Homework 7

Instructions

Due: 5pm on Wednesday, November 10th

1. Add your name between the quotation marks on the author line in the YAML above.
2. Compose your answer to each problem between the bars of red stars.
3. Commit your changes frequently.
4. Be sure to knit your .Rmd to a .pdf file.
5. Push both the .pdf and the .Rmd file to your repo on the class organization before the deadline.

Theory

Problem 1

Based on ISLR Exercise 4.3

This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature.

Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X \sim N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is not linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.2

The Bayes Classifier assigns an observation x to the class A_j for which $P(Y = A_j|X = x)$ is largest. Assume that $X|Y = j \sim N(\mu_j, \sigma_j^2)$. Using Bayes' Rule

$$P(Y = A_j|X = x) = \frac{P(X = x|Y = A_j)P(Y = A_j)}{P(X = x)}$$

Since the denominator $P(X = x)$ is the same for all conditional probabilities $P(Y = A_j|X = x)$, and since the natural log is an increasing function, in order to find the value of j that maximizes these probabilities, it suffices to maximize the natural log of the numerator. Using the formula for the Normal density,

$$P(X = x|Y = A_j)P(Y = A_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right)\pi_j$$

And

$$\begin{aligned}\ln P(X = x|Y = A_j)P(Y = A_j) &= -\frac{(x-\mu_j)^2}{2\sigma_j^2} - \ln\sigma_j - \frac{1}{2}\ln 2\pi + \ln\pi_j \\ &= -\frac{x^2}{2\sigma_j^2} + \frac{2x\mu_j}{\sigma_j^2} - \frac{\mu_j^2}{2\sigma_j^2} - \ln\sigma_j - \frac{1}{2}\ln 2\pi + \ln\pi_j\end{aligned}$$

Dropping the $-\frac{1}{2}\ln 2\pi$ term (since it does not depend on j), we see that $\ln P(Y = A_j|X = x)$ maximized exactly when

$$\delta_j(x) = -\frac{x^2}{2\sigma_j^2} + \frac{2x\mu_j}{\sigma_j^2} - \frac{\mu_j^2}{2\sigma_j^2} - \ln\sigma_j + \ln\pi_j$$

is maximized. This a quadratic equation in x.

Problem 2

Based on ISLR Exercise 4.7

Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn't was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year.

Hint: Use Bayes' Theorem.

Assume that $X|Y = \text{Yes} \sim N(\mu_y, \sigma)$ and $X|Y = \text{No} \sim N(\mu_n, \sigma)$. Based on the data, our estimates for the Normal parameters are:

$$\hat{\mu}_y = 10 \quad \hat{\mu}_n = 0 \quad \hat{\sigma}^2 = 36$$

And our estimates for the prior π_y and π_n are

$$\hat{\pi}_y = 0.8 \quad \hat{\pi}_n = 0.2$$

By Bayes' theorem and the formula for the Normal density,

$$\begin{aligned}P(Y = \text{Yes}|X = 4) =& \frac{P(X = 4|Y = \text{Yes})\pi_y}{P(X = 4|Y = \text{Yes})\pi_y + P(X = 4|Y = \text{No})\pi_n} \\ =& \frac{\frac{1}{\sqrt{2\pi\cdot 36}}\exp\left(-\frac{(4-10)^2}{2\cdot 36}\right)\cdot 0.8}{\frac{1}{\sqrt{2\pi\cdot 36}}\exp\left(-\frac{(4-10)^2}{2\cdot 36}\right)\cdot 0.8 + \frac{1}{\sqrt{2\pi\cdot 36}}\exp\left(-\frac{(4-0)^2}{2\cdot 36}\right)\cdot 0.2} \\ =& \frac{\exp\left(-\frac{36}{2\cdot 36}\right)\cdot 0.8}{\exp\left(-\frac{36}{2\cdot 36}\right)\cdot 0.8 + \exp\left(-\frac{16}{2\cdot 36}\right)\cdot 0.2} \\ \approx& 0.7519\end{aligned}$$

Applied

Problem 3

Based on ISLR Exercise 4.11

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set from the `ISLR` package, which can be loaded with the following code:

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.4
```

```
data(Auto)
```

a. Mutate the `Auto` data frame to create a binary variable `mpg01` that takes the value 1 if `mpg` is larger then the median value of `mpg` and 0 otherwise.

b. Produce some numerical and graphical summaries of the `Auto` data in order to investigate the relationship between `mpg01` and the other features. Discuss your findings. Which variables seem most useful in predicting `mpg01`?

c. Randomly divide the data into a training and a test set using a $75 - 25$ ratio. Be sure to set a seed for reproducibility.

d. Perform LDA on the training data to predict `mpg01` using the variables that seemed most associated with `mpg01` from part b. Then create a confusion matrix and compute the test error for the model. Finally, create an ROC curve for the model.

e. Perform QDA on the training data to predict `mpg01` using the variables that seemed most associated with `mpg01` from part b. Then create a confusion matrix and compute the test error for the model. Finally, create an ROC curve for the model.

f. Perform Logistic Regression on the training data to predict `mpg01` using the variables that seemed most associated with `mpg01` from part b. Then create a confusion matrix and compute the test error for the model. Finally, create an ROC curve for the model.

g. Perform KNN on the training data to predict `mpg01` using the variables that seemed most associated with `mpg01` from part b. Use cross-validation on the training set to select the optimal value of k. Then create a confusion matrix and compute the test error for the model with that value of k.

h. If you had to select *ONE* model to make predictions about `mpg01`, which would you use and why?

---

a.

```
M<-median(Auto$mpg)

Auto<-Auto %>% mutate(mpg01 = ifelse(mpg > M, "1", "0")) %>%
  mutate(mpg01 = as.factor(mpg01))

