Week 12 - Homework

STAT 420, Fall 2024, D. 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.

```
sample_size = 150
set.seed(120)
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:

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

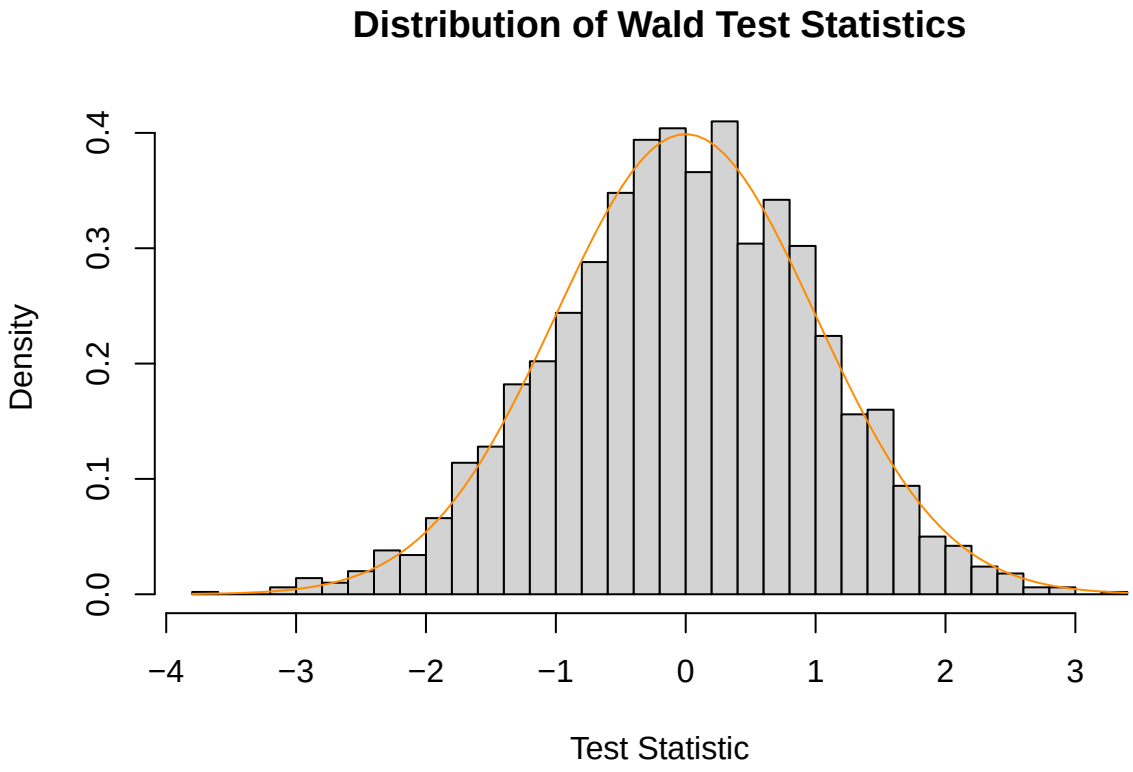
Store the test statistics for two tests:

- 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

```r
set.seed(120)
sample_size = 150
n_sims = 2500

simulate_and_test = function(sample_size, beta0=0.4, beta1=-0.35) {
    x1 = rnorm(sample_size)
    x2 = rnorm(sample_size)
    x3 = rnorm(sample_size)

    logit = beta0 + beta1 * x1
    p = exp(logit) / (1 + exp(logit))

    y = rbinom(sample_size, size=1, prob=p)

    full_model = glm(y ~ x1 + x2 + x3, family=binomial)

    reduced_model = glm(y ~ x1, family=binomial)

    wald_stat = coef(summary(full_model))[3, "z value"]

    lrt_stat = 2 * (logLik(full_model) - logLik(reduced_model))

    return(c(wald_stat, lrt_stat))
}

results = replicate(n_sims, simulate_and_test(sample_size))
wald_stats = results[1,]
lrt_stats = results[2,]
```

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

```r
hist(wald_stats, freq=FALSE, breaks=30,
     main="Distribution of Wald Test Statistics",
     xlab="Test Statistic")
curve(dnorm, add=TRUE, col="darkorange")
```



(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.

