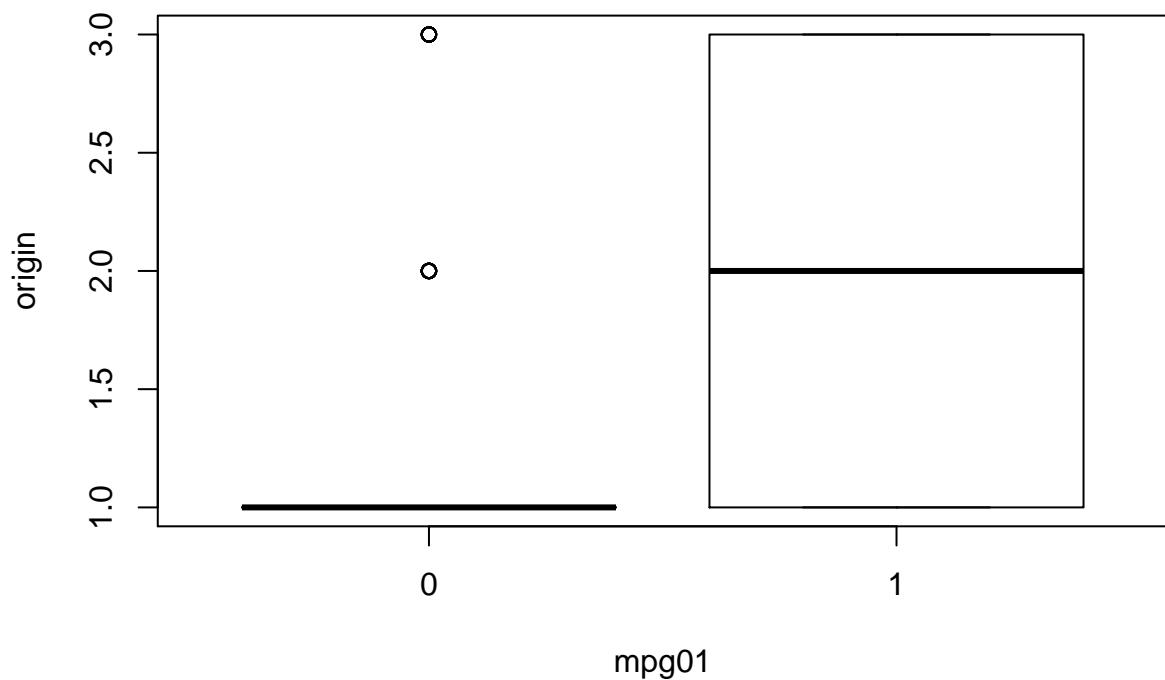
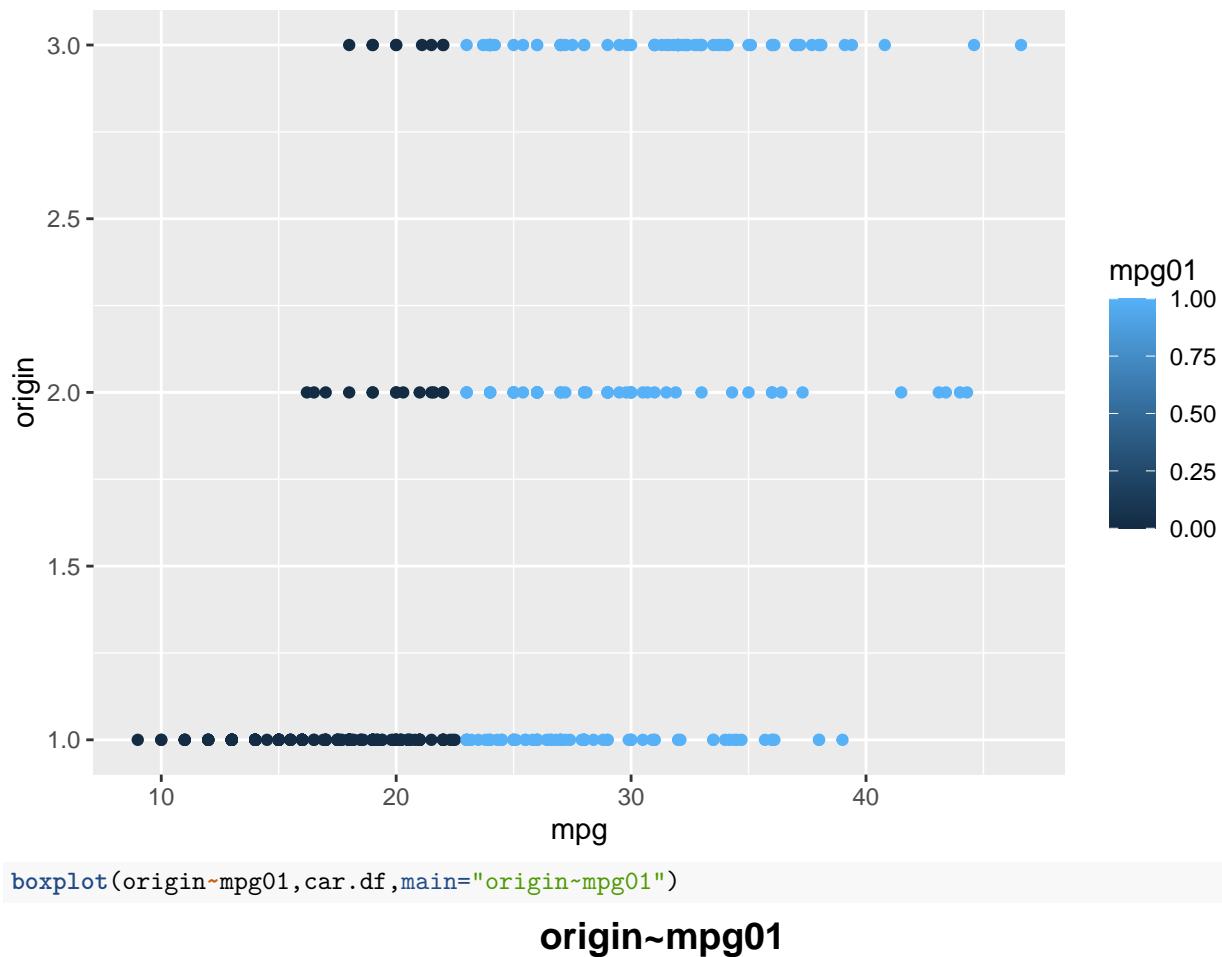


```
boxplot(acceleration~mpg01, car.df, main="acceleration~mpg01")
```