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 mpg01
```

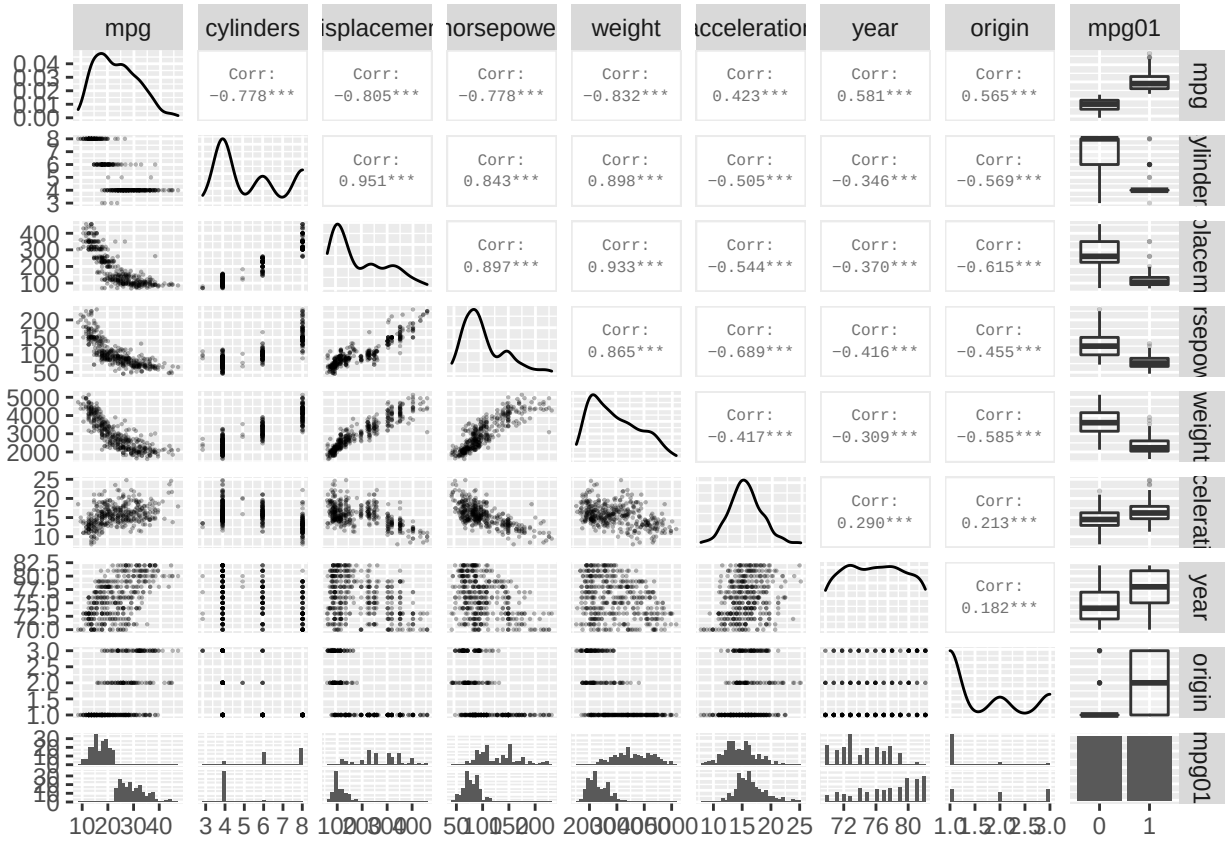
```
## 1 chevrolet chevelle malibu     0
## 2          buick skylark 320     0
## 3         plymouth satellite     0
## 4              amc rebel sst     0
## 5                ford torino     0
## 6           ford galaxie 500     0
```

b. We first create pairwise scatterplots for all predictors and the response.

```
library(GGally)

g<-ggpairs(Auto %>% dplyr::select(-name),
        lower = list(continuous = wrap("points", alpha = 0.3, size=0.1)),
        upper = list(combo = wrap("box_no_facet", alpha=0.25, outlier.size = .25),
                     continuous = wrap("cor", size=2)))

g
```



In particular, we are most interested in the boxplots showing the relationship between `mpg01` and other predictors. It appears `cylinders` `displacement`, `horsepower` and `weight` have greatest ratio of variance between classes to variance within classes, so we will use these variables as our primary predictors.

But it should be noted that displacement, weight and horsepower are very strongly correlated, so we should be careful of multicollinearity in our model.

c.

```
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.0.5
```

```
set.seed(199)
Auto_split <- initial_split(Auto, prop = .75)
Auto_train <- training(Auto_split)
Auto_test <- testing(Auto_split)
```

d. The LDA model also has an overall test error rate of 0.12

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
auto_lda<-lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto_train)
auto_lda_pred<-predict(auto_lda, Auto_test)
results_lda <- data.frame(obs = Auto_test$mpg01,
                          preds = auto_lda_pred$class,
                          probs = auto_lda_pred$posterior[,2])
```

```
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 4.0.5
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.
```

```
##
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

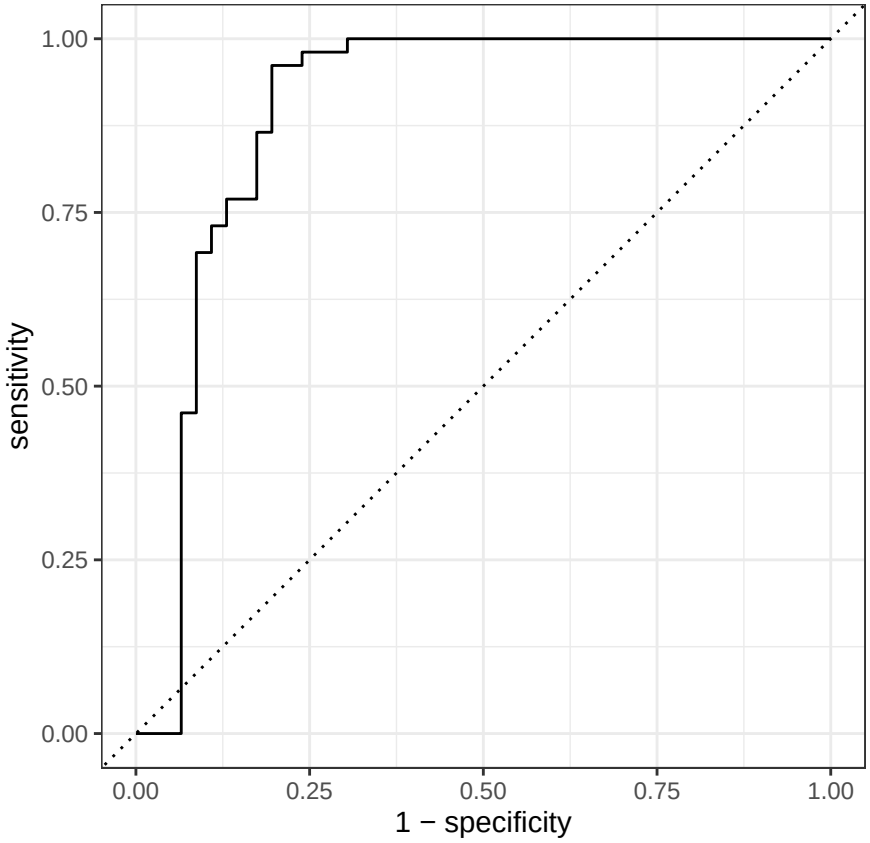
```
conf_mat(results_lda, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  0  1
##          0 37  3
##          1  9 49
```

```
accuracy(results_lda, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.878
```

```
r_lda <- roc_curve(results_lda, truth = obs, probs, event_level = "second")
autoplot(r_lda)
```



e. The QDA model also has an overall test error rate of 0.13

