Week 10 - Homework

STAT 420, Summer 2018, Unger

Exercise 1 (Simulating Wald and Likelihood Ratio Tests)

In this exercise we will investigate the distributions of hypothesis tests for logistic regression. For this exercise, we will use the following predictors.

```r
sample_size = 150
set.seed(420)
x1 = rnorm(n = sample_size)
x2 = rnorm(n = sample_size)
x3 = rnorm(n = sample_size)
```

Recall that

$$p(\mathbf{x}) = P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

Consider the true model

$$\log\left(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1$$

where

- $\beta_0 = 0.4$
- $\beta_1 = -0.35$

(a) To investigate the distributions, simulate from this model 2500 times. To do so, calculate

$$P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

for an observation, and then make a random draw from a Bernoulli distribution with that success probability. (Note that a Bernoulli distribution is a Binomial distribution with parameter $n = 1$. There is no direction function in `R` for a Bernoulli distribution.)

Each time, fit the model:

$$\log\left(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

Store the test statistics for two tests:

- The Wald test for $H_0 : \beta_2 = 0$, which we say follows a standard normal distribution for "large" samples
- The likelihood ratio test for $H_0 : \beta_2 = \beta_3 = 0$, which we say follows a χ^2 distribution (with some degrees of freedom) for "large" samples

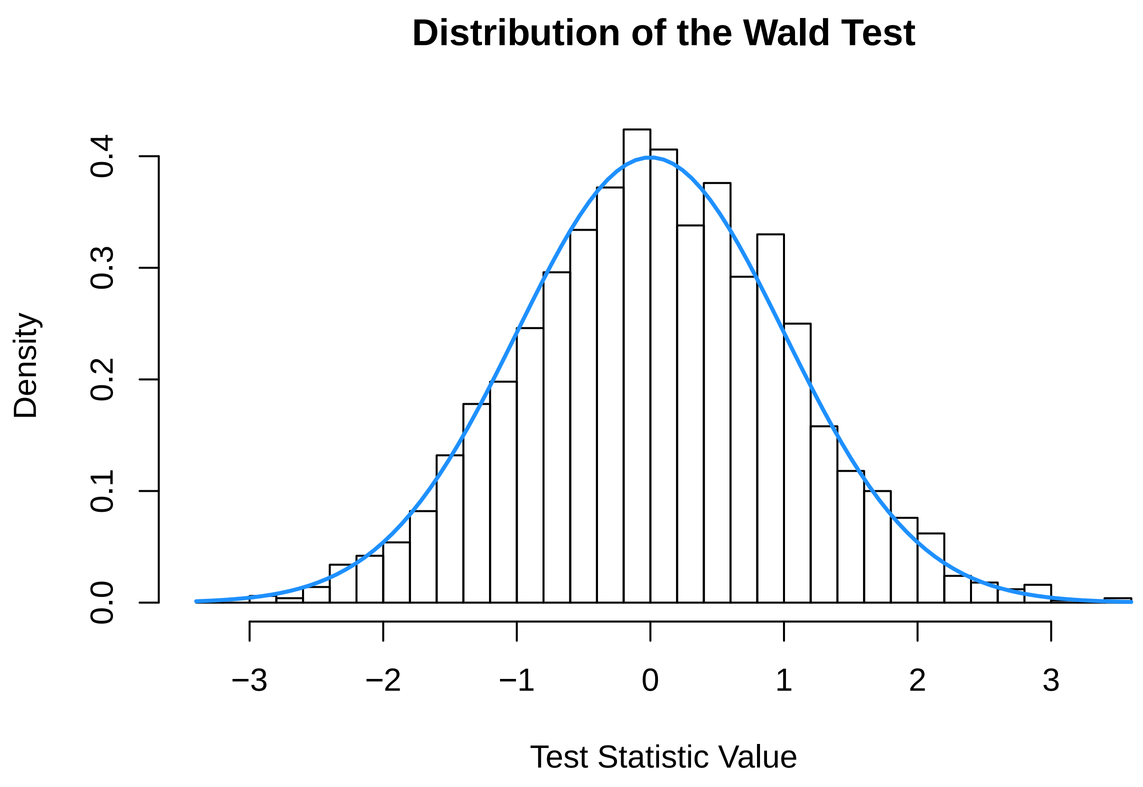
Solution:

```r
# setup
eta = 0.4 - 0.35 * x1
p = 1 / (1 + exp(-eta))
y = 0
sim_data = data.frame(y, x1, x2, x3)
num_sims = 2500
LRT_test_stat = rep(0, num_sims)
wald_test_stat = rep(0, num_sims)

# run simulations
for (i in seq_along(LRT_test_stat)) {

  # simulate response data
  sim_data$y = rbinom(n = sample_size, 1, prob = p)

  # fit requested models
  fit_0 = glm(y ~ x1, data = sim_data, family = "binomial")
  fit_1 = glm(y ~ ., data = sim_data, family = "binomial")

  # store statistics
  LRT_test_stat[i] = anova(fit_0, fit_1, test = "Chisq")[2, "Deviance"]
  wald_test_stat[i] = coef(summary(fit_1))["x2", "z value"]

}
```

