Exercise 1

Based on these results, which method should we prefer to use for classification of new observations? Why?

Generally, we prefer the model that performs better on the test set. In this case the exact performance of 1-nearest neighbors is not given, but only the average performance over both train and test set. However, we can deduct the test set performance of 1-nearest neighbors, because 1-nearest neighbors (with k=1) always has a train set error of 0. So, the test set performance of 1-nearest neighbors has to be 18 * 2 = 36, which is higher than the test set performance of the logistic regression. Therefore, we should use the logistic regression model.

Exercise 2

```
library(tidymodels)
library(GGally)
library(ISLR)
library(discrim)
library(poissonreg)

auto <- tibble(Auto)

?ISLR::Auto
```

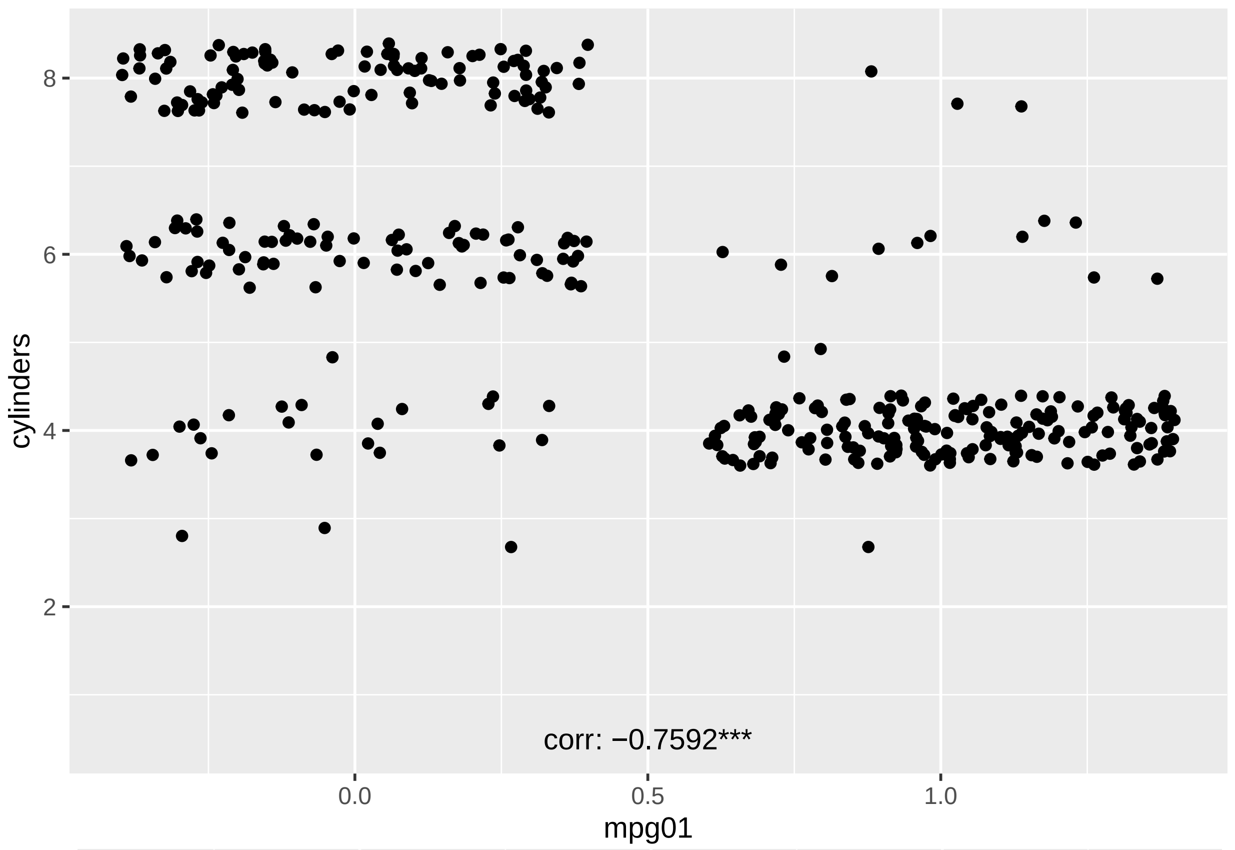
Create a binary variable, `mpg01` that specifies whether `mpg` is above its median.

```
auto_median <- Auto %>%
  select(mpg) %>%
  pull() %>%
  median()

auto <- auto %>%
  mutate(mpg01 = ifelse(mpg > auto_median, 1, 0))
```