```
auto_qda<-qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto_train)
auto_qda_pred<-predict(auto_qda, Auto_test)
results_qda <- data.frame(obs = Auto_test$mpg01,
                          preds = auto_qda_pred$class,
                          probs = auto_qda_pred$posterior[,2])
```

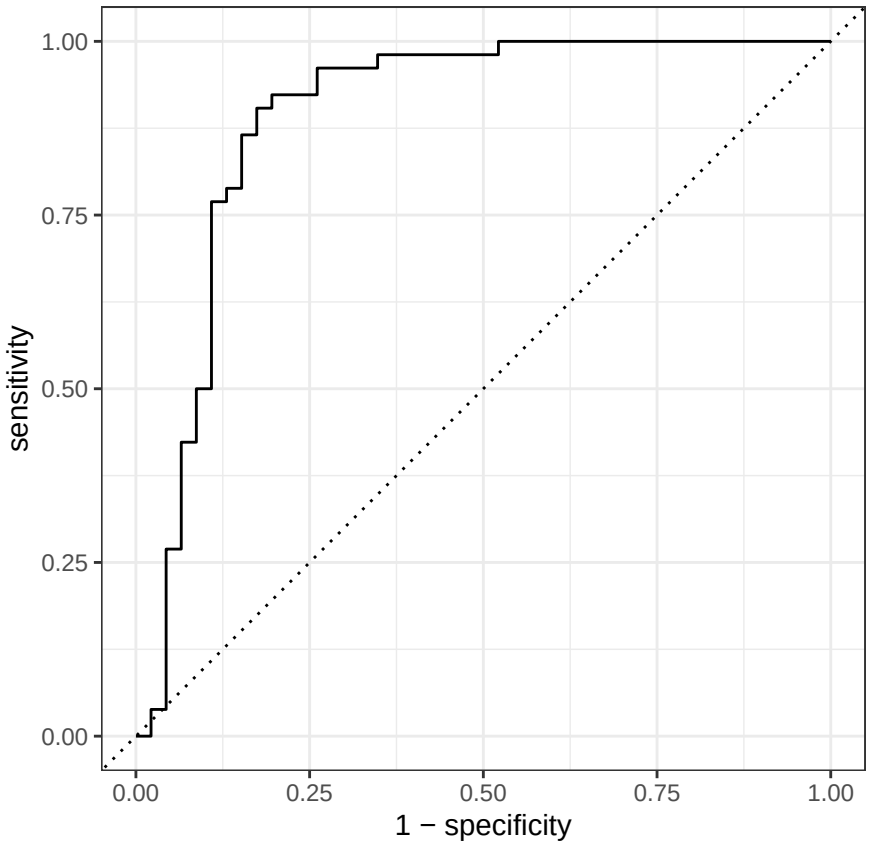
```
conf_mat(results_qda, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  0  1
##          0 37  4
##          1  9 48
```

```
accuracy(results_qda, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.867
```

```
r_qda <- roc_curve(results_qda, truth = obs, probs, event_level = "second")
autoplot(r_qda)
```



f. Note that neither cylinders nor displacement appear to be statistically significant predictors in the present of horsepower and weight, which isn't surprising given correlation observed in part (b). However, since both LDA and QDA models included these variables, we will include them for the Logistic Regression as well.

The test error rate for the logistic regression was about 0.14, slightly higher than the rate for both LDA and QDA.

```
auto_log <- glm(mpg01 ~ cylinders + displacement + horsepower + weight,
                data= Auto_train, family = "binomial")

auto_log_probs <- predict(auto_log, Auto_test, type = "response")
auto_log_preds <- as.factor(ifelse(auto_log_probs > 0.5, "1", "0"))

results_log <- data.frame(obs = Auto_test$mpg01, preds = auto_log_preds, probs = auto_log_probs)

summary(auto_log)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight, family = "binomial", data = Auto_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2891  -0.1657  -0.0053   0.3406   3.3975
##
```

```
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  12.6317406  2.0662828   6.113 9.76e-10 ***
## cylinders    -0.2741214  0.3969936  -0.690  0.48988
## displacement -0.0098289  0.0090539  -1.086  0.27765
## horsepower   -0.0344441  0.0169074  -2.037  0.04163 *
## weight       -0.0021696  0.0007888  -2.751  0.00595 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 407.45  on 293  degrees of freedom
## Residual deviance: 148.49  on 289  degrees of freedom
## AIC: 158.49
##
## Number of Fisher Scoring iterations: 7
```

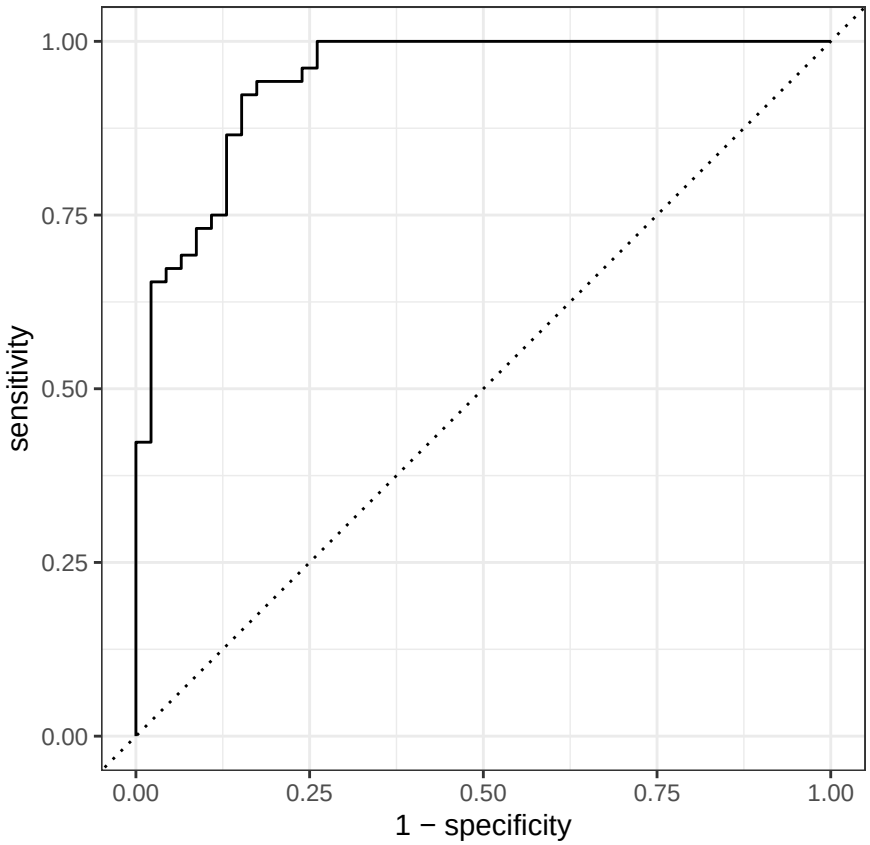
```
conf_mat(results_log, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  0  1
##          0 35  3
##          1 11 49
```

```
accuracy(results_log, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.857
```

```
r_log <- roc_curve(results_log, truth = obs, probs, event_level = "second")
autoplot(r_log)
```



g.