```
emp_prob_wald = mean(wald_stats > 1)
emp_prob_wald
```

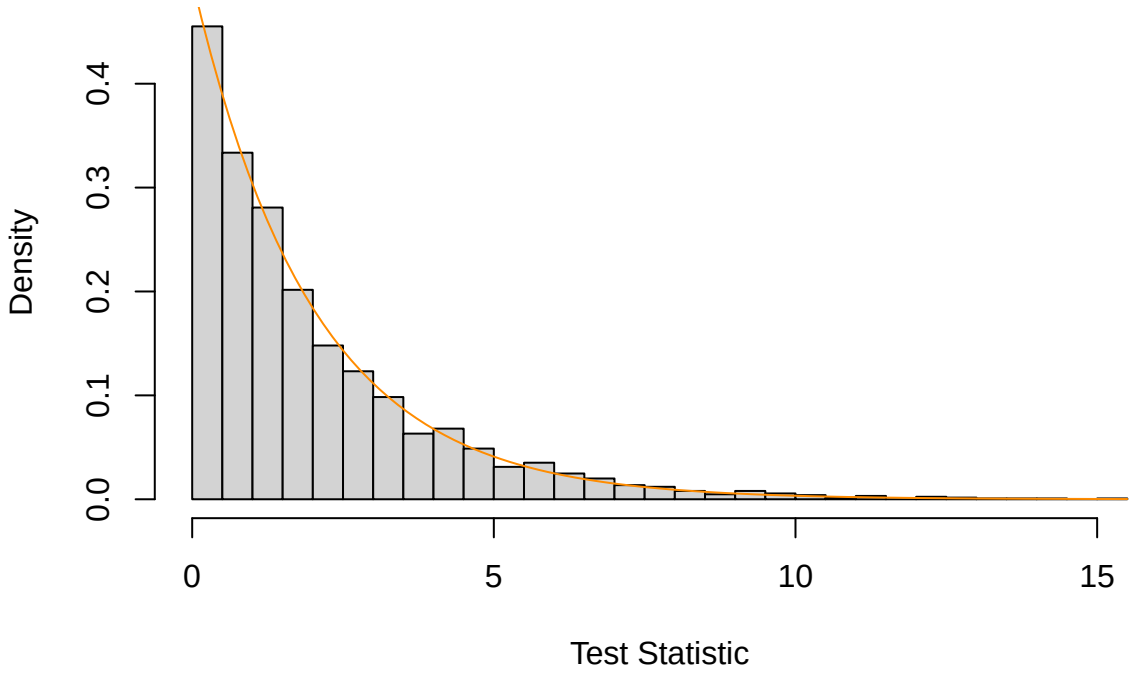
```
## [1] 0.1564
```

```
theo_prob_wald = 1 - pnorm(1)
theo_prob_wald
```

```
## [1] 0.1586553
```

(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.

```
hist(lrt_stats, freq=FALSE, breaks=30,
     main="Distribution of Likelihood Ratio Test Statistics",
     xlab="Test Statistic")
curve(dchisq(x, df=2), add=TRUE, col="darkorange")
```



(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.