acceleration~mpg01



```
car.df %>%
  ggplot() +
  geom_point(mapping = aes(x = mpg, y = origin, color = mpg01))
```



```

#
# car.df %>%
#   filter(mpg01 == 0) %>%
#   ggplot(mapping = aes(x = mpg01, y = year)) +
#   geom_boxplot()
#
#
# car.df %>%
#   filter(mpg01 != 0) %>%
#   ggplot(mapping = aes(x = mpg01, y = year)) +
#   geom_boxplot()
#
#
# ggplot(data = car.df) +
#   geom_point(mapping = aes(x = mpg01, y = year, size = cylinders))

sprintf("Features : year, cylinders, displacement, horsepower, weight, origin")

## [1] "Features : year, cylinders, displacement, horsepower, weight, origin"
# c) Split the data into training and test

# car.df %>%
#   group_by(year) %>%
#   summarize(n())

train <- (car.df$year <= 78)
# train.Y <- car.df[train, "mpg01"]
# test.Y <- car.df[!train, "mpg01"]
train.Y <- car.df[train, ]$mpg01
test.Y <- car.df[!train, ]$mpg01
test.X <- car.df[!train, ]

# d) Perform LDA
lda.fit <- MASS:::lda(mpg01 ~ year + cylinders + displacement + horsepower + weight + origin, data = car)

# lets predict on test data using the model:
lda.pred <- predict(lda.fit, test.X, type = "response")

stopifnot(length(lda.pred$class) == length(test.Y))
(confusion_table <- table(lda.pred$class, test.Y))

##      test.Y
##      0 1
## 0 17 13
## 1  1 83
#
# Null classifier:
sprintf("LDA: Null classifier : %s ", (13+83)/(17+1+13+83) )

## [1] "LDA: Null classifier : 0.842105263157895 "
#
# overall fraction of correct predictions:
sprintf("LDA: overall fraction of correct predictions: %s", mean(lda.pred$class == test.Y))

## [1] "LDA: overall fraction of correct predictions: 0.87719298245614"