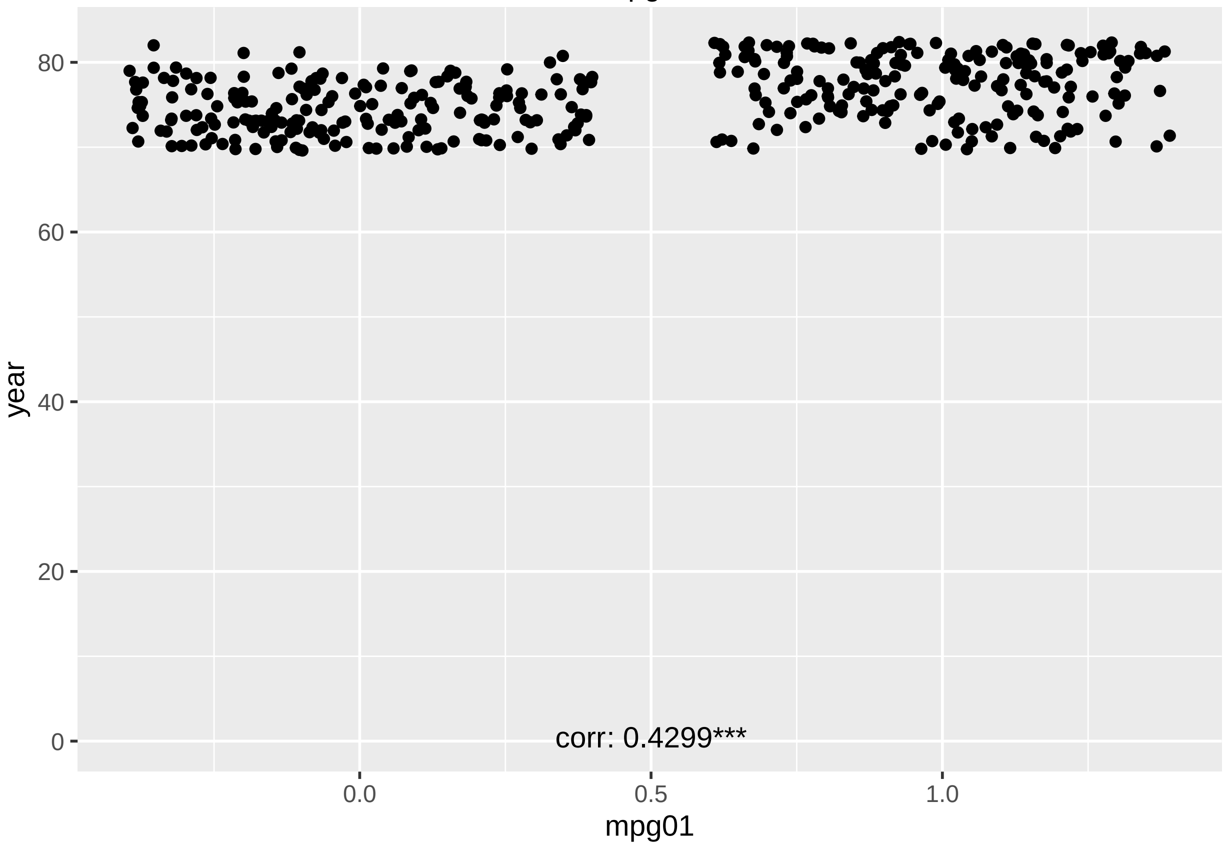
To explore the data and the relationship between `mpg01` and the other variables, we will plot all continuous variables with a boxplot for `mpg01 = 0` and `mpg01 = 1`.

For variables that are "more or less" categorical, e.g. they have a small number of unique values we will use a jitter plot instead.