```
emp_prob_lrt = mean(lrt_stats > 5)
emp_prob_lrt
```

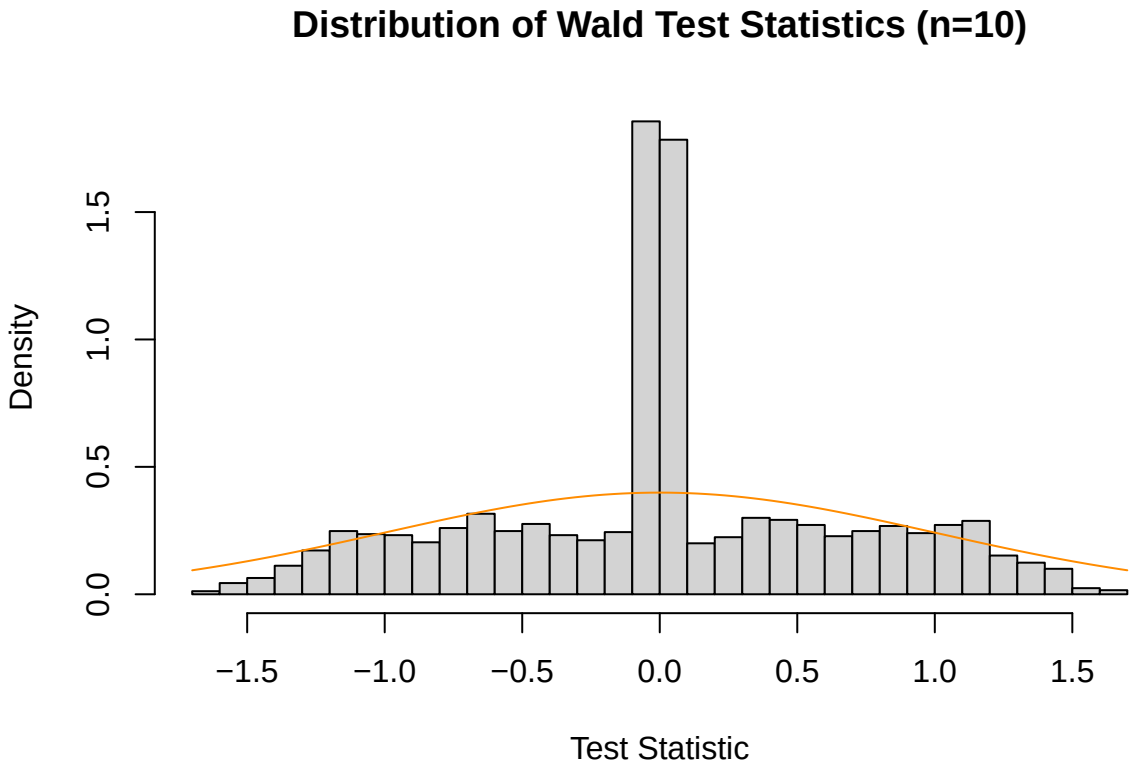
```
## [1] 0.0896
```

```
theo_prob_lrt = 1 - pchisq(5, df=2)
theo_prob_lrt
```

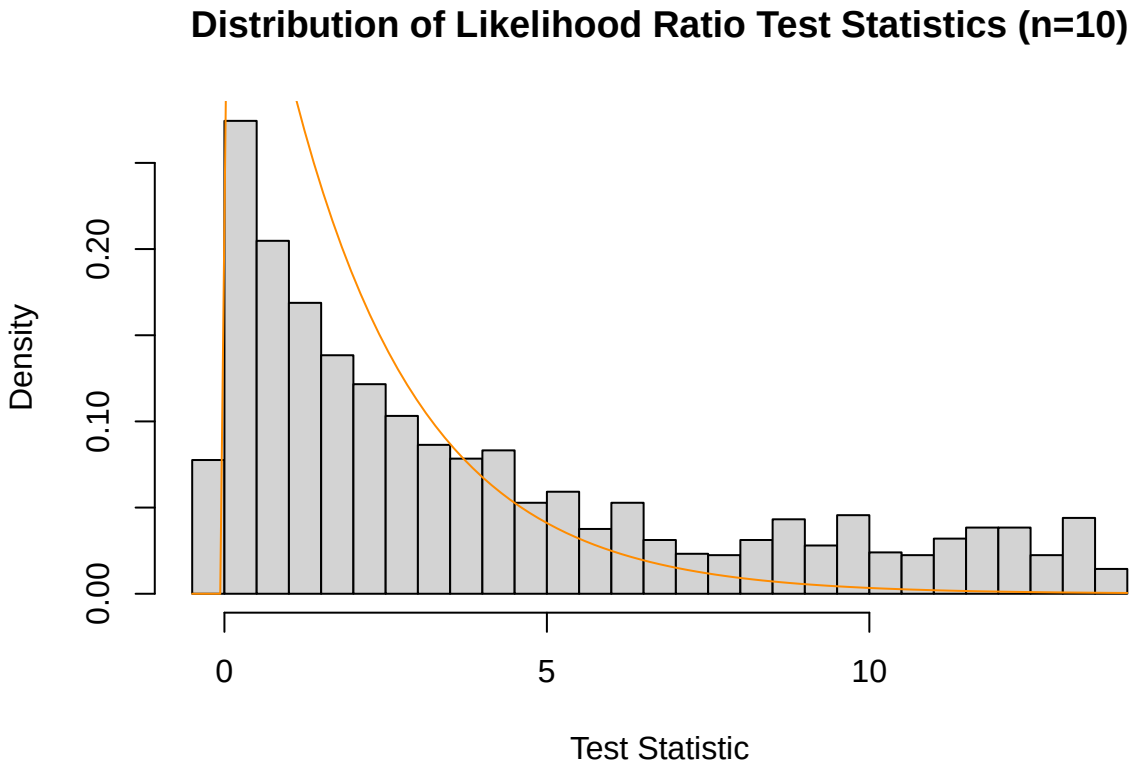
```
## [1] 0.082085
```

(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.