(b) Plot a histogram of the empirical values for the Wald test statistic. Overlay the density of the true distribution assuming a large sample.

Solution:

```r
hist(wald_test_stat, breaks = 25, prob = TRUE,
     main = "Distribution of the Wald Test", xlab = "Test Statistic Value")
curve(dnorm(x), add = TRUE, col = "dodgerblue", lwd = 2)
```



(c) Use the empirical results for the Wald test statistic to estimate the probability of observing a test statistic larger than 1. Also report this probability using the true distribution of the test statistic assuming a large sample.

Solution:

```r
mean(wald_test_stat > 1) # using empirical
```

```
## [1] 0.168
```

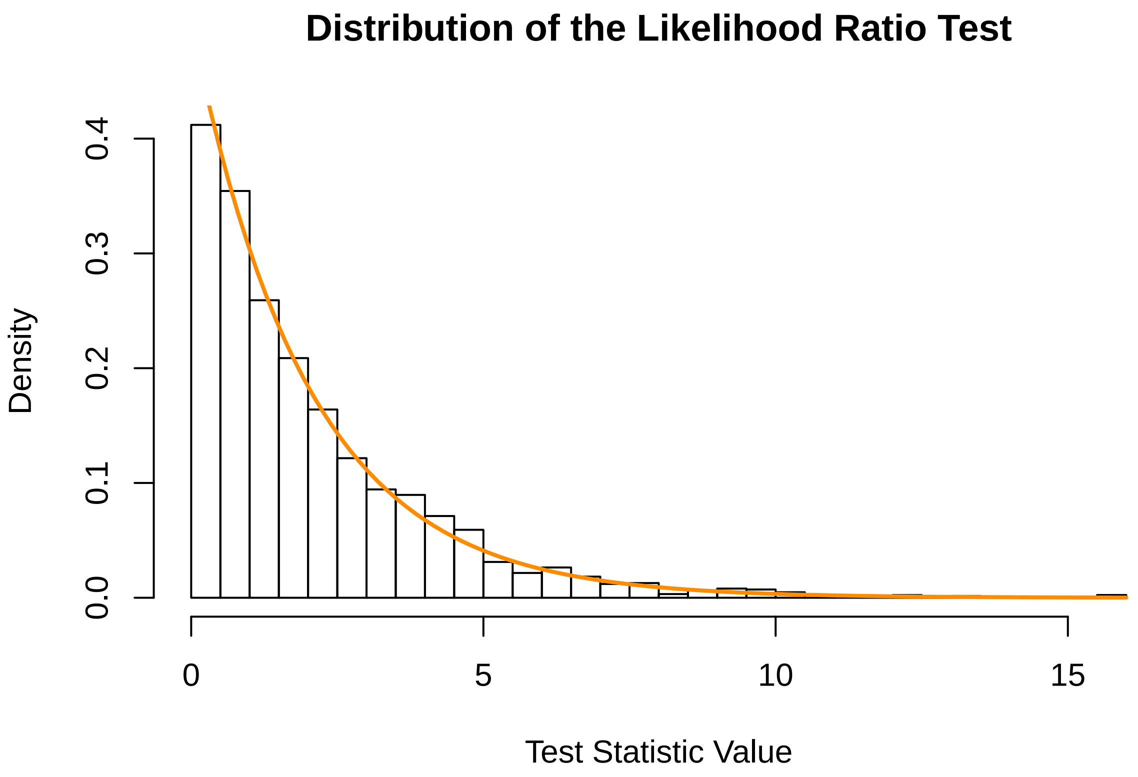
```r
1 - pnorm(1) # using standard normal
```

```
## [1] 0.1587
```

