Homework 3

STAT 430, Spring 2017

Due: Friday, February 17 by 11:59 PM

For this homework we return to the data found in `auto-train.csv` and `auto-test.csv` which contain train and test data respectively. `auto.csv` is provided but not used. It is a modification of the `Auto` data from the `ISLR` package.

We will use this data for each exercise in this homework. `mpg` will be the response for the entire assignment.

For information on the original data:

```
library(ISLR)
#?Auto
```

Exercise 1

[**10 points**] Use the training data to fit both a linear and logistic regression using only `displacement` as the predictor. Use both to create classifiers which seek to minimize the classification error.

For both:

- Plot the training data and add a line (or curve) with the predicted probabilities.
- Find decision boundary c. That is, find c such that

$$\hat{C}(\text{displacement}) = \begin{cases} 0 & \text{displacement} > c \\ 1 & \text{displacement} \leq c \end{cases}$$

- Report the test accuracy.

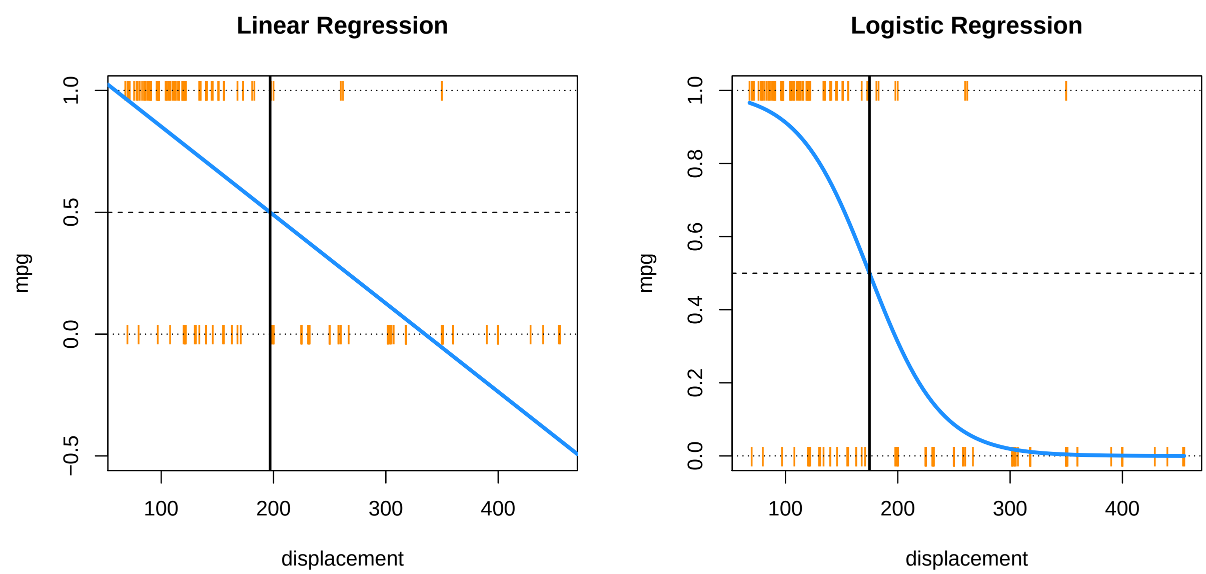
Solution:

```
auto_train = read_csv("auto-train.csv")
auto_test = read_csv("auto-test.csv")
```

```
auto_lm_disp = lm(mpg ~ displacement, data = auto_train)
auto_lr_disp = glm(mpg ~ displacement, data = auto_train, family = "binomial")
```

```
# two plots, one row, two columns
par(mfrow = c(1, 2))

# linear regression plot
plot(mpg ~ displacement, data = auto_train,
     col = "darkorange", pch = "|", ylim = c(-0.5, 1),
     main = "Linear Regression")
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)
abline(auto_lm_disp, lwd = 3, col = "dodgerblue")
auto_lm_disp_dec_bound = (0.5 - coef(auto_lm_disp)[1]) / coef(auto_lm_disp)[2]
abline(v = auto_lm_disp_dec_bound, lwd = 2)

# logistic regression plot
```

```r
plot(mpg ~ displacement, data = auto_train,
     col = "darkorange", pch = "|",
     main = "Logistic Regression")
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)
curve(predict(auto_lr_disp, data.frame(displacement = x), type = "response"),
      add = TRUE, lwd = 3, col = "dodgerblue")
auto_lr_disp_dec_bound = -coef(auto_lr_disp)[1] / coef(auto_lr_disp)[2]
abline(v = auto_lr_disp_dec_bound, lwd = 2)
```