```
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.0.5
```

```
auto_knn_cv <- train.kknn(mpg01 ~ cylinders + displacement + horsepower + weight,
                          data = Auto_train, kmax = 20, kernel = "rectangular")

best_k <- auto_knn_cv$best.parameters[[2]]
```

Based on cross-validation, the optimal value of k = 7. Overall, knn achieved a test error rate of approximately 0.09.

```
auto_knn_mod <- kknn(mpg01 ~ cylinders + displacement + horsepower + weight,
                     train = Auto_train, test = Auto_test,
                     k = best_k, kernel = "rectangular")

auto_knn_preds <- auto_knn_mod$fitted.values
auto_knn_probs <- auto_knn_mod$prob[,2]

results_knn <- data.frame(obs = Auto_test$mpg01, preds = auto_knn_preds, probs = auto_knn_probs)
```

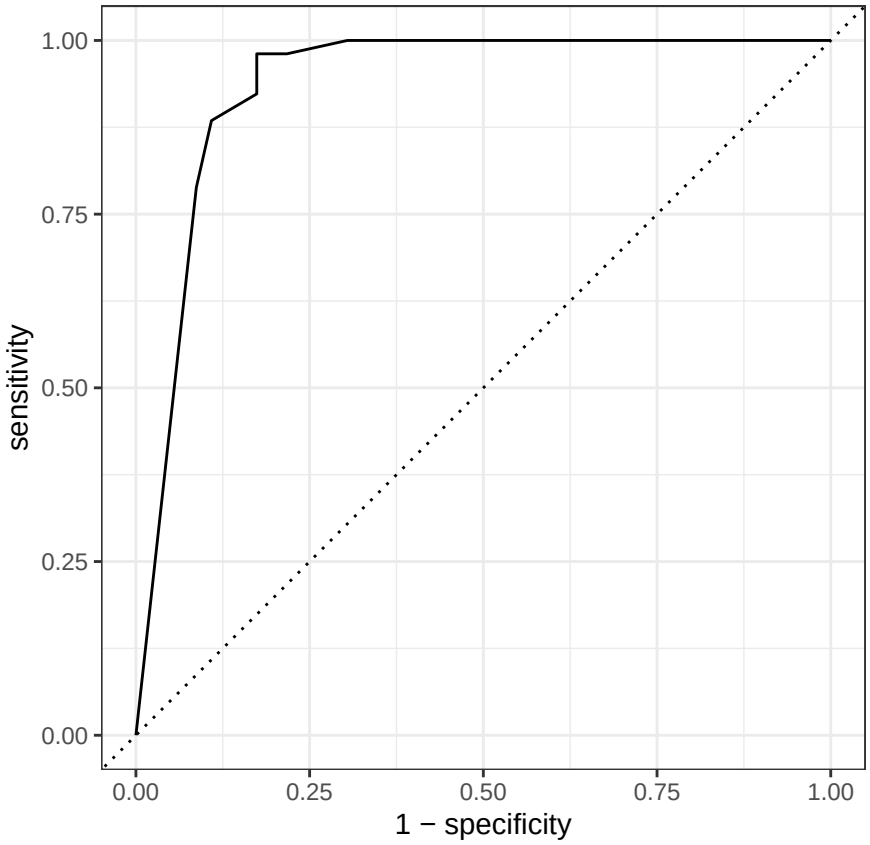
```
conf_mat(results_knn, truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  0  1
##          0 38  1
##          1  8 51
```

```r
accuracy(results_knn, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.908
```

```r
r_knn <- roc_curve(results_knn, truth = obs, probs, event_level = "second")
autoplot(r_knn)
```



h. KNN provided the best error rate on the test set. However, the difference in error rates between KNN, LDA, and QDA was relatively small. Based on the principle of parsimony, it may be most appropriate to choose the simplest model with comparable error. Here, this is the lda model.

Problem 4

For this exercise we will use data found in `wisc.csv`, which is a modified version of the Breast Cancer Wisconsin (Diagnostic) dataset from the UCI Machine Learning Repository.

The data set contains values of 11 variables computed from 569 digitized images of a fine needle aspirate of a breast mass.

- `radius` (mean of distances from center to points on the perimeter)
- `texture` (standard deviation of gray-scale values)
- `perimeter`
- `area`

- `smoothness` (local variation in radius lengths)
- `compactness` (perimeter^2 / area - 1.0)
- `concavity` (severity of concave portions of the contour)
- `concave` points (number of concave portions of the contour)
- `symmetry`
- `fractal` dimension ("coastline approximation" - 1)
- `class`, whether mass is malignant (M) or benign (B)

We are interested in models predicting the value of `class` based on other variables in the data set.

Decision Boundaries

a. Create a logistic regression model using **training** data with two predictors, `radius` and `symmetry`. Plot the **test** data with `radius` on the x axis and `symmetry` on the y axis, with points colored according to their tumor status. Use `geom_abline` to add a line representing the decision boundary for the logistic regression classifier, using 0.5 as the threshold for predicted probability. *Hint: If $p(X) = 0.5$, what must the log-odds be equal to?*

b. Based on a visual inspection of the graphic you created in the part (b), how well did the logistic regression model do?

c. Repeat parts (b) using 0.1 as the threshold for predicted probability. How did model accuracy change? Consider both the rate of false positives and the rate of false negatives.

A Handmade Model

d. Use an appropriate visualization to investigate the conditional distributions of each predictor (`symmetry`) given the two values of `class`. Does each appear to be approximately Normal? Do the two conditional distributions for `symmetry` appear to have the same variance?

e. Even if the conditions for LDA are not met, we will proceed as if they are. Create estimates for the mean for each conditional distribution and the pooled variance, and use the training proportions as estimates for the prior distribution of `class`.

f. Use the formula for the discriminant in Section 4.4 to write explicit linear equations for the discriminants for the two levels of `class`. Plot these two lines using `geom_abline`. Estimate the value of the decision "boundary" based on this graph and then compute the value exactly using the equations of your two lines.

g. Classify the points in the training set according to your decision boundary. What is your error rate?

h. **Challenge Problem** (This part is optional and requires some linear algebra) Create estimates for the mean of the conditional distribution of `radius` and estimate the covariance matrix for `symmetry` and `radius`. Then use the formula for the discriminant with $p = 2$ predictors (equation 4.19) to find explicit equations for the discriminant planes. Find the decision boundary corresponding to the intersection of these planes and add it to the scatterplot of `radius` and `symmetry` colored by `class`.

---

a.

To use logistic regression, we first code `class` as a numeric variable, with 1 indicating malignant `M` and 0 indicating benign `B`.

We then create a test-training split, using `strata` to ensure the response levels are reasonably represented in both sets.

