SDS 323 – HW 3

Ben Howell

Question 14 (A, B, C, F, H)

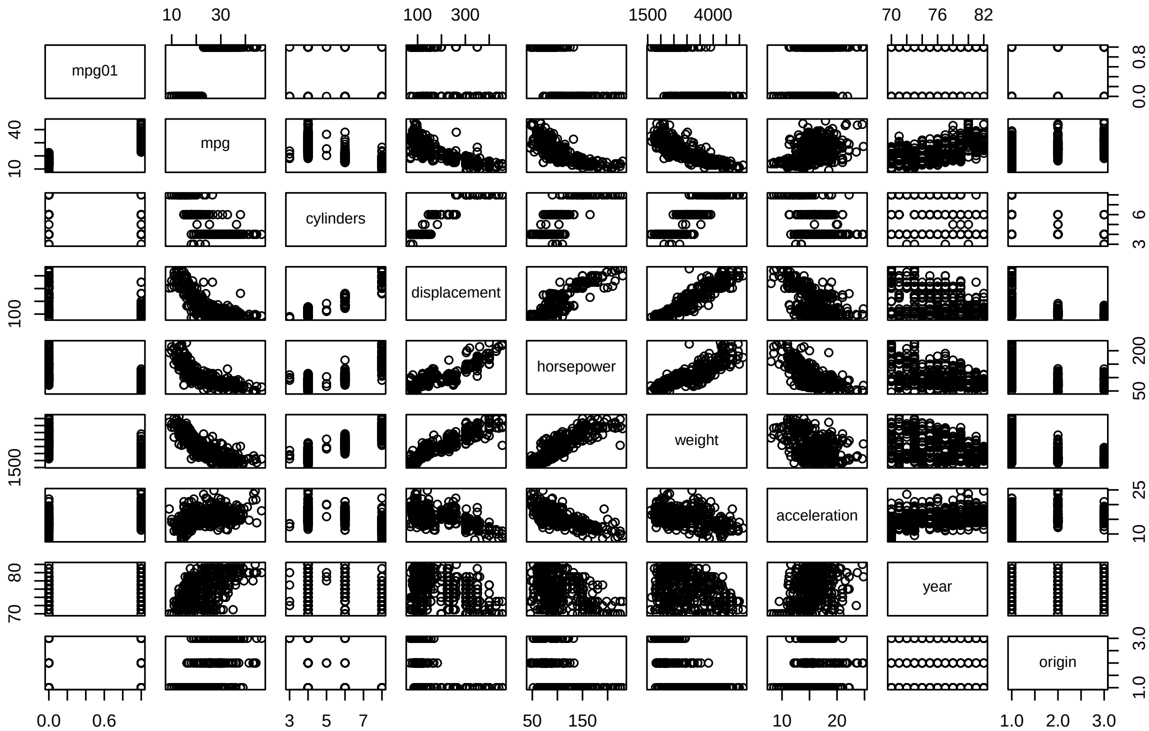
```
require(tidyverse)
require(ISLR2)
```

```
head(Auto)
```