- Linear Regression Cutoff: 196.9697183
- Logistic Regression Cutoff: 174.7747264

```r
lm_pred = ifelse(predict(auto_lm_disp, newdata = auto_test) > 0.5, 1, 0)
mean(lm_pred == auto_test$mpg) # linear regression test accuracy
```

```
## [1] 0.9130435
```

```r
lr_pred = ifelse(predict(auto_lr_disp, newdata = auto_test, type = "response") > 0.5, 1, 0)
mean(lr_pred == auto_test$mpg) # logistic regression test accuracy
```

```
## [1] 0.9130435
```

Intersetingly, the linear and logistic regression obtain the same test error. (This will **not** always be the case. This is a small test set, and they had a similar cutoff.)

Exercise 2

[**12 points**] Now consider a logistic regression that considers two predictors, `acceleration` and `weight` in an additive model. Do the following:

- Plot the training data with `acceleration` as the x axis, and `weight` as the y axis, with the points colored according to their class. Add a line which represents the decision boundary for a classifier using

0.5 as a cutoff for predicted probability. **This may be challenging.**

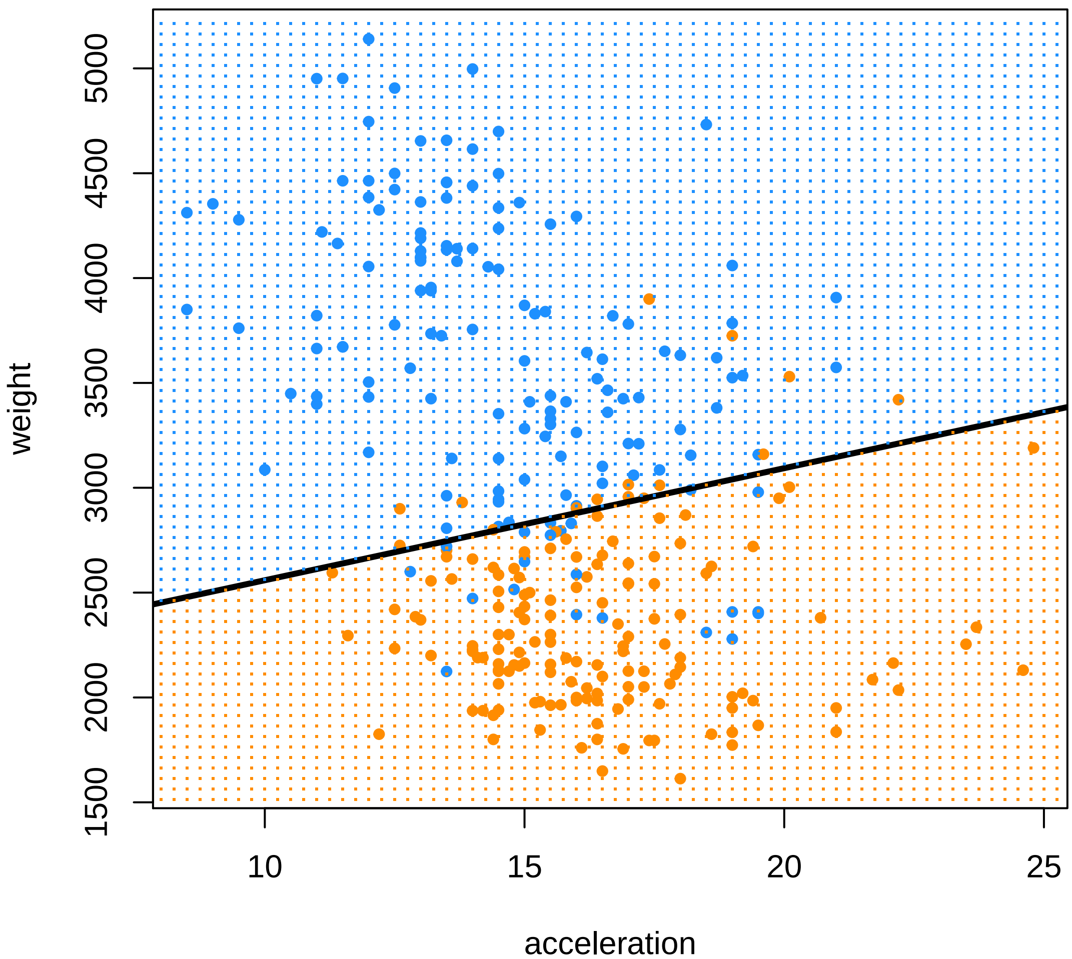
- Report test sensitivity, test specificity, and test accuracy for three classifiers, each using a different cutoff for predicted probability:
  - 0.2
  - 0.5
  - 0.8
- Plot an ROC curve and report the AUC.