```
wisc <- wisc %>% mutate(class01 = ifelse(class == "M", 1, 0 ))
library(rsample)
set.seed(2)
```

```
wisc_split <- initial_split(wisc, stata = class)
wisc_train <- training(wisc_split)
wisc_test <- testing(wisc_split)
```

```
wisc_log <- glm(class01 ~ radius + symmetry, data = wisc_train, family = "binomial")

summary(wisc_log)$coefficients
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -23.610671   2.417764 -9.765500 1.583294e-22
## radius        1.063489   0.112845  9.424327 4.328707e-21
## symmetry     43.622879   7.051288  6.186512 6.150985e-10
```

```
b0<-summary(wisc_log)$coefficients[1,1]
br<-summary(wisc_log)$coefficients[2,1]
bs<-summary(wisc_log)$coefficients[3,1]
```

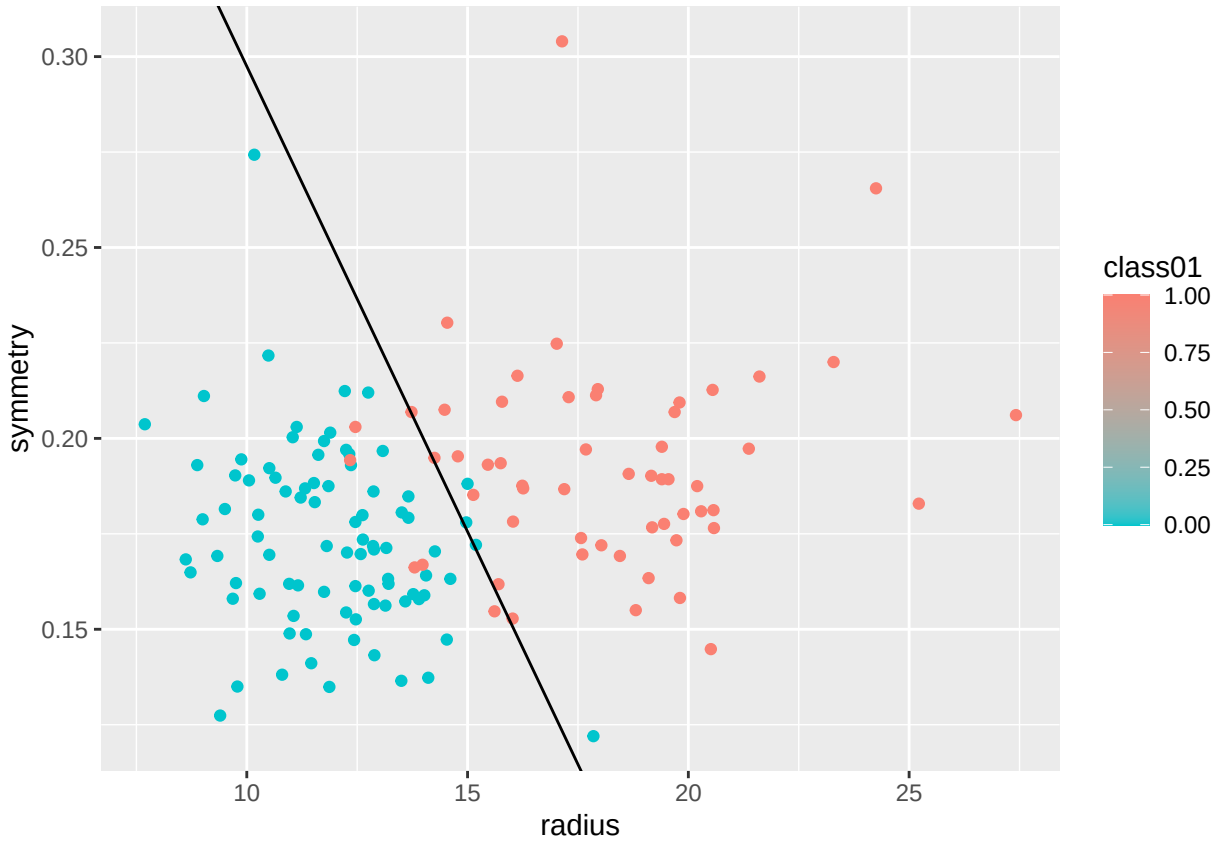
This gives the log-odds regression equation as

$$\ln \frac{p(X)}{1-p(X)} = -23.611 + 1.063 \cdot \text{radius} + 43.623 \cdot \text{symmetry}$$

To get the decision boundary for logistic regression, note that the two classes $Y = 0$ and $Y = 1$ are equally likely when $\ln \frac{p(X)}{1-p(X)} = 0$. Solving for the symmetry as a function of radius:

$$\text{symmetry} = \frac{23.611}{43.623} - \frac{1.063}{43.623}\text{radius}$$