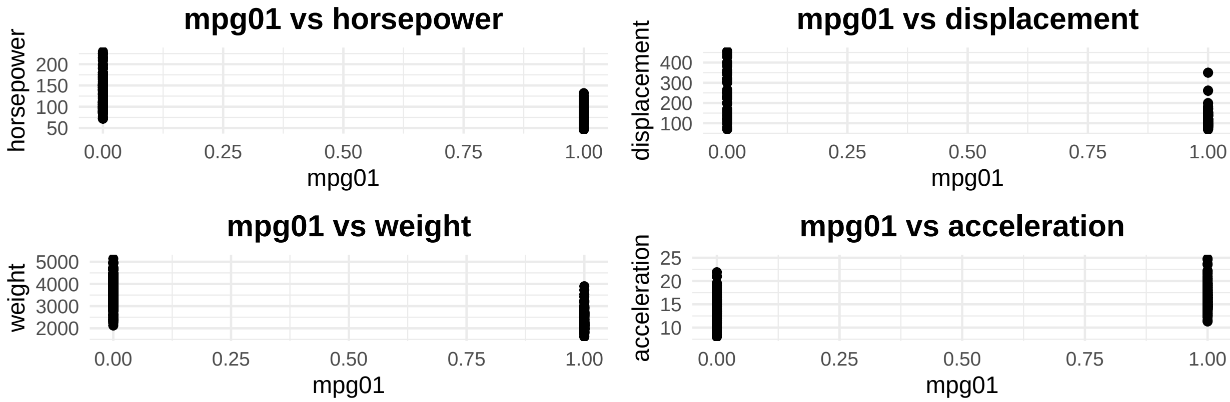
```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

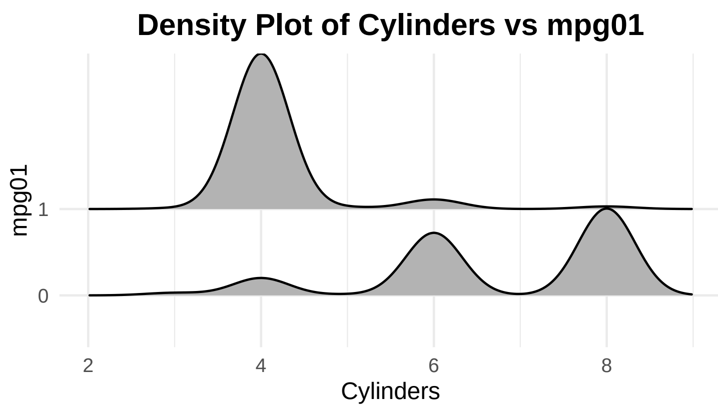
```
df <- Auto %>%
  dplyr::mutate(mpg01 = ifelse(mpg >= median(Auto$mpg, na.rm = TRUE), 1, 0))

# hist(df$mpg01)
```

```
pairs(df %>%
        dplyr::select(mpg01, mpg, cylinders, displacement,
                      horsepower, weight, acceleration, year, origin))
```



Well, clearly the best predictor of `mpg01` is going to be `mpg`, but it doesn't make sense to include `mpg` as a variable in a model attempting to predict `mpg01`. Beyond that, it looks like horsepower, weight, displacement, and acceleration may have some impact on `mpg01`.





We can see a semblance of linear relationships between `mpg01` and the top four variables. When we compare `mpg01` to `cylinders`, we see that there appears to be a heavy concentration of high `mpg` vehicles with only 4 cylinders, whereas low `mpg` vehicles (a `mpg01` = 0) appear to be concentrated at 6 or 8 cylinders.

Before we run the model, we first split the data into training and testing datasets using the `caret` library.

```r
set.seed(123)
ti <- caret::createDataPartition(df$mpg01, p = 0.75, list = FALSE)