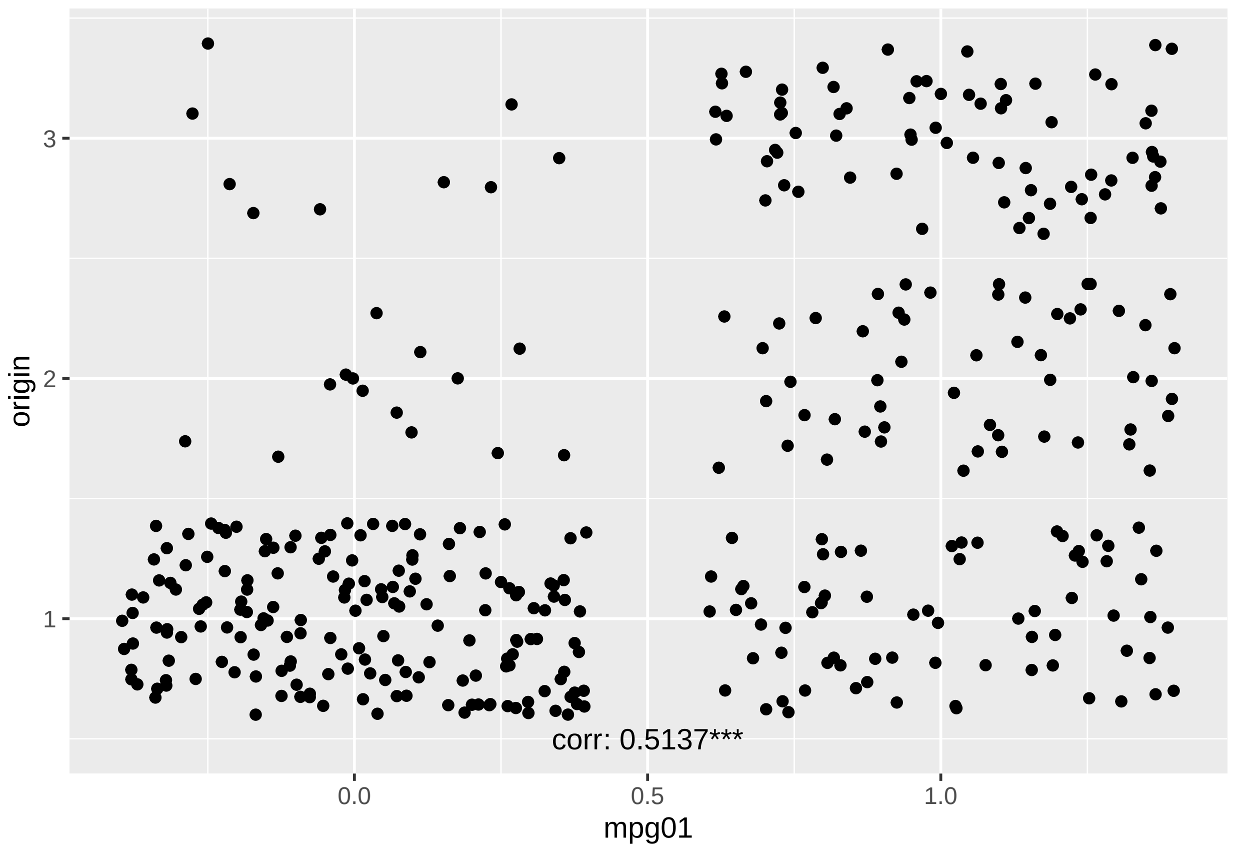
```
categorical_cols <- c("cylinders", "year", "origin")

for (col in categorical_cols) {
  test <- cor.test(pull(auto[, col]), auto$mpg01)
  correlation <- test$estimate %>% round(4)
  p <- test$p.value
  stars <- ifelse(p < 0.01, "***",
    ifelse(p < 0.05, "**",
      ifelse(p < 0.1, "*", "")
    )
  )
  print(
    ggplot(auto, aes_string(x = "mpg01", y = col)) +
      geom_jitter() +
      annotate("text", label = paste0("corr: ", correlation, stars), x = 0.5, y = 0.5)
  )
}
```