```

```

# FP rate:
sprintf("LDA: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "LDA: FP rate (TypeI error, 1 - specificity) : 0.055555555555556"

# TP rate:
sprintf("LDA: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "LDA: TP rate (1-TypeII error, power, sensetivity, recall) : 0.864583333333333"

# precision:
sprintf("LDA: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "LDA: precision: 0.988095238095238"

# specificity 1-FP/N:
sprintf("LDA: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[2,2]))

## [1] "LDA: specificity 1-FP/N: 0.944444444444444"

sprintf("test Error is about %s", mean(lda.pred$class != test.Y))

## [1] "test Error is about 0.12280701754386"

# e) Perform QDA
qda.fit <- MASS::qda(mpg01 ~ year + cylinders + displacement + horsepower + weight + origin, data = car)

# lets predict on test data using the model:
qda.pred <- predict(qda.fit, test.X, type = "response")

stopifnot(length(qda.pred$class) == length(test.Y))
(confusion_table <- table(qda.pred$class, test.Y))

##      test.Y
##      0 1
## 0 17 19
## 1 1 77

# Null classifier:
sprintf("QDA: Null classifier : %s ",(77+19)/(77+19+1+17) )

## [1] "QDA: Null classifier : 0.842105263157895 "

# overall fraction of correct predictions:
sprintf("QDA: overall fraction of correct predictions: %s", mean(qda.pred$class == test.Y))

## [1] "QDA: overall fraction of correct predictions: 0.824561403508772"

# FP rate:
sprintf("QDA: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "QDA: FP rate (TypeI error, 1 - specificity) : 0.055555555555556"

# TP rate:
sprintf("QDA: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "QDA: TP rate (1-TypeII error, power, sensetivity, recall) : 0.802083333333333"

# precision:
sprintf("QDA: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "QDA: precision: 0.987179487179487"

```

```

# specificity 1-FP/N:
sprintf("QDA: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[1,1]+ confusion_table[2,2]+ confusion_table[1,2]))
## [1] "QDA: specificity 1-FP/N: 0.9444444444444444"

## [1] "test Error is about %s", mean(qda.pred$class != test.Y))

## [1] "test Error is about 0.175438596491228"

# f) Perform Logistic regression
glm.fit <- glm(mpg01 ~ year + cylinders + displacement + horsepower + weight + origin, data = car.df, family = binomial)
sprintf("summary of logistic regression: ")
## [1] "summary of logistic regression: "
summary(glm.fit)

## 
## Call:
## glm(formula = mpg01 ~ year + cylinders + displacement + horsepower +
##       weight + origin, family = binomial, data = car.df, subset = train)
## 

## Deviance Residuals:
##      Min        1Q     Median        3Q        Max
## -2.4231   -0.1160   -0.0028    0.1674    2.2930
## 

## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.583967  7.769287 -0.075  0.94008
## year         0.241226  0.111557  2.162  0.03059 *
## cylinders   -0.454012  0.636865 -0.713  0.47592
## displacement -0.006523  0.017002 -0.384  0.70123
## horsepower   -0.042541  0.025010 -1.701  0.08894 .
## weight        -0.003957  0.001326 -2.985  0.00283 **
## origin        -0.030567  0.423935 -0.072  0.94252
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 

## (Dispersion parameter for binomial family taken to be 1)
## 

## Null deviance: 363.21  on 277  degrees of freedom
## Residual deviance: 101.47  on 271  degrees of freedom
## AIC: 115.47
## 

## Number of Fisher Scoring iterations: 8

glm.probs <- predict(glm.fit, test.X, type = "response")

glm.pred <- ifelse(glm.probs > 0.5, 1, 0)

stopifnot(length(glm.pred) == length(test.Y))

(confusion_table <- table(glm.pred, test.Y))

## 
## test.Y
## glm.pred  0   1

```

```

##      0 17 14
##      1  1 82
# Null classifier:
sprintf("Logistic Regression : Null classifier : %s ", (14+82)/(14+82+1+17) )

## [1] "Logistic Regression : Null classifier : 0.842105263157895 "
# overall fraction of correct predictions:
sprintf("Logistic Regression : overall fraction of correct predictions: %s", mean(glm.pred == test.Y))

## [1] "Logistic Regression : overall fraction of correct predictions: 0.868421052631579"
# FP rate:
sprintf("Logistic Regression : FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,1]))

## [1] "Logistic Regression : FP rate (TypeI error, 1 - specificity) : 0.0555555555555556"
# TP rate:
sprintf("Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[1,1]/(confusion_table[1,1]+confusion_table[2,2]))

## [1] "Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : 0.854166666666667"
# precision:
sprintf("Logistic Regression : precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[1,1]+confusion_table[2,1]))

## [1] "Logistic Regression : precision: 0.987951807228916"
# specificity 1-FP/N:
sprintf("Logistic Regression : specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,1]))

## [1] "Logistic Regression : specificity 1-FP/N: 0.9444444444444444"
sprintf("Logistic Regression : Test Error Rate: %s", mean(glm.pred != test.Y))

## [1] "Logistic Regression : Test Error Rate: 0.131578947368421"
library(pROC)
roc_obj <- roc(test.Y, glm.probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.9624
# g) Perform KNN
# First scale the data
str(car.df)

## tibble [392 x 10] (S3: tbl_df/tbl/data.frame)
## $ mpg      : num [1:392] 18 15 18 16 17 15 14 14 14 15 ...
## $ cylinders : int [1:392] 8 8 8 8 8 8 8 8 8 ...
## $ displacement: num [1:392] 307 350 318 304 302 429 454 440 455 390 ...
## $ horsepower : int [1:392] 130 165 150 150 140 198 220 215 225 190 ...
## $ weight     : int [1:392] 3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
## $ acceleration: num [1:392] 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
## $ year       : int [1:392] 70 70 70 70 70 70 70 70 70 70 ...
## $ origin     : int [1:392] 1 1 1 1 1 1 1 1 1 1 ...
## $ name       : Factor w/ 301 levels "amc ambassador brougham",...: 49 36 230 14 160 141 54 222 240 ...