```
wisc_test %>% ggplot(aes( x = radius, y  = symmetry, color = class01 ))+
  geom_point() + scale_colour_gradient(low = "turquoise3", high = "salmon")+
  geom_abline(slope = -br/bs, intercept = -b0/bs)
```



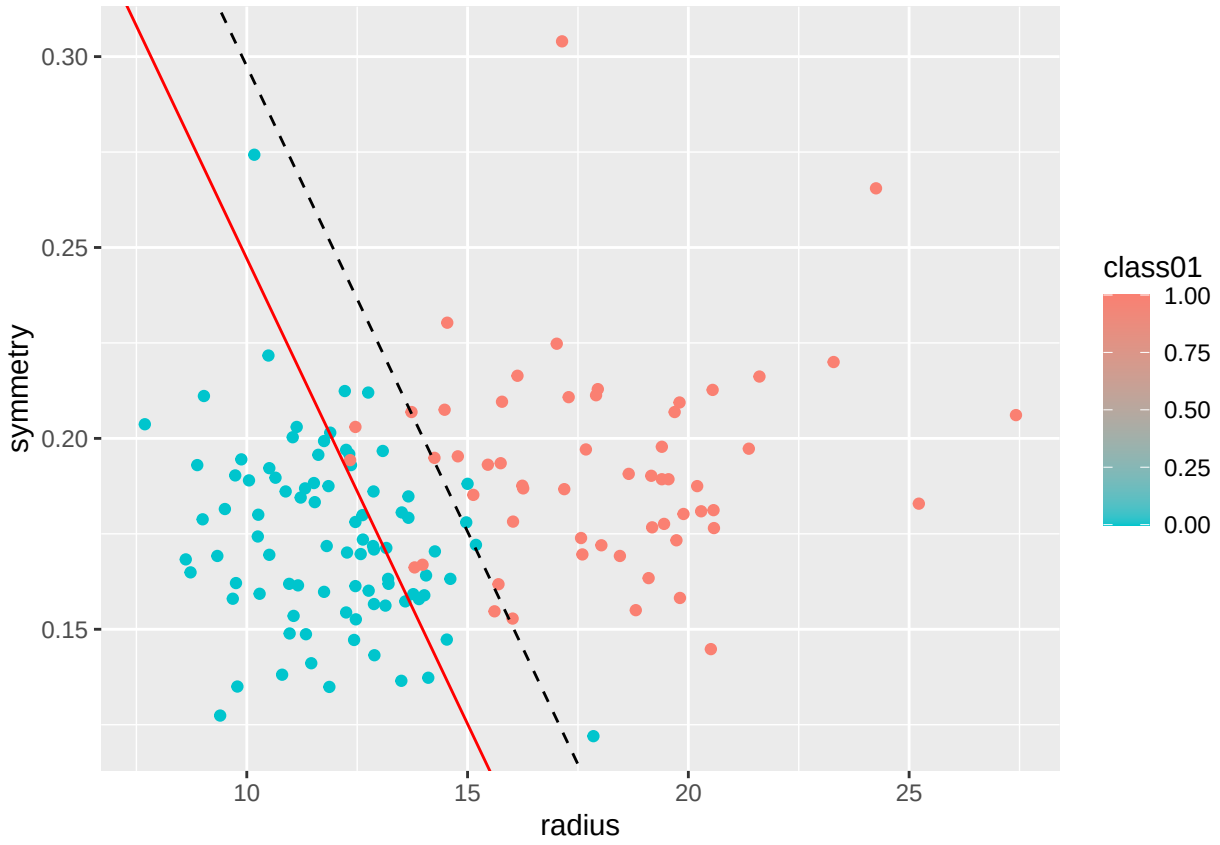
b. The logistic regression model performed relatively well, although it appears to have a higher rate of false negatives (type II error; malignant tumors diagnosed as benign) than false positives (type I; benign tumors diagnosed as malignant) error.

c. With $p(X) = 0.1$ as the threshold, the log odds are $\ln 0.1/0.9 = -2.198$. Solving for symmetry in terms of radius gives the equation

$$\text{symmetry} = \frac{23.611}{43.623} - \frac{1.063}{43.623}\text{radius}$$

In the plot below, the red line is the new classification boundary, and the dashed black line is the old classification boundary

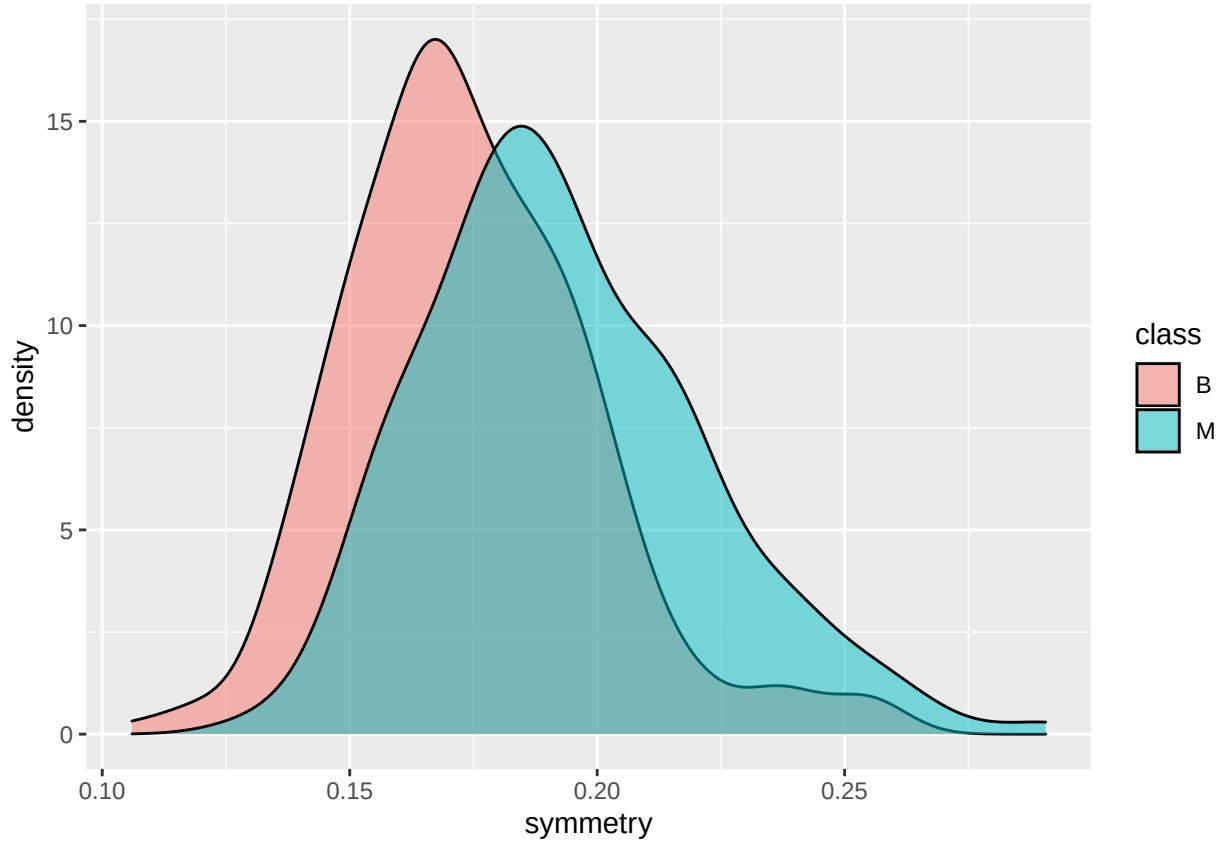
```
wisc_test %>% ggplot(aes( x = radius, y  = symmetry, color = class01))+
  geom_point() +
  geom_abline(slope = -br/bs, intercept = ( log(1/9) - b0 )/bs, color = "red") +
  geom_abline(slope = -br/bs, intercept = -b0/bs, color = "black", linetype = "dashed")+
  scale_colour_gradient(low = "turquoise3", high = "salmon")
```



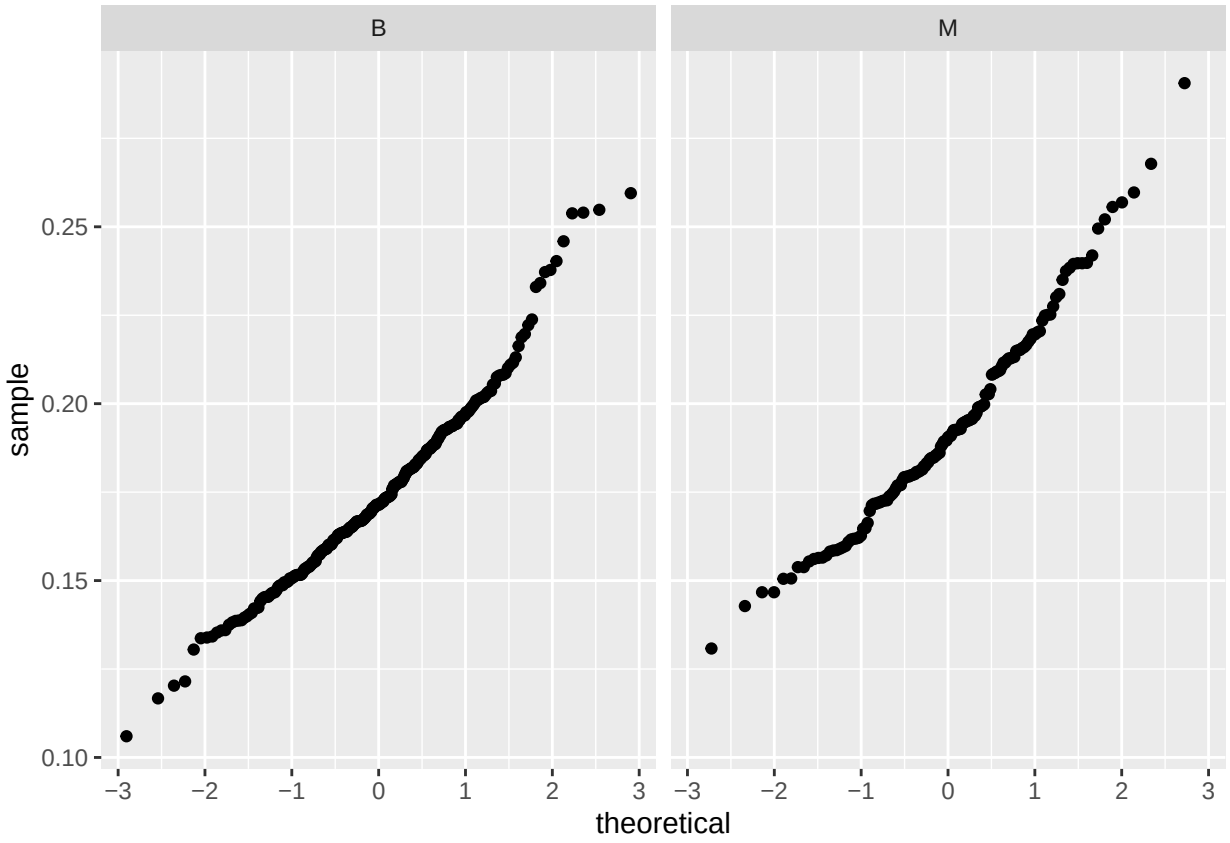
Changing the threshold to 0.1 appears to have shifted the decision boundary to the left of the plot, and increasing the error rate overall. However, it has decreased the type II error to almost zero, at the cost of much greater type I error rate. In a medical setting where Type II error is more catastrophic than Type I, this may be the appropriate choice (even if it means more errors overall).

d.