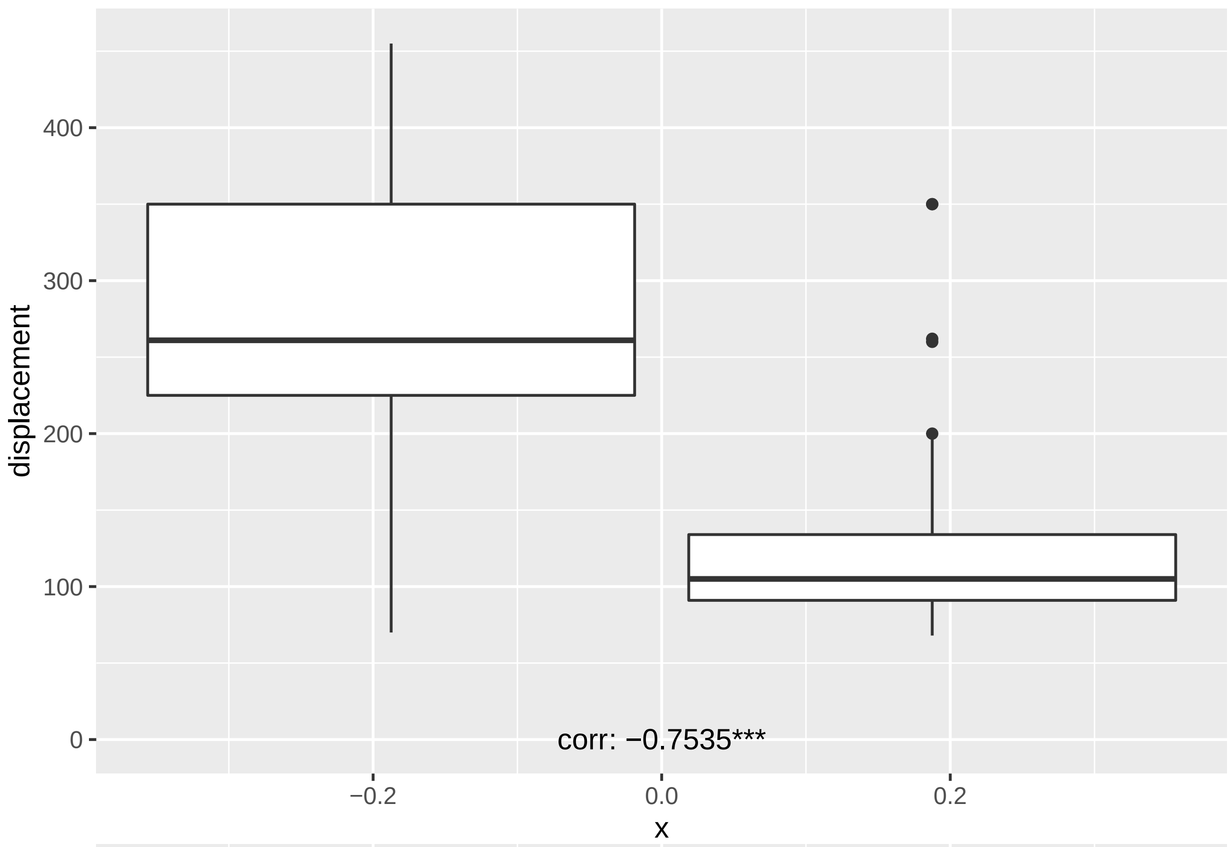
corr: −0.7592***
cylinders
mpg01
8
6
4
2
0.0
0.5
1.0