Solution:

```
auto_lr_acc_weight = glm(mpg ~ acceleration + weight, data = auto_train, family = "binomial")
```

```
glm_boundary_line = function(glm_fit) {
  intercept = as.numeric(-coef(glm_fit)[1] / coef(glm_fit)[3])
  slope = as.numeric(-coef(glm_fit)[2] / coef(glm_fit)[3])
  c(intercept = intercept, slope = slope)
}

add_glm_boundary = function(glm_fit, line_col = "black") {
  abline(glm_boundary_line(glm_fit), col = line_col, lwd = 3)
}
```

```
a = glm_boundary_line(auto_lr_acc_weight)[1]
b = glm_boundary_line(auto_lr_acc_weight)[2]
wt = seq(min(auto_train$weight) - 100, max(auto_train$weight) + 100, by = 50)
ac = seq(min(auto_train$acceleration) - 0.5, max(auto_train$acceleration) + 0.5, by = 0.25)
grid = expand.grid(ac = ac, wt = wt)
bgcol = ifelse(grid$wt > a + b * grid$ac, "dodgerblue", "darkorange")
mpg_col = ifelse(auto_train$mpg == 1, "darkorange", "dodgerblue")

plot(weight ~ acceleration, data = auto_train, col = mpg_col, pch = 20)
add_glm_boundary(auto_lr_acc_weight)
points(expand.grid(ac, wt), col = bgcol, pch = ".")
```