```

```

## $ mpg01      : num [1:392] 0 0 0 0 0 0 0 0 0 ...
scaled.car.df <- scale(car.df[, -9])
colnames(scaled.car.df)

## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"        "origin"       "mpg01"

(helperDF <- tibble(year = car.df$year, mpg01 = car.df$mpg01 , cylinders = scaled.car.df[, "cylinders"]
                     displacement = scaled.car.df[, "displacement"], horsepower = scaled.car.df[, "horsepower"]
                     weight = scaled.car.df[, "weight"], origin = scaled.car.df[, "origin"]))

## # A tibble: 392 x 7
##   year mpg01 cylinders displacement horsepower weight origin
##   <int> <dbl>     <dbl>      <dbl>      <dbl>   <dbl>   <dbl>
## 1    70 0.0000000 1.4800000 1.0800000 0.6630000 0.6200000 -0.716
## 2    70 0.0000000 1.4800000 1.4900000 1.5700000 0.8420000 -0.716
## 3    70 0.0000000 1.4800000 1.1800000 1.1800000 0.5400000 -0.716
## 4    70 0.0000000 1.4800000 1.0500000 1.1800000 0.5360000 -0.716
## 5    70 0.0000000 1.4800000 1.0300000 0.9230000 0.5550000 -0.716
## 6    70 0.0000000 1.4800000 2.2400000 2.4300000 1.6100000 -0.716
## 7    70 0.0000000 1.4800000 2.4800000 3.0000000 1.6200000 -0.716
## 8    70 0.0000000 1.4800000 2.3500000 2.8700000 1.5700000 -0.716
## 9    70 0.0000000 1.4800000 2.4900000 3.1300000 1.7000000 -0.716
## 10   70 0.0000000 1.4800000 1.8700000 2.2200000 1.0300000 -0.716
## # ... with 382 more rows

train.X <- helperDF[train, c("cylinders", "displacement", "horsepower", "weight", "origin")]
train.Y <- helperDF[train, ]$mpg01
test.X <- helperDF[!train, c("cylinders", "displacement", "horsepower", "weight", "origin")]
test.Y <- helperDF[!train, ]$mpg01

# now do the KNN
set.seed(1)
knn.pred <- class::knn(train.X, test.X, train.Y, k = 15)

(confusion_table <- table(knn.pred, test.Y))

##           test.Y
## knn.pred  0  1
##            0 17 14
##            1  1 82
stopifnot(length(knn.pred) == length(test.Y))

# Null classifier:
sprintf("KNN: Null classifier : %s ", (24+72)/(24+72+17+1) )

## [1] "KNN: Null classifier : 0.842105263157895 "
# overall fraction of correct predictions:
sprintf("KNN: overall fraction of correct predictions: %s", mean(knn.pred == test.Y))

## [1] "KNN: overall fraction of correct predictions: 0.868421052631579"
# FP rate:
sprintf("KNN: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "KNN: FP rate (TypeI error, 1 - specificity) : 0.0555555555555556"