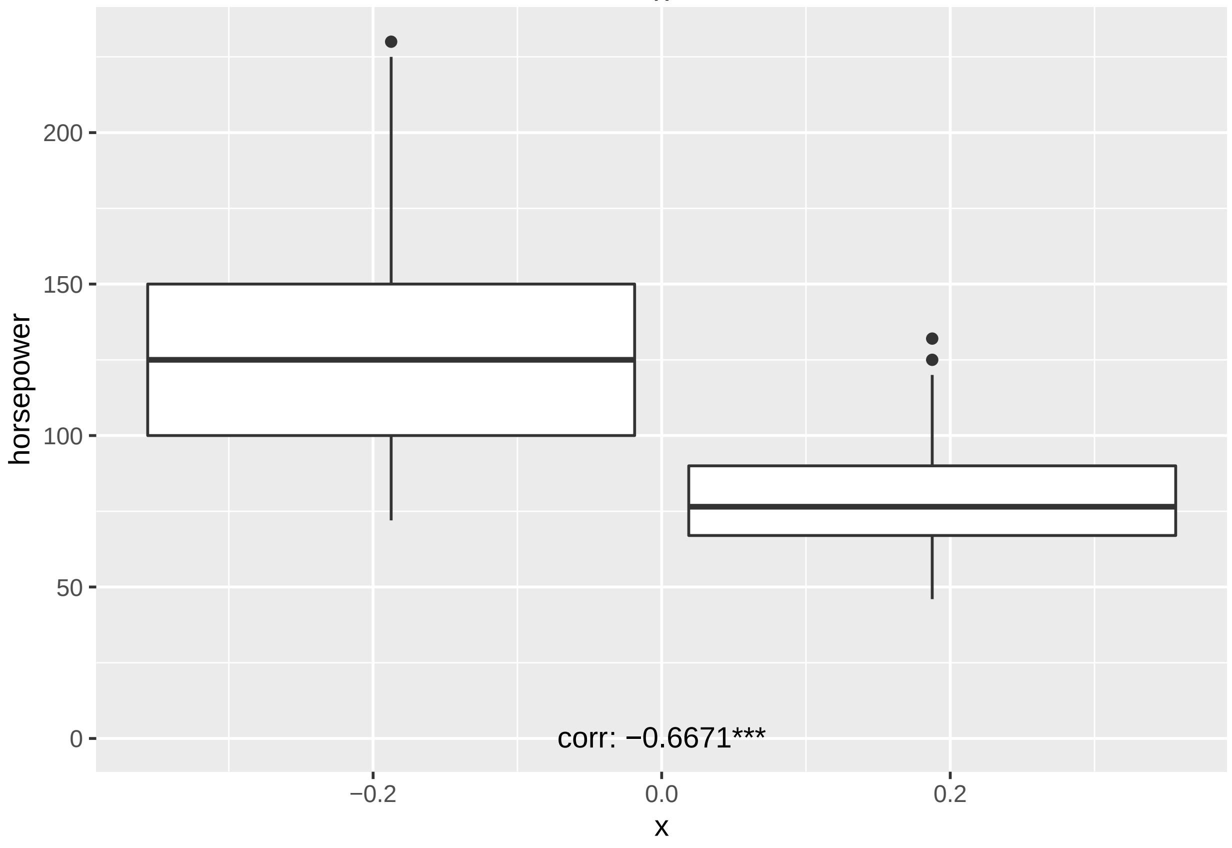
corr: 0.4299***
year
mpg01
80
60
40
20
0
0.0
0.5
1.0