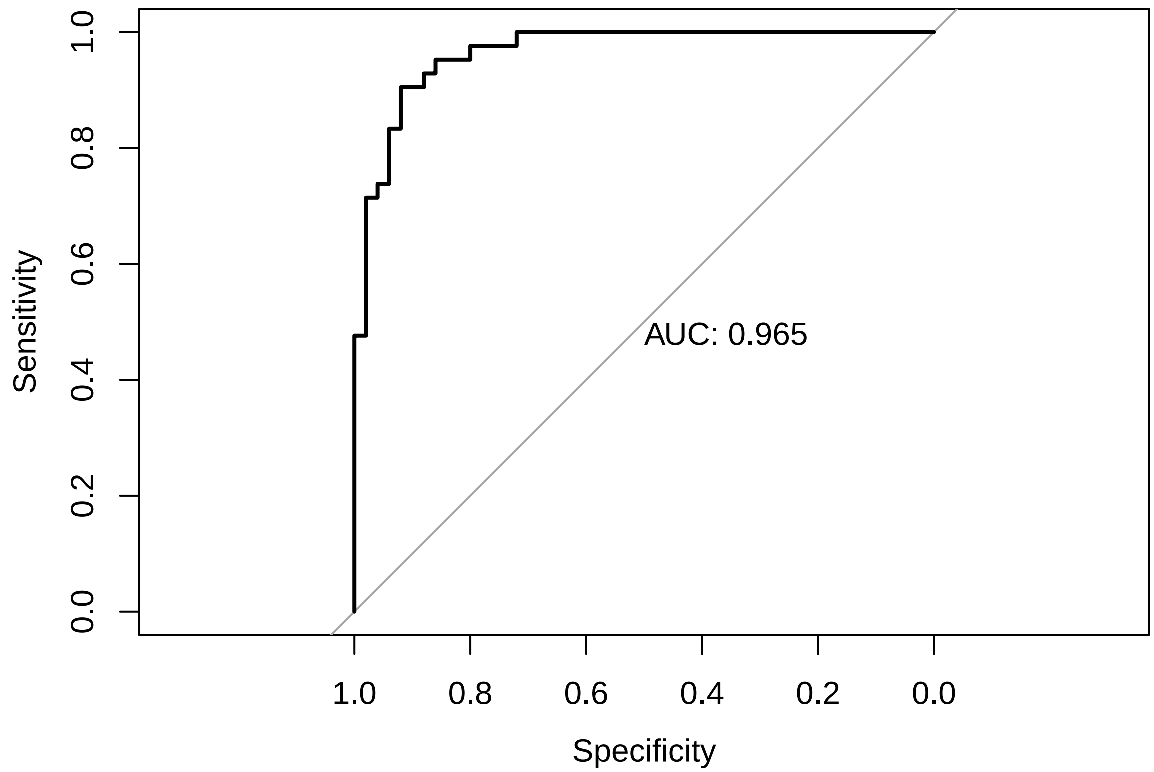

```
get_pred = function(mod, data, res = "y", pos = 1, neg = 0, cut = 0.5) {
  probs = predict(mod, newdata = data, type = "response")
  ifelse(probs > cut, pos, neg)
}
```

```
pred_20 = get_pred(auto_lr_acc_weight, auto_test, res = "mpg", cut = 0.2)
pred_50 = get_pred(auto_lr_acc_weight, auto_test, res = "mpg", cut = 0.5)
pred_80 = get_pred(auto_lr_acc_weight, auto_test, res = "mpg", cut = 0.8)

tab_20 = table(predicted = pred_20, actual = auto_test$mpg)
tab_50 = table(predicted = pred_50, actual = auto_test$mpg)
tab_80 = table(predicted = pred_80, actual = auto_test$mpg)

con_mat_20 = caret::confusionMatrix(tab_20, positive = "1")
con_mat_50 = caret::confusionMatrix(tab_50, positive = "1")
con_mat_80 = caret::confusionMatrix(tab_80, positive = "1")
```

```r
metrics = rbind(

  c(con_mat_20$overall["Accuracy"],
    con_mat_20$byClass["Sensitivity"],
    con_mat_20$byClass["Specificity"]),

  c(con_mat_50$overall["Accuracy"],
    con_mat_50$byClass["Sensitivity"],
    con_mat_50$byClass["Specificity"]),

  c(con_mat_80$overall["Accuracy"],
    con_mat_80$byClass["Sensitivity"],
    con_mat_80$byClass["Specificity"])

)

rownames(metrics) = c("c = 0.20", "c = 0.50", "c = 0.80")
knitr::kable(metrics)
```

	Accuracy	Sensitivity	Specificity
c = 0.20	0.8369565	1.0000000	0.70
c = 0.50	0.9021739	0.9285714	0.88
c = 0.80	0.8804348	0.8095238	0.94

```r
library(pROC)
test_prob = predict(auto_lr_acc_weight, newdata = auto_test, type = "response")
test_roc = roc(auto_test$mpg ~ test_prob, plot = TRUE, print.auc = TRUE)
```




```r
as.numeric(test_roc$auc)
```