```

```

# TP rate:
sprintf("KNN: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+confusion_table[1,2]+confusion_table[2,1]+confusion_table[1,1]))

## [1] "KNN: TP rate (1-TypeII error, power, sensetivity, recall) : 0.854166666666667"

# precision:
sprintf("KNN: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]+confusion_table[1,1]))

## [1] "KNN: precision: 0.987951807228916"

# specificity 1-FP/N:
sprintf("KNN: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,1]))

## [1] "KNN: specificity 1-FP/N: 0.944444444444444"

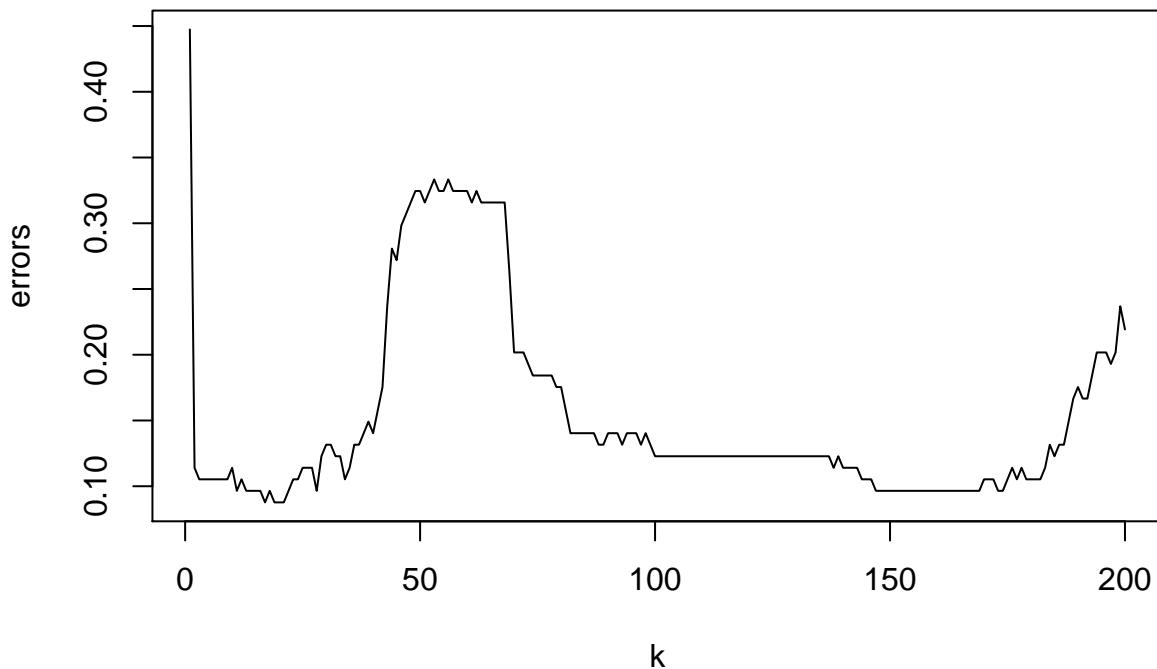
# test error rate
sprintf("KNN: Test error rate: %s", mean(knn.pred != test.Y))

## [1] "KNN: Test error rate: 0.131578947368421"

# accumulate errors:
errors <- c()
maxK <- 200
step <- 1
for(k in seq(1,maxK,step)){
  knn.pred <- class::knn(train.X, test.X, train.Y, k)
  (confusion_table <- table (knn.pred, test.Y))
  errors <- c(mean(knn.pred != test.Y), errors)
}

data <- cbind(seq(1,maxK,step),errors)
plot(data,type="l",xlab="k")