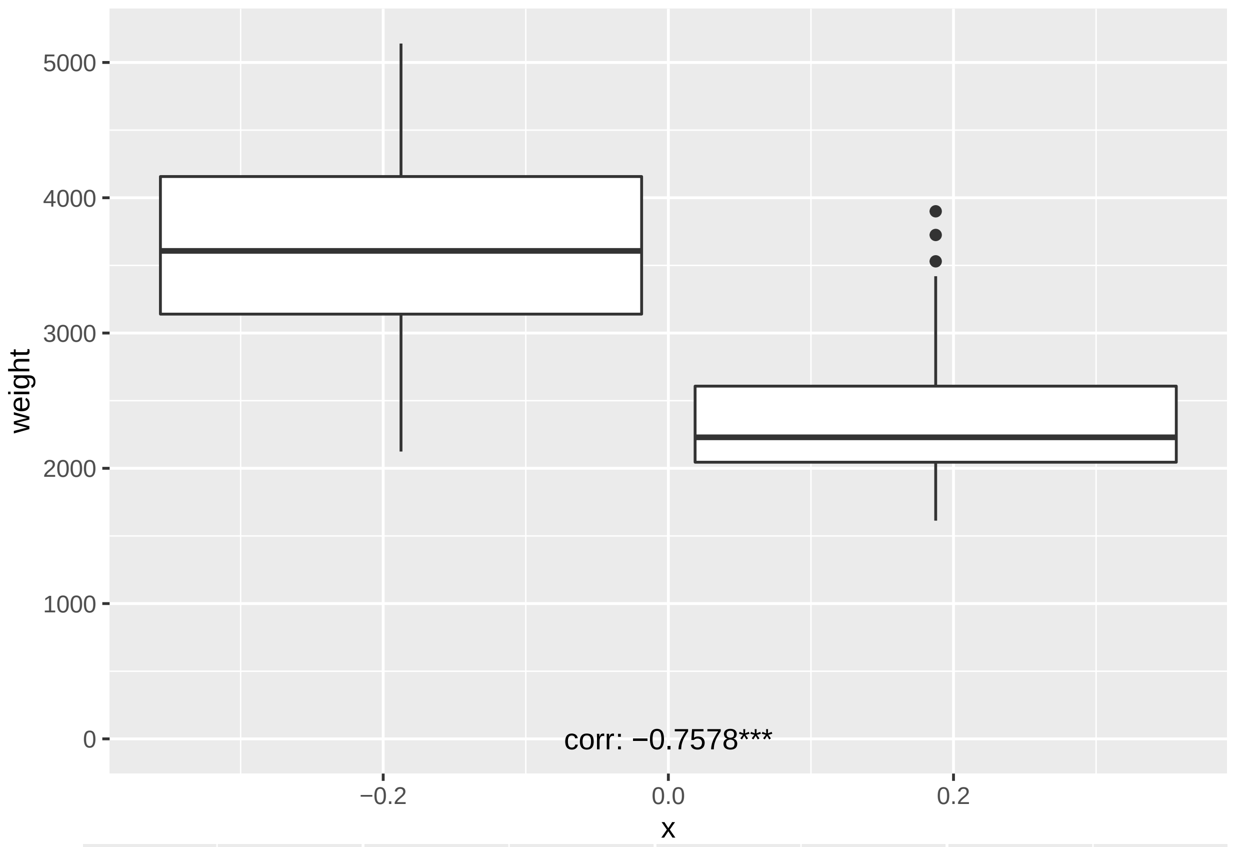
```r
continuous_cols <- c("displacement", "horsepower", "weight", "acceleration")

for (col in continuous_cols) {
  test <- cor.test(pull(auto[, col]), auto$mpg01)
  correlation <- test$estimate %>% round(4)
  p <- test$p.value
  stars <- ifelse(p < 0.01, "***",
    ifelse(p < 0.05, "**",
      ifelse(p < 0.1, "*", "")
    )
  )
  print(
    ggplot(auto, aes_string(group = "mpg01", y = col)) +
      geom_boxplot() +
      annotate("text", label = paste0("corr: ", correlation, stars), x = 0, y = 0.5)
  )
}
```



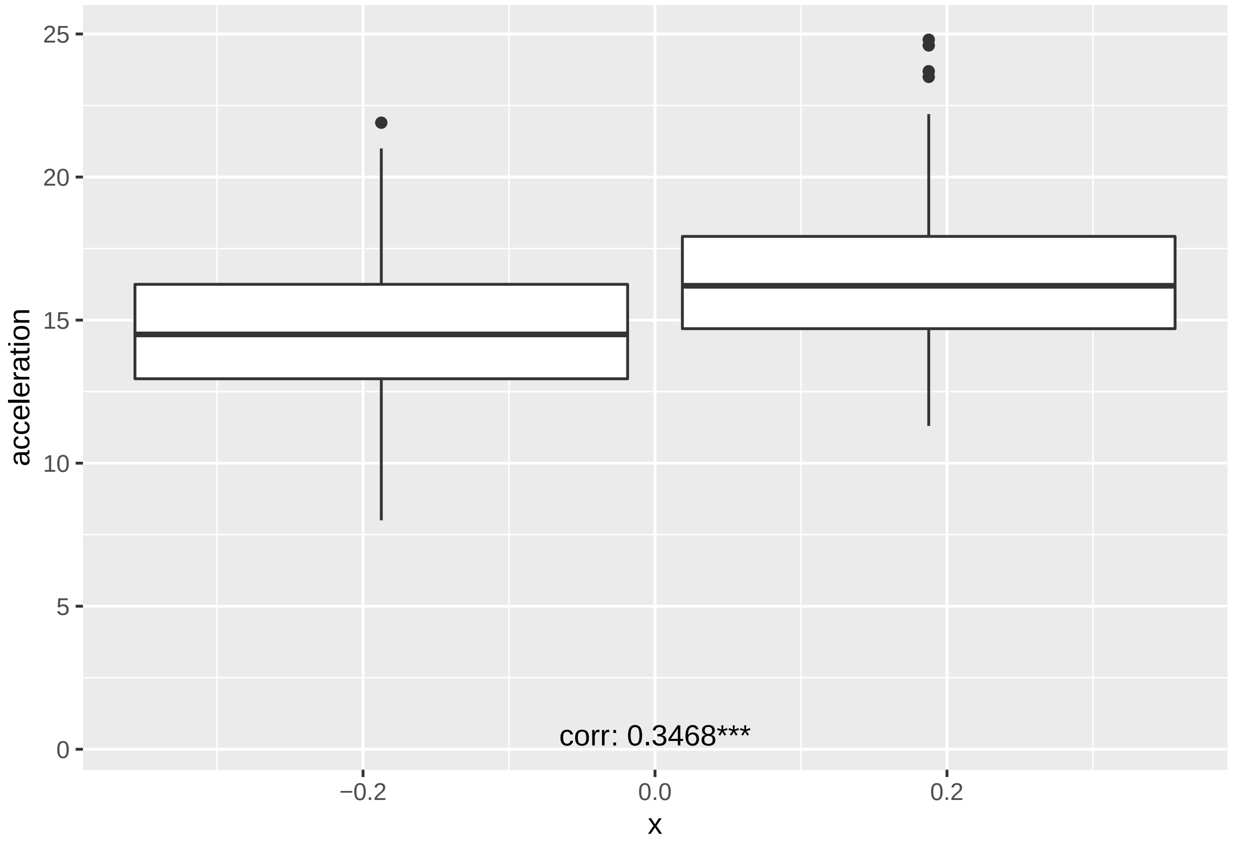
displacement
400
300
200
100
0
corr: −0.7535***
−0.2
0.0
0.2
x