(d) Plot a histogram of the empirical values for the likelihood ratio test statistic. Overlay the density of the true distribution assuming a large sample.

Solution:

```r
hist(LRT_test_stat, breaks = 25, prob = TRUE,
     main = "Distribution of the Likelihood Ratio Test", xlab = "Test Statistic Value")
curve(dchisq(x, df = 2), add = TRUE, col = "darkorange", lwd = 2)
```



(e) Use the empirical results for the likelihood ratio test statistic to estimate the probability of observing a test statistic larger than 5. Also report this probability using the true distribution of the test statistic assuming a large sample.

Solution:

```
mean(LRT_test_stat > 5) # using empirical
```

```
## [1] 0.0828
```

```
1 - pchisq(5, df = 2) # using chi-square
```

```
## [1] 0.08208
```

(f) Repeat **(a)**-**(e)** but with simulation using a smaller sample size of 10. Based on these results, is this sample size large enough to use the standard normal and χ^2 distributions in this situation? Explain.

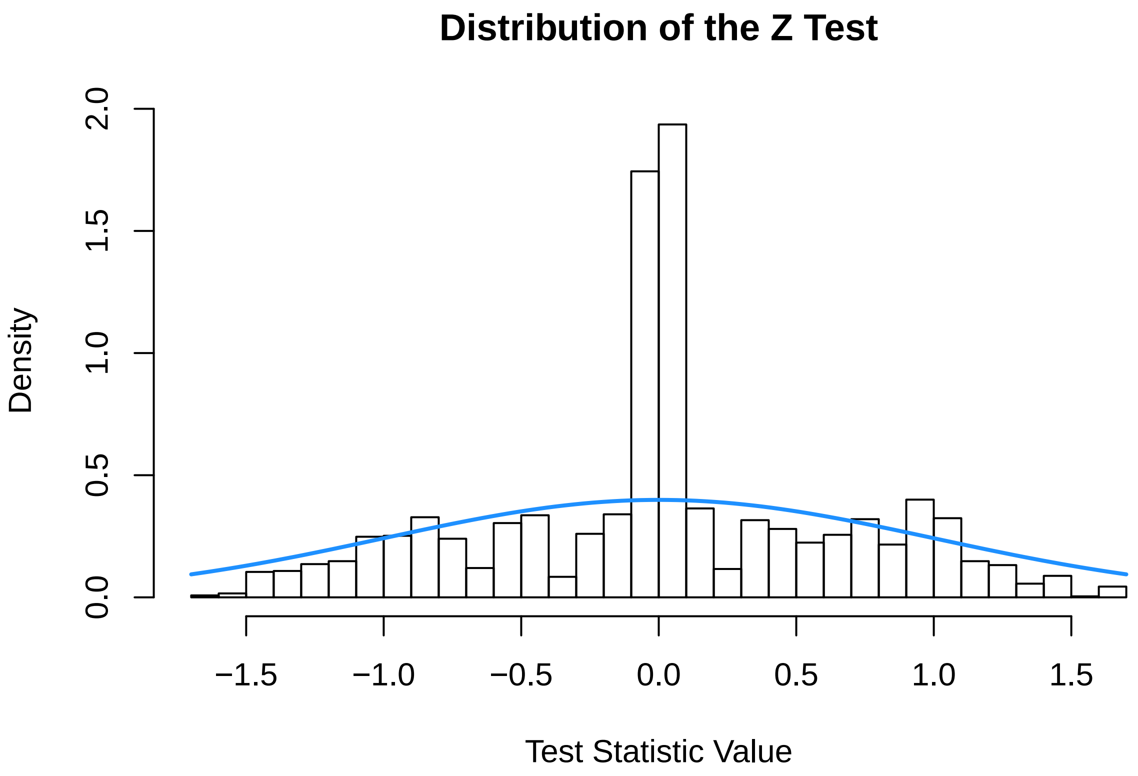
```
sample_size = 10
set.seed(420)
x1 = rnorm(n = sample_size)
x2 = rnorm(n = sample_size)
x3 = rnorm(n = sample_size)
```