train_data = df[ti, ]
test_data = df[-ti, ]
```

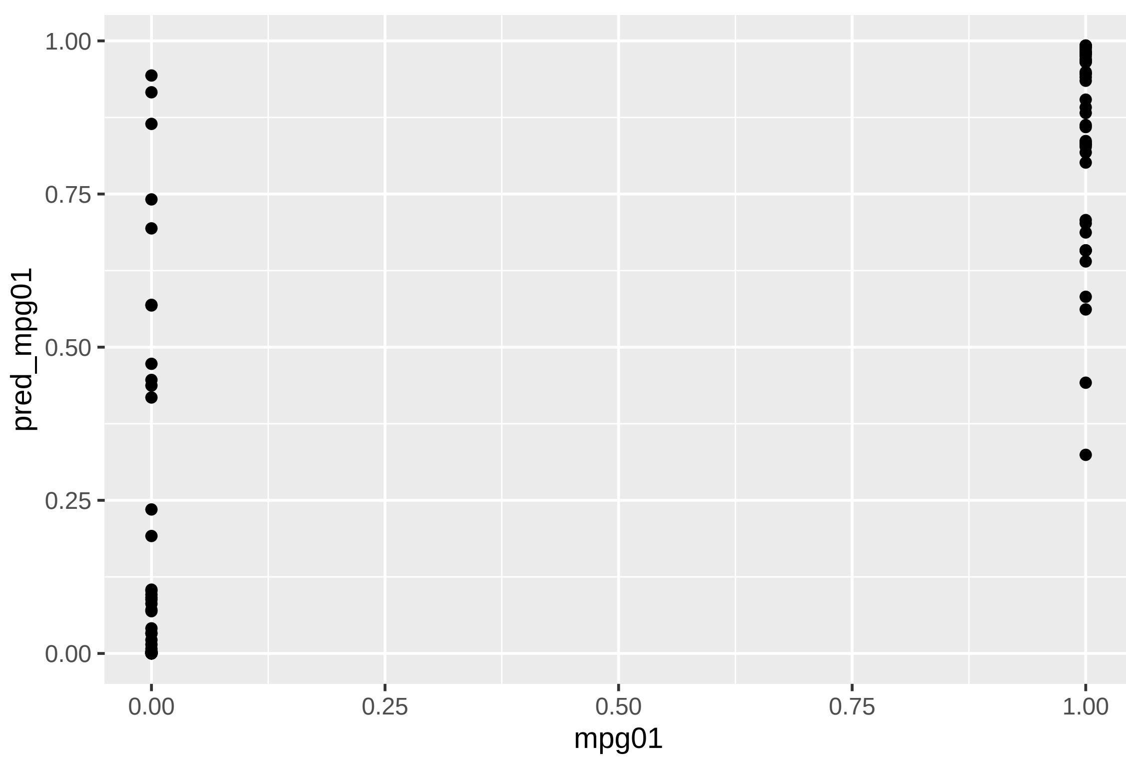
```r
log_model <- glm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration,
                 data = train_data, family = "binomial")
summary(log_model)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight + acceleration, family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1515  -0.2190   0.0473   0.3623   3.3259
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  11.097855   3.038713   3.652  0.00026 ***
## cylinders     0.117531   0.386559   0.304  0.76109
## displacement -0.015151   0.009201  -1.647  0.09963 .
## horsepower   -0.042977   0.022694  -1.894  0.05826 .
## weight       -0.001817   0.001018  -1.786  0.07415 .
## acceleration  0.011349   0.142132   0.080  0.93636
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 407.57  on 293  degrees of freedom
## Residual deviance: 160.11  on 288  degrees of freedom
## AIC: 172.11
##
## Number of Fisher Scoring iterations: 7
```

With our data split into training and testing data sets, we see that `displacement` and `horsepower` are significant variables (using a 5% level of significance). Variables like `weight` and `cylinders` provide some value, but are not statistically significant at a reasonable level, and `acceleration` provides no value.

```
test_data$pred_mpg01 <- predict(log_model, newdata = test_data, type = "response")

test_data %>%
  ggplot() +
  geom_point(aes(x = mpg01, y = pred_mpg01))
```



```r
test_data <- test_data %>%
  dplyr::mutate(pred_binary = ifelse(pred_mpg01 >= 0.5, 1, 0))

acc <- mean(test_data$mpg01 == test_data$pred_binary)
table(test_data$mpg01, test_data$pred_binary)
```