horsepower
200
150
100
50
0
corr: −0.6671***
−0.2
0.0
0.2
x



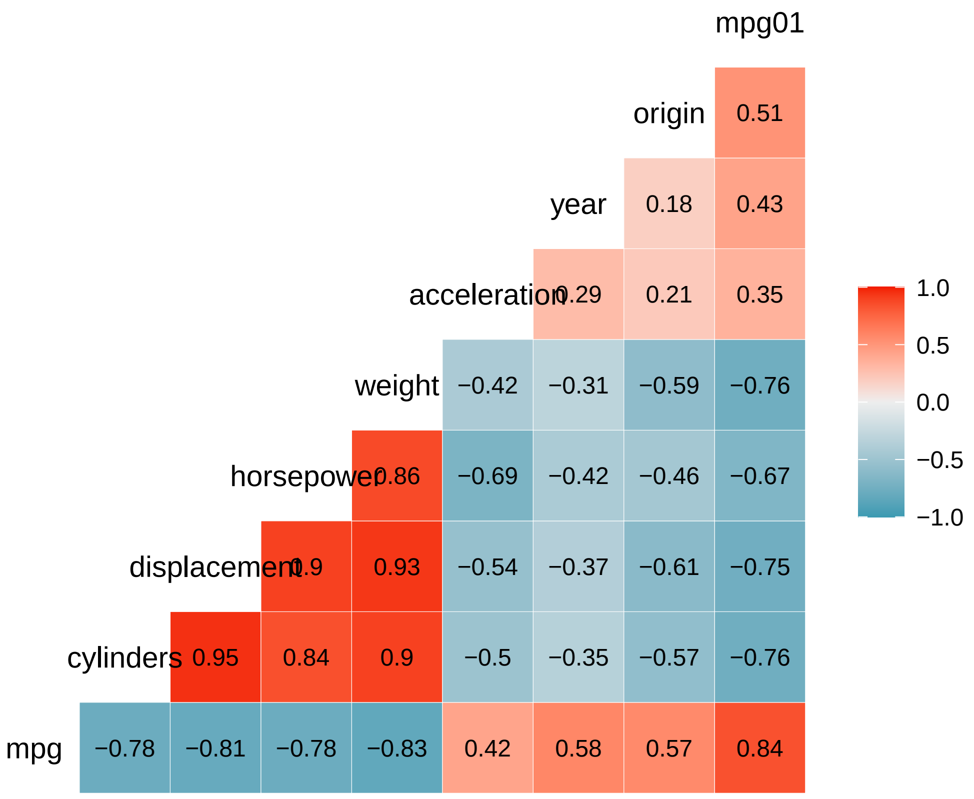
5000
4000
3000
2000
1000
0
weight
corr: −0.7578***
−0.2
0.0
0.2
x