```



```

Power <- function(){
  2 ^ 3
}

Power()
## [1] 8
# b)
Power2 <- function(x, a) { x^a }
Power2(2,3)
## [1] 8
# c)
Power2(10,3)
## [1] 1000
Power2(8,17)
## [1] 2.2518e+15
Power2(131,3)
## [1] 2248091
# d)
Power3 <- function(x, a){
  result <- x^a
  return (result)
}

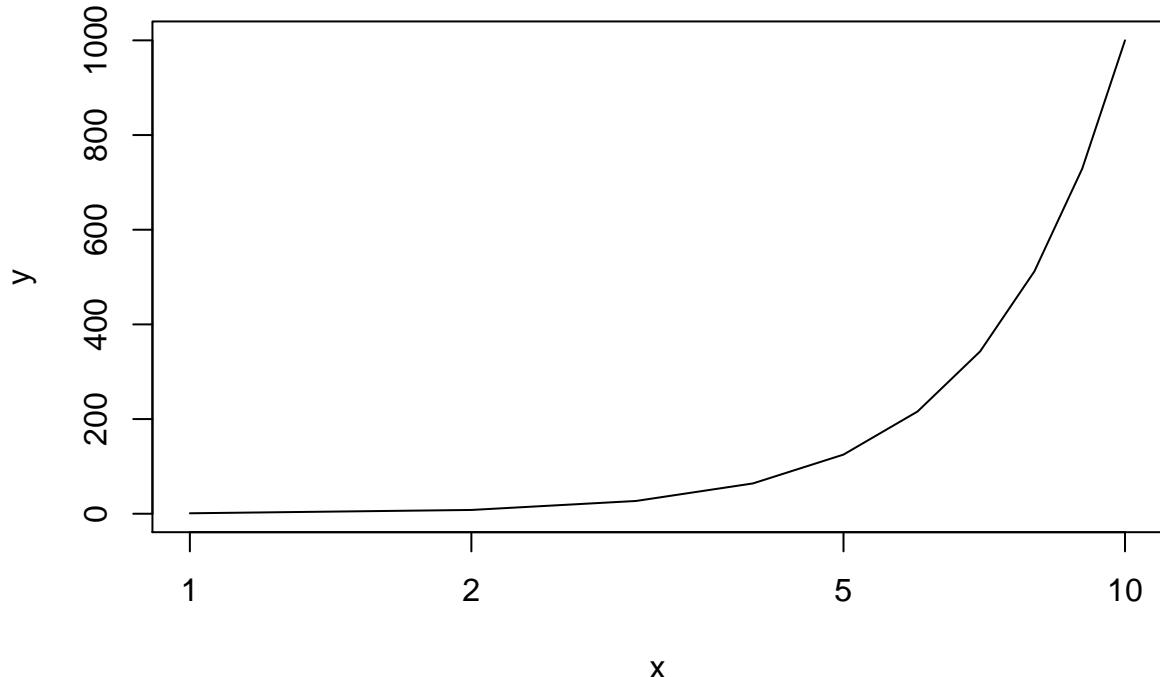
Power3(2,3)
## [1] 8
# e)

x = 1:10
y = sapply(x,function(x){Power3(x,3)})

plot(x,y,log='x',main='x ~ x^2',type='l')

```

$x \sim x^2$



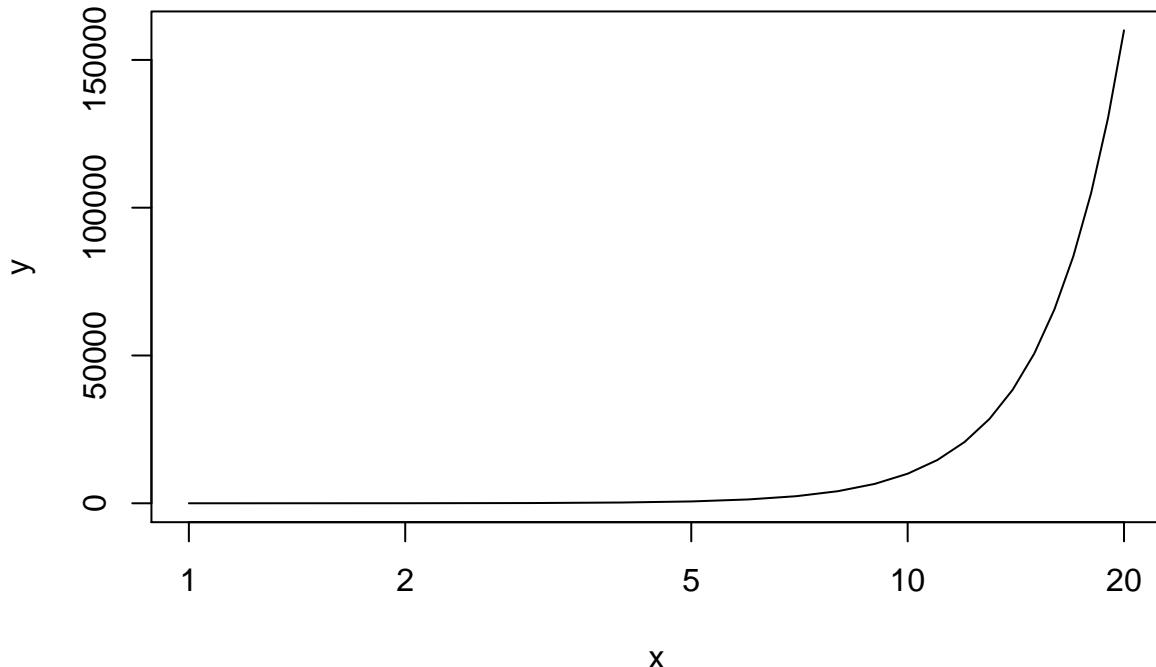
```
# f)
PlotPower <- function(x, a){

  y = sapply(x,function(x){Power3(x,a)})

  plot(x,y,log='x',main='x ~ x^a',type='l')

}
PlotPower(1:20,4)
```