Solution:

```
# setup
eta = 0.4 - 0.35 * x1
p = 1 / (1 + exp(-eta))
y = 0
sim_data = data.frame(y, x1, x2, x3)
num_sims = 2500
LRT_test_stat = rep(0, num_sims)
wald_test_stat = rep(0, num_sims)
```

```r
# run simulations
for (i in seq_along(LRT_test_stat)) {

  # simulate response data
  sim_data$y = rbinom(n = sample_size, 1, prob = p)

  # fit requested models
  fit_0 = glm(y ~ x1, data = sim_data, family = "binomial")
  fit_1 = glm(y ~ ., data = sim_data, family = "binomial")

  # store statistics
  LRT_test_stat[i] = anova(fit_0, fit_1, test = "Chisq")[2, "Deviance"]
  wald_test_stat[i] = coef(summary(fit_1))["x2", "z value"]

}
```

```r
hist(wald_test_stat, breaks = 25, prob = TRUE,
     main = "Distribution of the Z Test", xlab = "Test Statistic Value")
curve(dnorm(x), add = TRUE, col = "dodgerblue", lwd = 2)
```



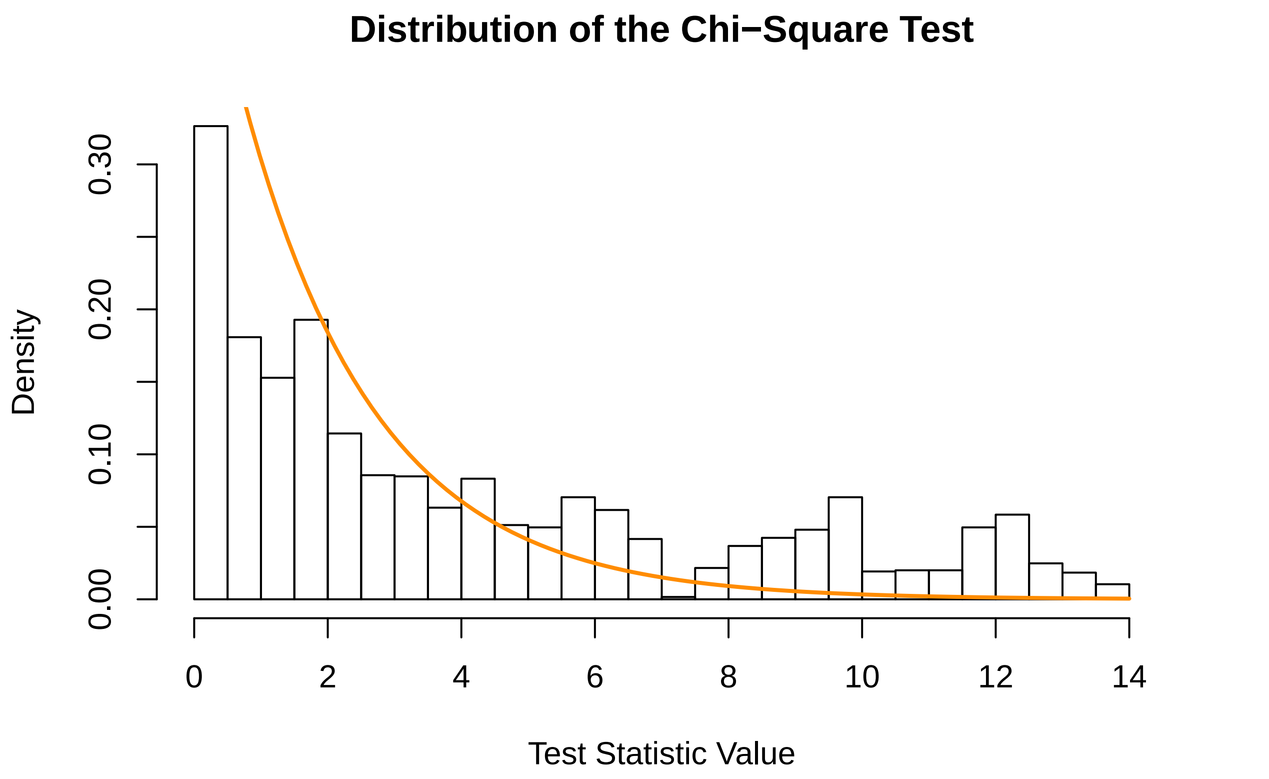
```r
mean(wald_test_stat > 1) # using empirical
```

```
## [1] 0.0796
```

```r
1 - pnorm(1) # using standard normal
```

```
## [1] 0.1587
```

```r
hist(LRT_test_stat, breaks = 25, prob = TRUE,
     main = "Distribution of the Chi-Square Test", xlab = "Test Statistic Value")
curve(dchisq(x, df = 2), add = TRUE, col = "darkorange", lwd = 2)
```



```r
mean(LRT_test_stat > 5) # using empirical
```

```
## [1] 0.3324
```

```r
1 - pchisq(5, df = 2) # using chi-square
```

```
## [1] 0.08208
```