```
set.seed(120)
sample_size = 10
results_small = replicate(n_sims, simulate_and_test(sample_size))
wald_stats_small = results_small[1,]
lrt_stats_small = results_small[2,]

hist(wald_stats_small, freq=FALSE, breaks=30,
     main="Distribution of Wald Test Statistics (n=10)",
     xlab="Test Statistic")
curve(dnorm, add=TRUE, col="darkorange")
```




```r
hist(lrt_stats_small, freq=FALSE, breaks=30,
     main="Distribution of Likelihood Ratio Test Statistics (n=10)",
     xlab="Test Statistic")
curve(dchisq(x, df=2), add=TRUE, col="darkorange")
```



```
emp_prob_wald_small = mean(wald_stats_small > 1)
theo_prob_wald_small = 1 - pnorm(1)
emp_prob_lrt_small = mean(lrt_stats_small > 5)
theo_prob_lrt_small = 1 - pchisq(5, df=2)

cat("Small sample (n=10) results:\n")
```

```
## Small sample (n=10) results:
```

```
cat("Empirical probability (Wald > 1):", emp_prob_wald_small, "\n")
```

```
## Empirical probability (Wald > 1): 0.0976
```

```
cat("Theoretical probability (Wald > 1):", theo_prob_wald_small, "\n")
```

```
## Theoretical probability (Wald > 1): 0.1586553
```

```
cat("Empirical probability (LRT > 5):", emp_prob_lrt_small, "\n")
```

```
## Empirical probability (LRT > 5): 0.3052
```

```
cat("Theoretical probability (LRT > 5):", theo_prob_lrt_small, "\n")
```

```
## Theoretical probability (LRT > 5): 0.082085
```

This sample size is not large enough to use standard normal and and χ^2 distributions in this situation. Due to small sample size, the models fitted with this sample data vary among different iterations, leading to non-normal distribution of Wald test and Likelihood Ratio Test.

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.

```
# 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.

```
ptitanic = na.omit(ptitanic)
set.seed(2021)
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.

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

```
## [1] 259.1653
```

(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

```
reduced_model_class = glm(survived ~ sex + age + sex:age,
                          data = ptitanic_trn,
                          family = binomial)

lrt_stats = 2 * (logLik(model) - logLik(reduced_model_class))
df = length(coef(model)) - length(coef(reduced_model_class))
p_value = 1 - pchisq(lrt_stats, df)
```

- The null hypothesis of the test: $\beta_1 = \beta_2 = 0$ (class has no effect on survival)
- The test statistic of the test: 45.0617314
- The p-value of the test: $1.6404744 \times 10^{-10}$
- A statistical decision: since $1.6404744 \times 10^{-10} < \alpha = 0.01$, we reject the null hypothesis, meaning that we will choose the larger model.
- A practical conclusion: There is strong statistical evidence that a passenger's class had a significant effect on their probability of survival on 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