$$x \sim x^a$$



```

library(tidyverse)
library(class)
boston.df = read.csv("/Users/shahrdadshadab/env/my-R-project/ISLR/Data/datasets/BostonHousing.csv",
                     header=T, stringsAsFactors = T, na.strings = "?")
boston.df = tibble(boston.df)
dim(boston.df)

## [1] 506 14
names(boston.df)

## [1] "crim"      "zn"        "indus"     "chas"      "nox"       "rm"        "age"
## [8] "dis"        "rad"       "tax"        "ptratio"   "b"         "lstat"     "medv"

boston.df <- boston.df %>%
  mutate (crim01 = ifelse (crim >= median(crim), 1, 0))

# split the data into test and train
train <- (boston.df$age <= 90)
test.X <- boston.df[!train,]
test.Y <- boston.df[!train,]$crim01
train.Y <- boston.df[train,]$crim01

# first use Logistic regression to calculate P(crime01=1 | X):
glm.fit <- glm(crim01 ~ . , data = boston.df, family = binomial, subset = train)

## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
sprintf("summary of logistic regression: ")

## [1] "summary of logistic regression: "

```

```

summary(glm.fit)

##
## Call:
## glm(formula = crim01 ~ ., family = binomial, data = boston.df,
##      subset = train)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -1.277e-03 -2.000e-08 -2.000e-08  2.000e-08  1.306e-03
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.541e+02  1.207e+05 -0.005   0.996
## crim         1.077e+03  3.057e+04  0.035   0.972
## zn            2.079e+00  3.823e+02  0.005   0.996
## indus        -2.053e+00  1.537e+03 -0.001   0.999
## chas          -6.263e+01  2.256e+04 -0.003   0.998
## nox           1.120e+02  1.715e+05  0.001   0.999
## rm            4.531e+01  9.435e+03  0.005   0.996
## age           1.654e-01  7.844e+01  0.002   0.998
## dis            -8.349e-01  3.747e+03  0.000   1.000
## rad            1.379e+00  5.268e+03  0.000   1.000
## tax            -3.621e-02  2.026e+02  0.000   1.000
## ptratio        2.826e+00  2.921e+03  0.001   0.999
## b              2.910e-02  3.482e+01  0.001   0.999
## lstat          1.904e+00  4.980e+02  0.004   0.997
## medv          -1.369e+00  8.238e+02 -0.002   0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 4.2502e+02 on 337 degrees of freedom
## Residual deviance: 5.5710e-06 on 323 degrees of freedom
## AIC: 30
##
## Number of Fisher Scoring iterations: 25
glm.probs <- predict(glm.fit, test.X, type = "response")

glm.pred <- ifelse(glm.probs > 0.5, 1, 0)

stopifnot(length(glm.pred) == length(test.Y))

(confusion_table <- table(glm.pred, test.Y))

##
## test.Y
## glm.pred  0   1
##          0 21   1
##          1   3 143
#
# Null classifier:
sprintf("Logistic Regression : Null classifier : %s ", (1+143)/(1+143+21+3) )

## [1] "Logistic Regression : Null classifier : 0.857142857142857 "

```

```

# overall fraction of correct predictions:
sprintf("Logistic Regression : overall fraction of correct predictions: %s", mean(glm.pred == test.Y))

## [1] "Logistic Regression : overall fraction of correct predictions: 0.976190476190476"
# FP rate:
sprintf("Logistic Regression : FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,1]))

## [1] "Logistic Regression : FP rate (TypeI error, 1 - specificity) : 0.125"
# TP rate:
sprintf("Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[1,1]/(confusion_table[1,1]+confusion_table[0,1]))

## [1] "Logistic Regression : TP rate (1-TypeII error, power, sensetivity, recall) : 0.993055555555556"
# precision:
sprintf("Logistic Regression : precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[0,2]))

## [1] "Logistic Regression : precision: 0.979452054794521"
# specificity 1-FP/N:
sprintf("Logistic Regression : specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[1,1]))

## [1] "Logistic Regression : specificity 1-FP/N: 0.875"
sprintf("Logistic Regression : Test Error Rate: %s", mean(glm.pred != test.Y))

## [1] "Logistic Regression : Test Error Rate: 0.0238095238095238"
library(pROC)
roc_obj <- roc(test.Y, glm.probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.9988

# Now lets do the LDA
lda.fit <- MASS:::lda(crim01 ~ ., data = boston.df, family = binomial, subset = train)

# lets predict on test data using the model:
lda.pred <- predict(lda.fit, test.X, type = "response")

stopifnot(length(lda.pred$class) == length(test.Y))
(confusion_table <- table(lda.pred$class, test.Y))

##      test.Y
##      0   1
## 0 24 36
## 1  0 108

# Null classifier:
sprintf("LDA: Null classifier : %s ", (36+108)/(36+108+24+0) )

## [1] "LDA: Null classifier : 0.857142857142857 "
# overall fraction of correct predictions:
sprintf("LDA: overall fraction of correct predictions: %s", mean(lda.pred$class == test.Y))