```
ggplot(wisc_train , aes( x = symmetry,  fill = class) )+geom_density( alpha = .5 )
```




```
ggplot(wisc_train)+geom_qq(aes(sample = symmetry))+facet_wrap(~class)
```




```
wisc_train %>% group_by(class) %>%
  summarize(mean = mean(symmetry), var = var(symmetry), n = n())
```

```
## # A tibble: 2 x 4
##   class  mean      var     n
##   <chr> <dbl>    <dbl> <int>
## 1 B     0.174 0.000633   271
## 2 M     0.193 0.000786   155
```

Based on density plots and qq-plots, symmetry does appear to be conditionally Normally distributed. Moreover, both conditional distributions seem to have approximately the same variance. It would appear LDA modeling is appropriate.

e.

```
symmetry_stats<-wisc_train %>% group_by(class) %>%
  summarize(mean = mean(symmetry), var = var(symmetry), n = n()) %>%
  mutate(prior = n/sum(n), ssx = var*(n-1) )

symmetry_stats
```

```
## # A tibble: 2 x 6
##   class  mean      var     n prior   ssx
##   <chr> <dbl>    <dbl> <int> <dbl> <dbl>
## 1 B     0.174 0.000633   271 0.636 0.171
## 2 M     0.193 0.000786   155 0.364 0.121
```

f. The discriminant equations are given by

$$\delta_j(x) = x\frac{\mu_j}{\sigma^2} - \frac{\mu_j^2}{2\sigma^2} + \ln \pi_i$$

```
delta_f <- function(x, mu, sigma2, pi){
  x*(mu/sigma2) - (mu^2 / (2*sigma2)) + log(pi)
}
```

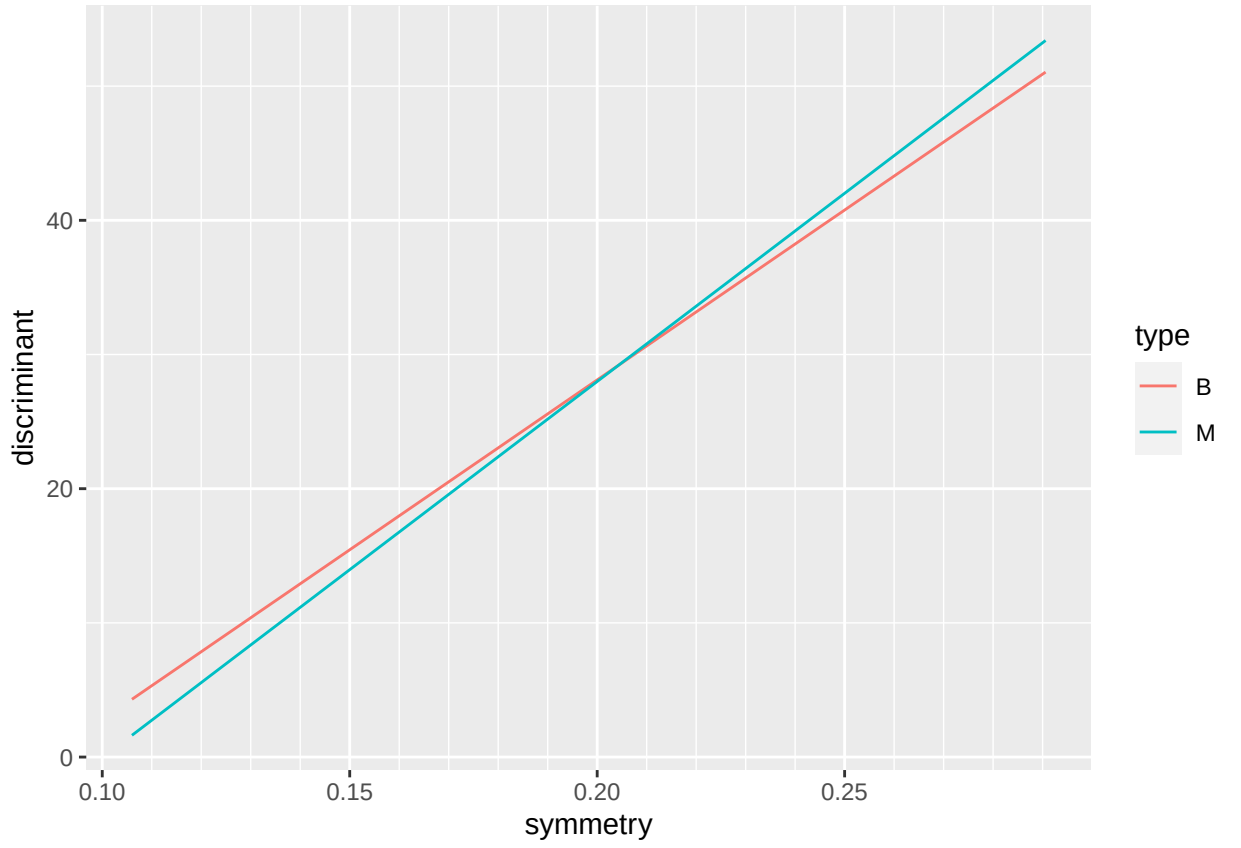
```
mu_b<-symmetry_stats$mean[1]
mu_m<-symmetry_stats$mean[2]
pi_b<-symmetry_stats$prior[1]
pi_m<-symmetry_stats$prior[2]