```
reduced_model_interact = glm(survived ~ pclass + sex + age,
                             data = ptitanic_trn,
                             family = binomial)

lrt_stats = 2 * (logLik(model) - logLik(reduced_model_interact))
```

```
df = length(coef(model)) - length(coef(reduced_model_interact))
p_value = 1 - pchisq(lrt_stats, df)
```

- The null hypothesis of the test: $\beta_5 = 0$ (interaction between age and sex has no effect on survival)
- The test statistic of the test: 11.3716894
- The p-value of the test: 7.4572022×10^{-4}
- A statistical decision: since $7.4572022 \times 10^{-4} < \alpha = 0.01$, we reject the null hypothesis, meaning that we will choose the larger model.
- A practical conclusion: There is strong statistical evidence that the interaction between age and sex had a significant effect on their probability of survival 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.)

```
pred_probs = predict(model, newdata = ptitanic_tst, type = "response")
predictions = ifelse(pred_probs > 0.5, 1, 0)

conf_matrix = table(Actual = ptitanic_tst$survived, Predicted = predictions)
misclass_rate = 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
misclass_rate
```

```
## [1] 0.2171582
```

```
sensitivity = conf_matrix[2,2] / sum(conf_matrix[2,])
sensitivity
```

```
## [1] 0.5714286
```

```
specificity = conf_matrix[1,1] / sum(conf_matrix[1,])
specificity
```

```
## [1] 0.937355
```

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.

```r
library(boot)

train_data = read.csv("wisc-train.csv")
test_data = read.csv("wisc-test.csv")

train_data$class = factor(train_data$class)
test_data$class = factor(test_data$class)

get_misclass_rate = function(model, data) {
  preds = predict(model, data, type = "response")
  pred_class = ifelse(preds > 0.5, "M", "B")
  mean(pred_class != data$class)
}

cv_misclass = function(model_formula, data, k = 5) {
  cost = function(y, pred) {
    pred_class = ifelse(pred > 0.5, "M", "B")
    mean(pred_class != y)
  }

  glm_fit = glm(model_formula, data = data, family = binomial)
  cv.glm(data, glm_fit, cost, K = k)
}

model1_formula = class ~ radius + smoothness + texture
model1 = glm(model1_formula, data = train_data, family = binomial)
cv_error1 = cv_misclass(model1_formula, train_data)

model2_formula = class ~ .
model2 = glm(model2_formula, data = train_data, family = binomial)
cv_error2 = cv_misclass(model2_formula, train_data)

model3_formula = class ~ .^2
model3_start = glm(model3_formula, data = train_data, family = binomial)

model3 = step(model3_start, trace = 0)
cv_error3 = cv_misclass(formula(model3), train_data)

test_errors = c(
  get_misclass_rate(model1, test_data),
  get_misclass_rate(model2, test_data),
  get_misclass_rate(model3, test_data)
)