```

```

## [1] "LDA: overall fraction of correct predictions: 0.785714285714286"
# FP rate:
sprintf("LDA: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "LDA: FP rate (TypeI error, 1 - specificity) : 0"
# TP rate:
sprintf("LDA: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "LDA: TP rate (1-TypeII error, power, sensetivity, recall) : 0.75"
# precision:
sprintf("LDA: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "LDA: precision: 1"
# specificity 1-FP/N:
sprintf("LDA: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+confusion_table[2,2]))

## [1] "LDA: specificity 1-FP/N: 1"
sprintf("test Error is about %s", mean(lda.pred$class != test.Y))

## [1] "test Error is about 0.214285714285714"
library(pROC)
roc_obj <- roc(test.Y, glm.probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
auc(roc_obj)

## Area under the curve: 0.9988
# Now lets do the KNN

# First scale the predictors
scaled_boston <- scale(boston.df)
colnames(scaled_boston)

## [1] "crim"      "zn"        "indus"     "chas"      "nox"       "rm"        "age"
## [8] "dis"        "rad"       "tax"        "ptratio"   "b"         "lstat"     "medv"
## [15] "crim01"

helperDF <- tibble(age = boston.df$age, crim01 = boston.df$crim01,
                     zn = scaled_boston[, "zn"], indus = scaled_boston[, "indus"], chas = scaled_boston[, "chas"],
                     nox = scaled_boston[, "nox"], rm = scaled_boston[, "rm"], medv = scaled_boston[, "medv"],
                     dis = scaled_boston[, "dis"], rad = scaled_boston[, "rad"], tax = scaled_boston[, "tax"],
                     ptratio = scaled_boston[, "ptratio"], b = scaled_boston[, "b"], lstat = scaled_boston[, "lstat"])

train.X <- helperDF[train, c("zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat")]
train.Y <- helperDF[train, ]$crim01
test.X <- helperDF[!train, c("zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat")]
test.Y <- helperDF[!train, ]$crim01

set.seed(1)
knn.pred <- class::knn(train.X, test.X, train.Y, k = 85)

(confusion_table <- table(knn.pred, test.Y))

```

```

##           test.Y
## knn.pred   0   1
##           0   0   0
##           1  24 144
stopifnot(length(knn.pred) == length(test.Y))

# Null classifier:
sprintf("KNN: Null classifier : %s ", (144)/(144+24) )

## [1] "KNN: Null classifier : 0.857142857142857 "
# overall fraction of correct predictions:
sprintf("KNN: overall fraction of correct predictions: %s", mean(knn.pred == test.Y))

## [1] "KNN: overall fraction of correct predictions: 0.857142857142857"
# FP rate:
sprintf("KNN: FP rate (TypeI error, 1 - specificity) : %s", confusion_table[2,1]/(confusion_table[2,1]+

## [1] "KNN: FP rate (TypeI error, 1 - specificity) : 1"
# TP rate:
sprintf("KNN: TP rate (1-TypeII error, power, sensetivity, recall) : %s", confusion_table[2,2]/(confusion_table[2,2]+

## [1] "KNN: TP rate (1-TypeII error, power, sensetivity, recall) : 1"
# precision:
sprintf("KNN: precision: %s", confusion_table[2,2] / (confusion_table[2,2] + confusion_table[2,1]))

## [1] "KNN: precision: 0.857142857142857"
# specificity 1-FP/N:
sprintf("KNN: specificity 1-FP/N: %s", 1 - confusion_table[2,1]/(confusion_table[2,1]+ confusion_table[

## [1] "KNN: specificity 1-FP/N: 0"
# test error rate
sprintf("KNN: Test error rate: %s", mean(knn.pred != test.Y))

## [1] "KNN: Test error rate: 0.142857142857143"
# accumulate errors:
errors <- c()
maxK <- 100
step <- 1
for(k in seq(1,maxK,step)){
  knn.pred <- class::knn(train.X, test.X, train.Y, k)
  (confusion_table <- table(knn.pred, test.Y))
  errors <- c(mean(knn.pred != test.Y), errors)
}

data <- cbind(seq(1,maxK,step),errors)
plot(data,type="l",xlab="k")

```