sigma2<-symmetry_stats %>% summarize(S = sum(ssx), N = sum(n)) %>%
  mutate(sigma2 = S/(N-2) ) %>% pull(3)

d_b<-delta_f(wisc_train$symmetry, mu_b, sigma2, pi_b)
d_m<-delta_f(wisc_train$symmetry, mu_m, sigma2, pi_m)

d<-data.frame(X = rep(wisc_train$symmetry, 2),
              discriminant = c(d_b, d_m),
              type = rep(c("B", "M"), each = length(wisc_train$symmetry)))

ggplot(d) + geom_line(aes(x = X, y = discriminant,color = type )) +
  labs(x = "symmetry", y = "discriminant") +
  scale_x_continuous(minor_breaks = seq( from = .1, to = .3, by = .01))
```



Based on the graph, the decision boundary occurs at about symmetry = 0.2.

Setting the two discriminant equations equal to each other:

$$c\frac{\mu_m}{\sigma^2} - \frac{\mu_m^2}{2\sigma^2} + \ln \pi_m = c\frac{\mu_b}{\sigma^2} - \frac{\mu_b^2}{2\sigma^2} + \ln \pi_b$$

And solving for c:

$$c = \frac{\mu_b + \mu_m}{2} + \frac{\sigma^2(\ln \pi_m - \ln \pi_b)}{\mu_b - \mu_m}$$

We get $c = .204$

```
c<- (mu_b + mu_m)/2 + (sigma2*(log(pi_m) - log(pi_b)))/(mu_b - mu_m)
c
```

```
## [1] 0.2042905
```

g. The error rate on **training** data is approximately .312.

```
wisc_preds<- as.factor( ifelse(wisc_train$symmetry > c, "1", "0"))

wisc_results <- data.frame(obs = as.factor(wisc_train$class01), preds = wisc_preds)

conf_mat(wisc_results, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction   0   1
##          0 245 107
##          1  26  48
```

```
accuracy(wisc_results, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.688
```

```
err <- 1 - pull(accuracy(wisc_results, truth = obs, estimate = preds), .estimate)
err
```

```
## [1] 0.3122066
```

h. **This part was optional.**

```
wisc_cov<-wisc_train %>% group_by(class) %>%
  summarize(var_r = var(radius),
            var_s = var(symmetry),
            cov_rs = cov(radius, symmetry),
            N = n(), mean_r = mean(radius),
            mean_s = mean(symmetry)) %>% data.frame()
wisc_cov
```

```
##   class     var_r        var_s      cov_rs   N   mean_r    mean_s
## 1     B  3.135478 0.0006334322 -0.01012673 271 12.22872 0.1744207
## 2     M 10.518484 0.0007860744 -0.01150734 155 17.24090 0.1932239
```

```
mu_b2<-c(wisc_cov[1,6], wisc_cov[1,7])
mu_m2<-c(wisc_cov[2,6], wisc_cov[2,7])

cov_b<-matrix( c(wisc_cov[1,2], wisc_cov[1,4],
                 wisc_cov[1,4], wisc_cov[1,3]),
               nrow = 2,
               byrow = T)

cov_m<-matrix( c(wisc_cov[2,2], wisc_cov[2,4],
                 wisc_cov[2,4], wisc_cov[2,3]),
               nrow = 2,
               byrow = T)

cov_pool<-(cov_b*wisc_cov[1,5] + cov_m*wisc_cov[2,5])/( sum(wisc_cov[,5]) - 2 )

cov_pool
```

```
##            [,1]          [,2]
## [1,]  5.8492438 -0.0106791999
## [2,] -0.0106792  0.0006922209
```

```
cov_pool_inv<-solve(cov_pool)
```

$$x^T\Sigma^{-1}\mu_m - \frac{1}{2}\mu_m^T\Sigma^{-1}\mu_m + \log\pi_m = x^T\Sigma^{-1}\mu_b - \frac{1}{2}\mu_b^T\Sigma^{-1}\mu_b + \log\pi_b$$

$$x^T\Sigma^{-1}(\mu_m - \mu_b) = \frac{1}{2}\mu_m^T\Sigma^{-1}\mu_m - \frac{1}{2}\mu_b^T\Sigma^{-1}\mu_b + \log\pi_b - \log\pi_m$$

```
RHS<-.5*t(mu_m2)%*%cov_pool_inv%*%mu_m2
-.5*t(mu_b2)%*%cov_pool_inv%*%mu_b2
```

```
##           [,1]
## [1,] -41.55368
```

```
+ log(pi_b)- log(pi_m)
```

```
## [1] 0.5586937
```

```
RHS
```

```
##          [,1]
## [1,] 62.93622
```

```
LHS <- cov_pool_inv%*%(mu_m2 - mu_b2)
LHS
```

```
##            [,1]
## [1,]  0.9327599
## [2,] 41.5536967
```

```
slope<- -LHS[1,1]/LHS[2,1]
slope
```

```
## [1] -0.0224471
```

```
intercept = RHS/LHS[2,1]
intercept
```

```
##           [,1]
## [1,] 1.514576
```

Equation:

$$\text{symmetry} = 0.514576 - 0.0224 \cdot \text{radius}$$

```
## Red = LDA
## Black = Log Reg

wisc_test %>% ggplot(aes( x = radius, y  = symmetry, color = class01))+
  geom_point() +
  geom_abline(slope = slope, intercept = intercept-1 , color = "red" ) +
  geom_abline(slope = -br/bs, intercept = -b0/bs, color = "black")
```