It seems that 10 is not a large enough sample size to use the large sample distributional results. The empirical and true distributions seem to be rather far apart based on the histograms and probabilities.

Exercise 2 (Surviving the Titanic)

For this exercise use the `ptitanic` data from the `rpart.plot` package. (The `rpart.plot` package depends on the `rpart` package.) Use `?rpart.plot::ptitanic` to learn about this dataset. We will use logistic regression to help predict which passengers aboard the Titanic will survive based on various attributes.

```r
# install.packages("rpart")
# install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
data("ptitanic")
```

For simplicity, we will remove any observations with missing data. Additionally, we will create a test and train dataset.

```r
ptitanic = na.omit(ptitanic)
set.seed(42)
```

```r
trn_idx = sample(nrow(ptitanic), 300)
ptitanic_trn = ptitanic[trn_idx, ]
ptitanic_tst = ptitanic[-trn_idx, ]
```

(a) Consider the model

$$\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_3 x_4$$

where

$$p(\mathbf{x}) = P[Y = 1 \mid \mathbf{X} = \mathbf{x}]$$

is the probability that a certain passenger survives given their attributes and

- x_1 is a dummy variable that takes the value 1 if a passenger was 2nd class.
- x_2 is a dummy variable that takes the value 1 if a passenger was 3rd class.
- x_3 is a dummy variable that takes the value 1 if a passenger was male.
- x_4 is the age in years of a passenger.

Fit this model to the training data and report its deviance.