25
20
15
10
5
0
acceleration
corr: 0.3468***
−0.2
0.0
0.2
x

```
auto %>%
  ggcorr(
    method = c("pairwise", "pearson"),
    nbreaks = NULL, digits = 2, geom = "tile",
    label = T, label_alpha = F, label_size = 3, label_round = 2
  )
```

```
## Warning in ggcorr(., method = c("pairwise", "pearson"), nbreaks = NULL, : data
## in column(s) 'name' are not numeric and were ignored
```



As we can see all of the variables are correlated with `mpg01` with high statistical significance. In terms of magnitude, the correlation with `displacement`, `weight` and `cylinders` look the most promising.

Now we will split the data into a training and a test set.

```
auto$mpg01 <- as.factor(auto$mpg01)

set.seed(1)
auto_split <- initial_split(auto, strata = mpg01, prop = 0.5)
print(auto_split)
```

```
## <Analysis/Assess/Total>
## <196/196/392>
```

```
auto_train <- training(auto_split)
auto_test <- testing(auto_split)
```

Now we will perform LDA and use `displacement`, `weight` and `cylinders` as features.

```
lda_spec <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")
```

```
lda_fit <- lda_spec %>%
  fit(mpg01 ~ displacement + weight + cylinders, data = auto_train)

lda_fit
```

```
## parsnip model object
##
## Call:
## lda(mpg01 ~ displacement + weight + cylinders, data = data)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##   displacement   weight cylinders
## 0     274.6327 3602.449  6.816327
## 1     119.8316 2360.439  4.234694
##
## Coefficients of linear discriminants:
##                       LD1
## displacement -0.0014105302
## weight       -0.0006528192
## cylinders    -0.5512837489
```

Now we evaluate the model on the test set.

```
augment(lda_fit, new_data = auto_test) %>%
  accuracy(truth = mpg01, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.898
```

Now we will repeat the same with QDA.

```
qda_spec <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_fit <- qda_spec %>%
  fit(mpg01 ~ displacement + weight + cylinders, data = auto_train)

qda_fit
```

```
## parsnip model object
##
## Call:
## qda(mpg01 ~ displacement + weight + cylinders, data = data)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
```

```
##   displacement   weight cylinders
## 0     274.6327 3602.449  6.816327
## 1     119.8316 2360.439  4.234694
```

And evaluate the model on the test set.

```
augment(qda_fit, new_data = auto_test) %>%
  accuracy(truth = mpg01, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.898
```

Lastly, we will also perform the same analysis with the logistic regression.

```
logreg_spec <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm")

logreg_fit <- logreg_spec %>%
  fit(mpg01 ~ displacement + weight + cylinders, data = auto_train)

logreg_fit
```

```
## parsnip model object
##
##
## Call:  stats::glm(formula = mpg01 ~ displacement + weight + cylinders,
##     family = stats::binomial, data = data)
##
## Coefficients:
##  (Intercept)  displacement        weight     cylinders
##     9.222571     -0.012728     -0.001714     -0.387403
##
## Degrees of Freedom: 195 Total (i.e. Null);  192 Residual
## Null Deviance:       271.7
## Residual Deviance: 113.2     AIC: 121.2
```

And evaluate the model on the test set.

```
augment(logreg_fit, new_data = auto_test) %>%
  accuracy(truth = mpg01, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.893
```

Exercise 3

Explain how k-fold cross-validation is implemented

In k-fold cross-validation, we split the training data into k equal-sized subsets. Often the split is stratified, so that the proportion/distribution of each class or outcome in each subset is approximately the same. Then, we train our model(s) on k-1 of the subsets and test it on the remaining subset. We repeat this so that each subset is used as a test set once. The overall performance of the model is the average of the performance in each of the k folds.

What are the advantages and disadvantages of k-fold cross-validation relative to the validation set approach

The advantages of k-fold cross-validation is, that the average performance of all folds is less prone to noisy data and therefore has more value for us.

The disadvantage is that we have to train k models, instead of just one, therefore k-fold cross validation always takes k times longer than simple hold out validation. Another obvious disadvantage is that k-fold cross validation is more complex than simple hold out.