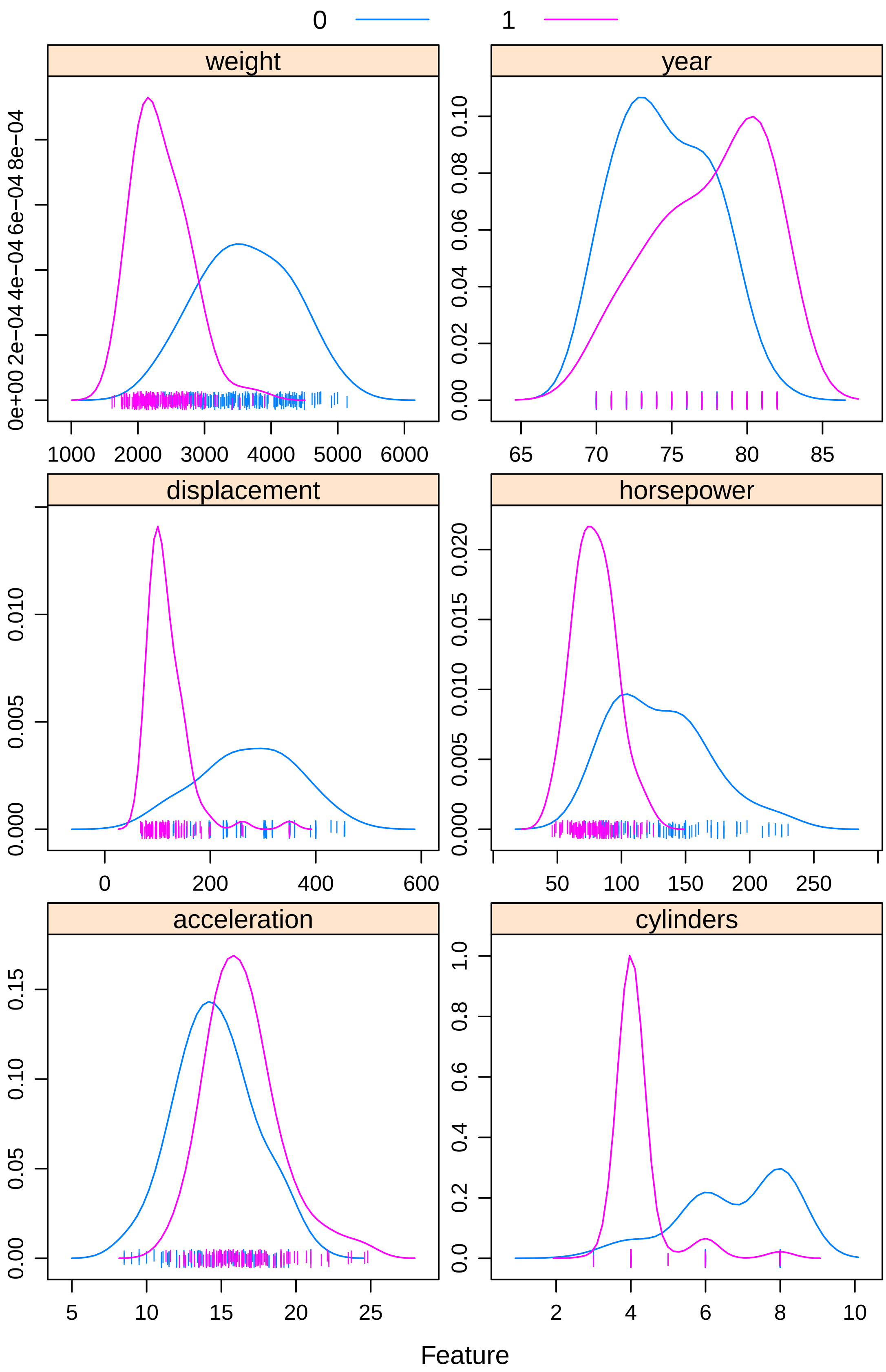
```
## [1] 0.9652381
```

Exercise 3

[**8 points**] Finally, consider the full additive logistic regression. Create an improved model for classification by adding (or removing) complexity. Report relevant metrics for both models to justify your model.