Solution:

```r
fit = glm(survived ~ pclass + sex + age + sex:age,
          data = ptitanic_trn, family = "binomial")
deviance(fit)
```

```
## [1] 281.5
```

(b) Use the model fit in **(a)** and an appropriate statistical test to determine if class played a significant role in surviving on the Titanic. Use $\alpha = 0.01$. Report:

- The null hypothesis of the test
- The test statistic of the test
- The p-value of the test
- A statistical decision
- A practical conclusion

Solution:

```r
fit_reduced = glm(survived ~ sex + age + sex:age, data = ptitanic_trn, family = "binomial")
res_b = anova(fit_reduced, fit, test = "LRT")[2, ]
```

- Null: $H_0 : \beta_1 = \beta_2 = 0$
- Test Statistic: 30.6734
- P-value: 2.1846×10^{-7}
- Decision: Reject H_0.
- Conclusion: Class plays a significant role in survival aboard the Titanic.

(c) Use the model fit in **(a)** and an appropriate statistical test to determine if an interaction between age and sex played a significant role in surviving on the Titanic. Use $\alpha = 0.01$. Report:

- The null hypothesis of the test
- The test statistic of the test
- The p-value of the test
- A statistical decision
- A practical conclusion

Solution:

```r
res_c = coef(summary(fit))["sexmale:age", ]
```

- Null: $H_0 : \beta_5 = 0$
- Test Statistic: -2.4862
- P-value: 0.0129
- Decision: Fail to reject H_0.
- Conclusion: An interaction between age and sex did **not** play a significant role in surviving on the Titanic.

(d) Use the model fit in **(a)** as a classifier that seeks to minimize the misclassification rate. Classify each of the passengers in the test dataset. Report the misclassification rate, the sensitivity, and the specificity of this classifier. (Use survived as the positive class.)

Solution:

```r
# calculate specificity
calc_spec = function(predicted, actual) {
  tab = table(predicted, actual)
  tab[1, 1] / sum(tab[, 1])
}

# calculate sensitivity
calc_sens = function(predicted, actual) {
  tab = table(predicted, actual)
  tab[2, 2] / sum(tab[, 2])
}
```

```r
ptitanic_tst_class = ifelse(predict(fit, ptitanic_tst) > 0, "survived", "died")

results = data.frame(
  "Metric" = c("Misclass", "Sensitivity", "Specificity"),
  "Value" = c(mean(ptitanic_tst_class != ptitanic_tst$survived),
              calc_sens(ptitanic_tst_class, ptitanic_tst$survived),
              calc_spec(ptitanic_tst_class, ptitanic_tst$survived))
)

knitr::kable(results)
```

Metric	Value
Misclass	0.2118
Sensitivity	0.7333
Specificity	0.8251

Exercise 3 (Breast Cancer Detection)

For this exercise we will use data found in `wisc-train.csv` and `wisc-test.csv`, which contain train and test data, respectively. `wisc.csv` is provided but not used. This is a modification of the Breast Cancer Wisconsin (Diagnostic) dataset from the UCI Machine Learning Repository. Only the first 10 feature variables have been provided. (And these are all you should use.)

- UCI Page

- Data Detail

You should consider coercing the response to be a factor variable if it is not stored as one after importing the data.

(a) The response variable `class` has two levels: `M` if a tumor is malignant, and `B` if a tumor is benign. Fit three models to the training data.

- An additive model that uses `radius`, `smoothness`, and `texture` as predictors
- An additive model that uses all available predictors
- A model chosen via backwards selection using AIC. Use a model that considers all available predictors as well as their two-way interactions for the start of the search.

For each, obtain a 5-fold cross-validated misclassification rate using the model as a classifier that seeks to minimize the misclassification rate. Based on this, which model is best? Relative to the best, are the other two underfitting or over fitting? Report the test misclassification rate for the model you picked as the best.