```
##
##      0  1
##   0 42  7
##   1  2 47
```

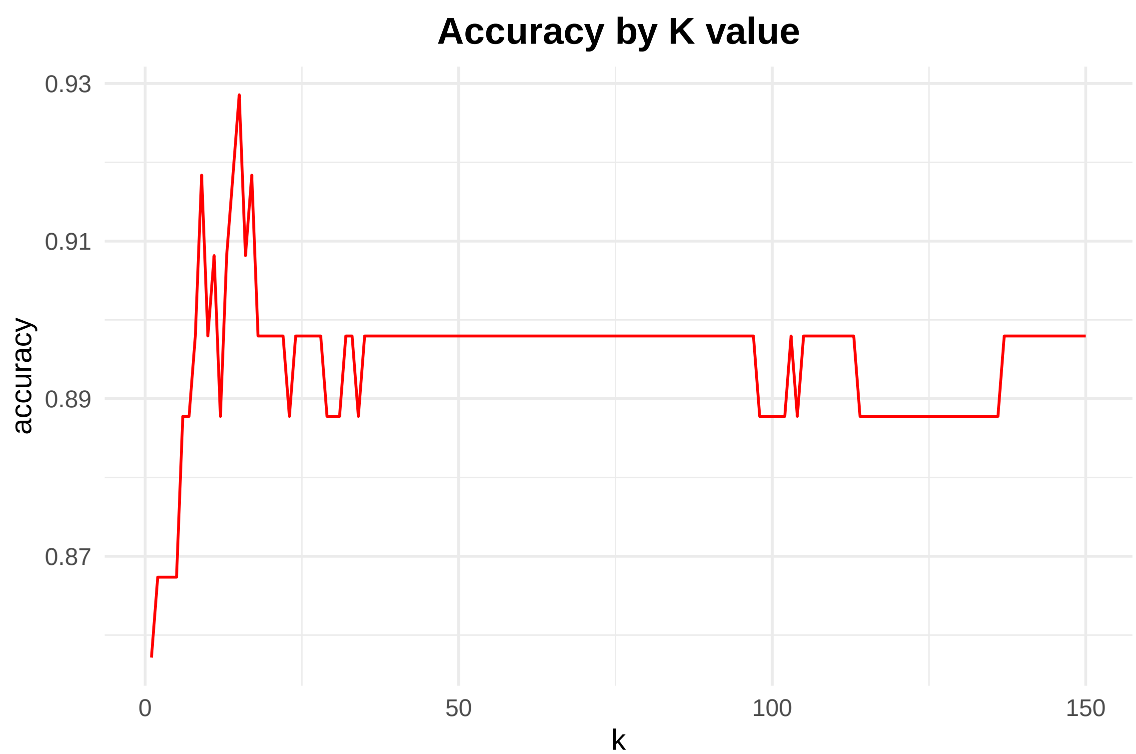
By running a confusion matrix on the predicted `mpg01` probabilities, we can see that 90.8 of the predictions were correct. The greatest error occurs when the `mpg01` $= 0$ in reality, but was predicted to be greater than 1.

```r
require(FNN)
```

```
## Loading required package: FNN
```

```r
set.seed(123)

mpg_train <- train_data %>%
  dplyr::select(cylinders, displacement, horsepower, weight, acceleration)
mpg_test <- test_data %>%
  dplyr::select(cylinders, displacement, horsepower, weight, acceleration)

train_label <- train_data %>%
  dplyr::select(mpg01)
test_label <- test_data %>%
  dplyr::select(mpg01)

knn_acc <- list()

for (z in 1:150) {
  knn_model <- knn(train = mpg_train, test = mpg_test, cl = train_data$mpg01, k = z)
  knn_table <- table(knn_model, test_data$mpg01)
  k_a <- sum(diag(knn_table))/ sum(knn_table)
  k_df <- data.frame(k_a, z)
  colnames(k_df) <- c("accuracy", "k")

  knn_acc[[z]] <- k_df

  # print(z)
}

knn_acc <- dplyr::bind_rows(knn_acc) %>%
  # dplyr::mutate(order = row_number()) %>%
  arrange(desc(accuracy))

knn_acc %>%
  ggplot() +
  geom_line(aes(x = k, y = accuracy), col = "red") +
```

```
  labs(title = "Accuracy by K value") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 14))
```




```
n <- floor(min(knn_acc$accuracy, na.rm = TRUE) * 100)
l <- ceiling(min(knn_acc$accuracy, na.rm = TRUE) * 100)

m <- knn_acc %>%
  dplyr::filter(accuracy == max(accuracy, na.rm = TRUE)) %>%
  dplyr::filter(k == max(k)) %>%
  pull(k)
```

We see that the accuracy values oscillate pretty heavily between 85% and 86%. The highest accuracy rate of 92.86 is achieved at `K` = 15.