names(test_errors) = c("Model1", "Model2", "Model3")
test_errors
```

```
##     Model1     Model2     Model3
## 0.08955224 0.11727079 0.15138593
```

The first model is the best, with other two overfitting. Its test misclassification rate is 0.08955224.

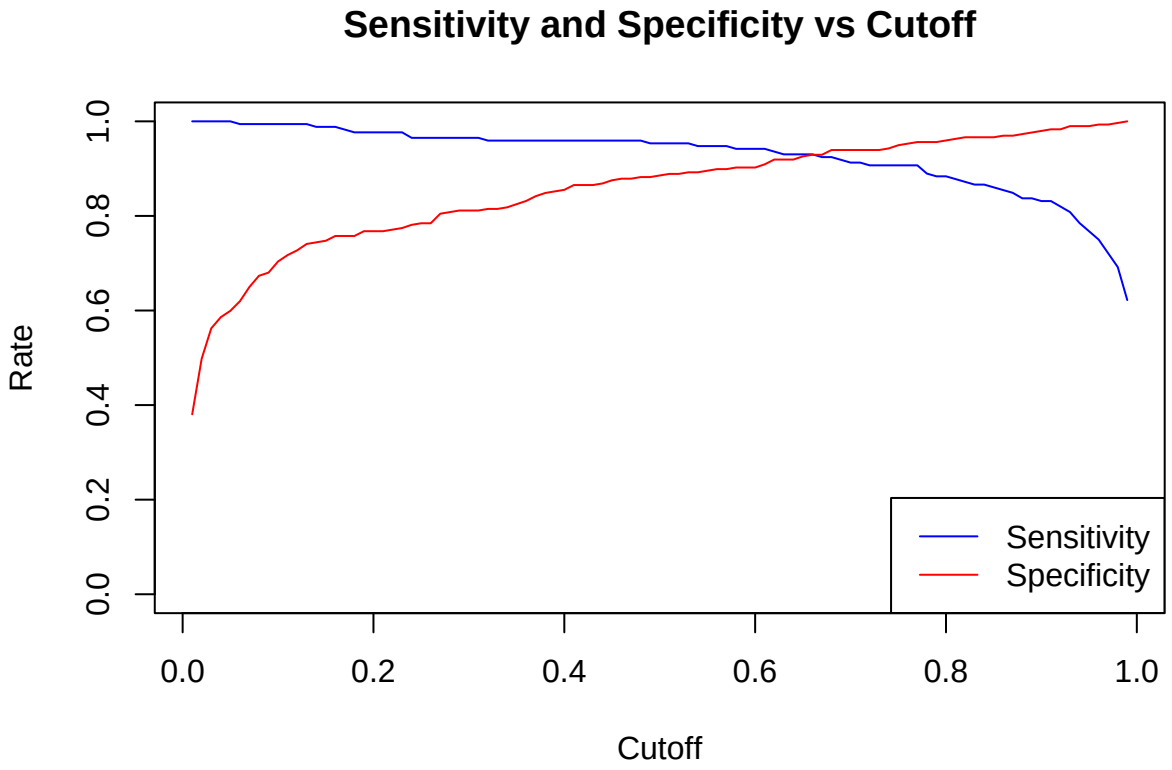
(b) In this situation, simply minimizing misclassifications might be a bad goal since false positives and false negatives carry very different consequences. Consider the `M` class as the "positive" label. Consider each of the probabilities stored in `cutoffs` in the creation of a classifier using the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

```
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.)

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

```
pred_probs = predict(model1, test_data, type = "response")

sensitivity = numeric(length(cutoffs))
specificity = numeric(length(cutoffs))

for (i in seq_along(cutoffs)) {
    pred_class = ifelse(pred_probs > cutoffs[i], "M", "B")

    tp = sum(pred_class == "M" & test_data$class == "M")
    tn = sum(pred_class == "B" & test_data$class == "B")
    fp = sum(pred_class == "M" & test_data$class == "B")
    fn = sum(pred_class == "B" & test_data$class == "M")

    sensitivity[i] = tp / (tp + fn)
    specificity[i] = tn / (tn + fp)
}

plot(cutoffs, sensitivity, type = "l", col = "blue",
     xlab = "Cutoff", ylab = "Rate",
     main = "Sensitivity and Specificity vs Cutoff",
     ylim = c(0, 1))
lines(cutoffs, specificity, col = "red")
legend("bottomright", legend = c("Sensitivity", "Specificity"),
       col = c("blue", "red"), lty = 1)
```




```
closest_point = which.min(abs(sensitivity - specificity))
optimal_cutoff = cutoffs[closest_point]
```

- Optimal cutoff where sensitivity and specificity are most similar is: 0.66, which is the cutoff that I would use.
- At this cutoff:
  - Sensitivity: 0.9302326
  - Specificity: 0.9292929