Solution:

```r
wisc_train = read.csv("wisc-train.csv")
wisc_test = read.csv("wisc-test.csv")
tibble::as.tibble(wisc_train)
```

```
## # A tibble: 100 x 11
##    class  radius texture perim~  area smooth~ compa~ conca~ conca~ symm~ fract~
##    <fctr>  <dbl>   <dbl>  <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>
##  1 M        19.7    21.2  130    1203  0.110  0.160  0.197  0.128  0.207 0.0600
##  2 M        11.4    20.4   77.6   386  0.142  0.284  0.241  0.105  0.260 0.0974
##  3 B        13.1    15.7   85.6   520  0.108  0.127  0.0457 0.0311 0.197 0.0681
##  4 M        15.3    14.3  102     704  0.107  0.214  0.208  0.0976 0.252 0.0703
##  5 M        17.1    16.4  116     913  0.119  0.228  0.223  0.140  0.304 0.0741
##  6 M        18.6    20.2  122    1094  0.0944 0.107  0.149  0.0773 0.170 0.0570
##  7 M        19.3    26.5  128    1162  0.0940 0.172  0.166  0.0759 0.185 0.0626
##  8 M        13.2    21.8   85.4   532  0.0971 0.105  0.0826 0.0525 0.175 0.0618
##  9 M        18.6    17.6  124    1076  0.110  0.169  0.197  0.101  0.191 0.0605
## 10 B        11.8    21.6   74.7   428  0.0864 0.0497 0.0166 0.0112 0.150 0.0589
## # ... with 90 more rows
```

```r
small = glm(class ~ radius + smoothness + texture, data = wisc_train,
            family = "binomial")
additive = glm(class ~ ., data = wisc_train, family = "binomial")
selected = step(glm(class ~ . ^ 2, data = wisc_train, family = "binomial"), trace = 0)
```

```r
library(boot)
results = data.frame(
  "Model" = c("`small`", "`additive`", "`selected`"),
  "CV-Misclass" = c(cv.glm(wisc_train, small, K = 5)$delta[1],
                    cv.glm(wisc_train, additive, K = 5)$delta[1],
                    cv.glm(wisc_train, selected, K = 5)$delta[1]),
  "Result" = c("Best", "Overfit", "Overfit")
)
knitr::kable(results)
```

Model	CV.Misclass	Result
`small`	0.0689	Best
`additive`	0.1222	Overfit
`selected`	0.1501	Overfit

```r
# test misclassification
mean(ifelse(predict(small, wisc_test) > 0, "M", "B") != wisc_test$class)
```

```
## [1] 0.08955
```

(b) In this situation, simply minimizing misclassifications might be a bad goal since false positives and false negatives carry very different consequences. Consider the `M` class as the "positive" label. Consider each of the probabilities stored in `cutoffs` in the creation of a classifier using the **additive** model fit in **(a)**.

```r
cutoffs = seq(0.01, 0.99, by = 0.01)
```

That is, consider each of the values stored in `cutoffs` as c. Obtain the sensitivity and specificity in the test set for each of these classifiers. Using a single graphic, plot both sensitivity and specificity as a function of the cutoff used to create the classifier. Based on this plot, which cutoff would you use? (0 and 1 have not been considered for coding simplicity. If you like, you can instead consider these two values.)