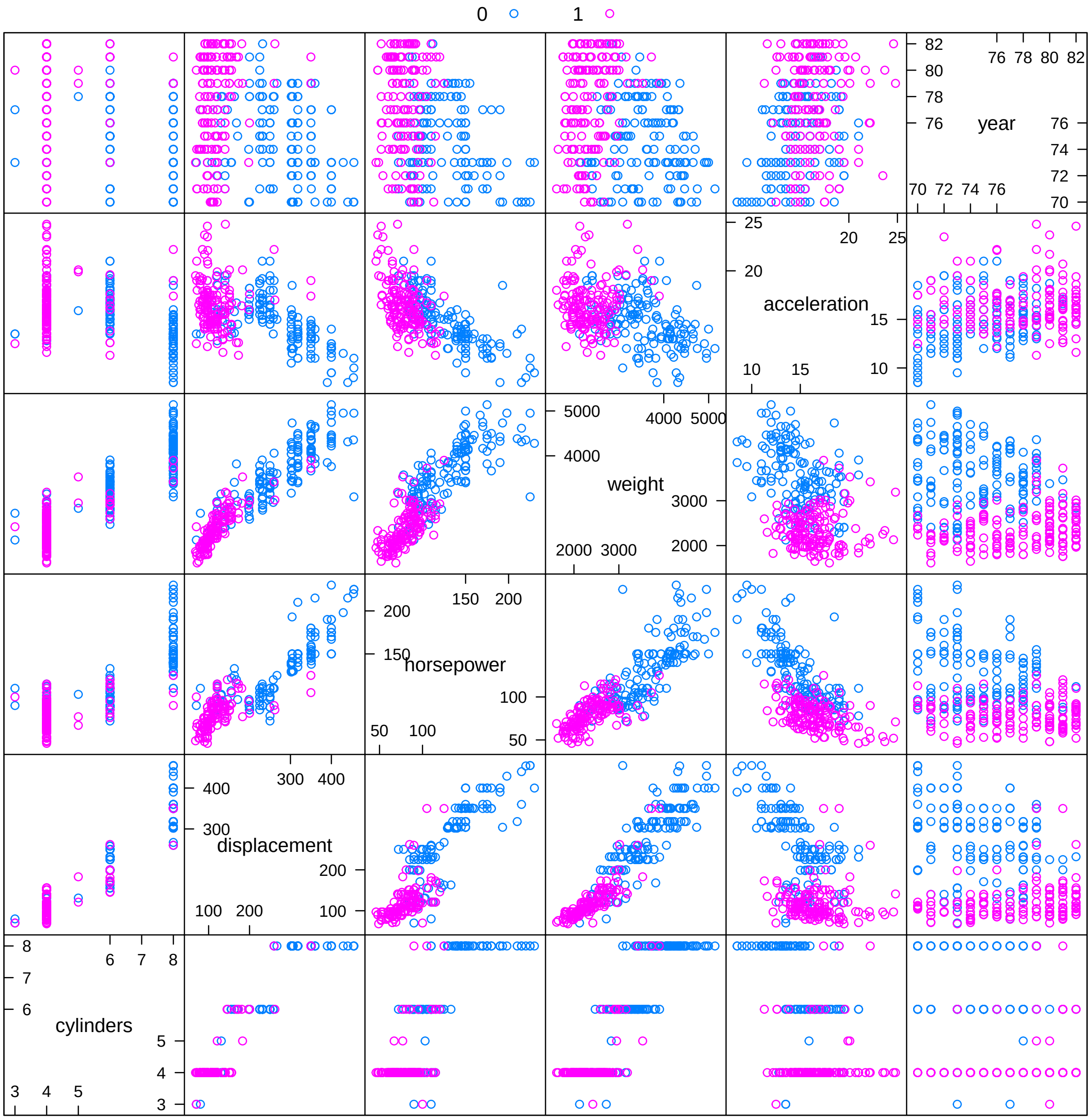
Solution:

```r
caret::featurePlot(x = auto_train[, -1],
            y = as.factor(auto_train$mpg),
            plot = "density",
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            adjust = 1.5,
            pch = "|",
            layout = c(2, 3),
            auto.key = list(columns = 2))
```



0
1
weight
year
displacement
horsepower
acceleration
cylinders
0e+00 2e−04 4e−04 6e−04 8e−04
1000 2000 3000 4000 5000 6000
0.00 0.02 0.04 0.06 0.08 0.10
65 70 75 80 85
0.000 0.005 0.010
0 200 400 600
0.000 0.005 0.010 0.015 0.020
50 100 150 200 250
0.00 0.05 0.10 0.15
5 10 15 20 25
0.0 0.2 0.4 0.6 0.8 1.0
2 4 6 8 10
Feature

```r
caret::featurePlot(x = auto_train[, -1],
                   y = as.factor(auto_train$mpg),
                   plot = "pairs",
                   auto.key = list(columns = 2))
```



Scatter Plot Matrix

```r
get_accuracy = function(mod, data, res = "y", pos = 1, neg = 0, cut = 0.5) {
  probs = predict(mod, newdata = data, type = "response")
  preds = ifelse(probs > cut, pos, neg)
  mean(data[, res] == preds)
}
```

```r
auto_lr_add = glm(mpg ~ ., data = auto_train, family = "binomial")
auto_lr_better = glm(mpg ~ . + I(weight ^ 2), data = auto_train, family = "binomial")
```

```r
# Additive Test Accuracy
add_acc = get_accuracy(auto_lr_add, data = auto_test,
                       res = "mpg", pos = 1, neg = 0, cut = 0.5)

# Improved Test Accuracy
imp_acc = get_accuracy(auto_lr_better, data = auto_test,
                       res = "mpg", pos = 1, neg = 0, cut = 0.5)
```

```r
c(Additive = add_acc, Improved = imp_acc)
```

```
##  Additive  Improved
## 0.9239130 0.9347826
```

Here we see that the "improved" model is improved. According the the classification accuracy, it is better at classification.