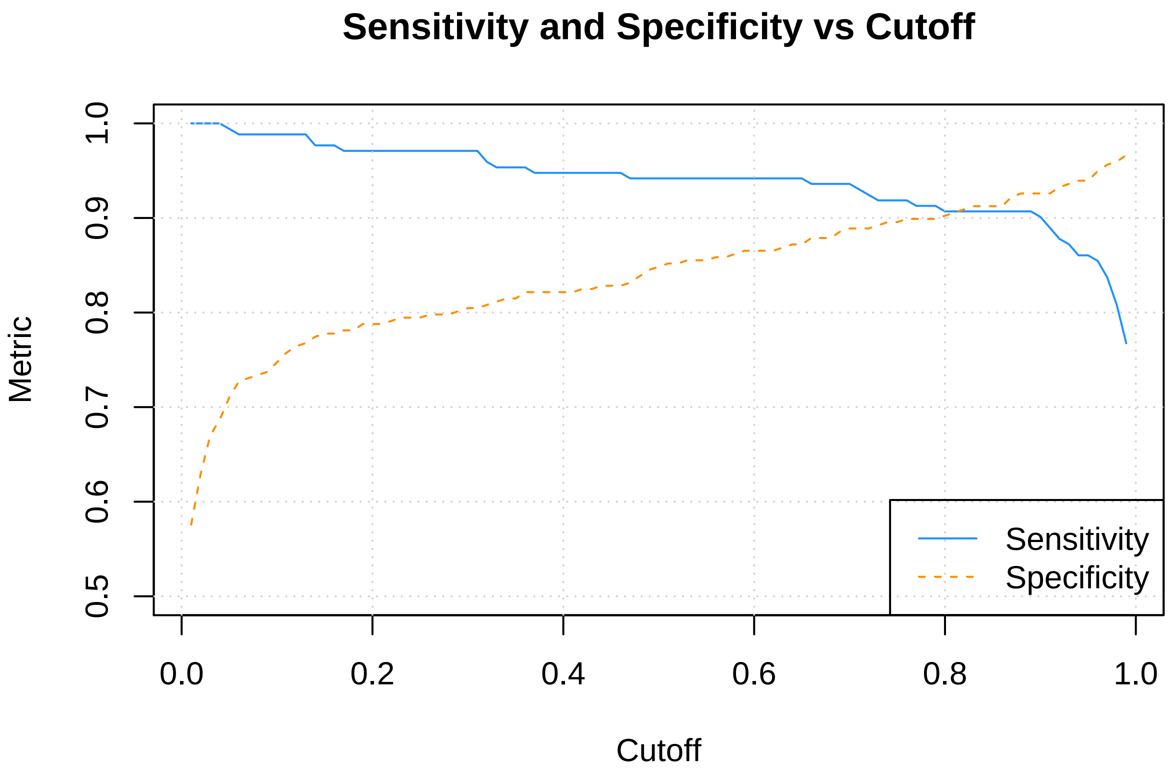
$$\hat{C}(\mathbf{x}) = \begin{cases} 1 & \hat{p}(\mathbf{x}) > c \\ 0 & \hat{p}(\mathbf{x}) \le c \end{cases}$$

Solution:

```r
# obtain predicted probabilities in the test set
wisc_probs = predict(additive, wisc_test, type = "response")

# make classifications for different probability cutoffs
make_class = function(probs, cutoff = 0.5) {
  ifelse(probs > cutoff, "M", "B")
}

# calculate specificity
calc_spec = function(predicted, actual) {
  tab = table(predicted, actual)
  tab[1, 1] / sum(tab[, 1])
}

# calculate sensitivity
calc_sens = function(predicted, actual) {
  tab = table(predicted, actual)
  tab[2, 2] / sum(tab[, 2])
}
```

```r
# setup
cutoffs = seq(0.01, 0.99, by = 0.01)
sens = rep(0, length(cutoffs))
spec = rep(0, length(cutoffs))

# calculations
for (i in seq_along(cutoffs)) {
  classifications = make_class(probs = wisc_probs, cutoff = cutoffs[i])
  truth = wisc_test$class
  sens[i] = calc_sens(classifications, truth)
  spec[i] = calc_spec(classifications, truth)
}
```

```r
# plotting
plot(cutoffs, sens, type = "l", ylim = c(0.5, 1), col = "dodgerblue",
     xlab = "Cutoff", ylab = "Metric", main = "Sensitivity and Specificity vs Cutoff")
grid()
lines(cutoffs, spec, type = "l", col = "darkorange", lty = 2)
legend("bottomright", col = c("dodgerblue", "darkorange"), lty = c(1, 2),
       legend = c("Sensitivity", "Specificity"))
```



Based on these results, the only way to prevent failing to detect a malignant tumor would be to use the lowest possible cutoff, thus classifying all observations as malignant. You might have a different tolerance for allowing cancer to go unnoticed, but in general, we should reduce the cutoff from the usual 0.5.

Note that in a machine learning course you will learn about ROC curves, which can be used to perform a similar analysis.