STAT 435 Homework 2

Jaiden Atterbury

2024-04-19

Note
In this homework, I briefly collaborated with Aarav Vishesh Dewangan. Furthermore, throughout all of the problems where randomness is involved, I used `set.seed(224)`.

Exercise 1
In this exercise, we will simulate data with a qualitative response, $y = \{0, 1\}$, and perform k-nearest neighbors. We will take $p = 2$ and $n = 200$. Thus, we have 200 data points and 2 different independent variables $\{(x_{i1}, x_{i2})\}$.

Part a
In this part of the exercise, we will first generate data with a non-linear Bayes decision boundary. In particular, we will start by sampling a vector x_1 of length 200 from a $N(1, 2^2)$ distribution. We will then sample a vector x_2 of length 200 from a $Uniform(-11, 11)$ distribution. After this we will generate a vector ϵ of random noise from a $N(0, 1)$ distribution, this will ensure that our decisions are imperfect.

Now that our sample vectors are computed, we will compute the decision boundary as

$$f(x_1, x_2) = x_2 + \frac{1}{3}(x_1 - 1)^2 - \frac{1}{2} + \epsilon. \tag{1}$$

Furthermore we will use the following classification criteria to label our data points

$$y_i = \begin{cases} 1, & f(x_{i1}, x_{i2}) > 1 \\ 0, & \text{else} \end{cases} \tag{2}$$

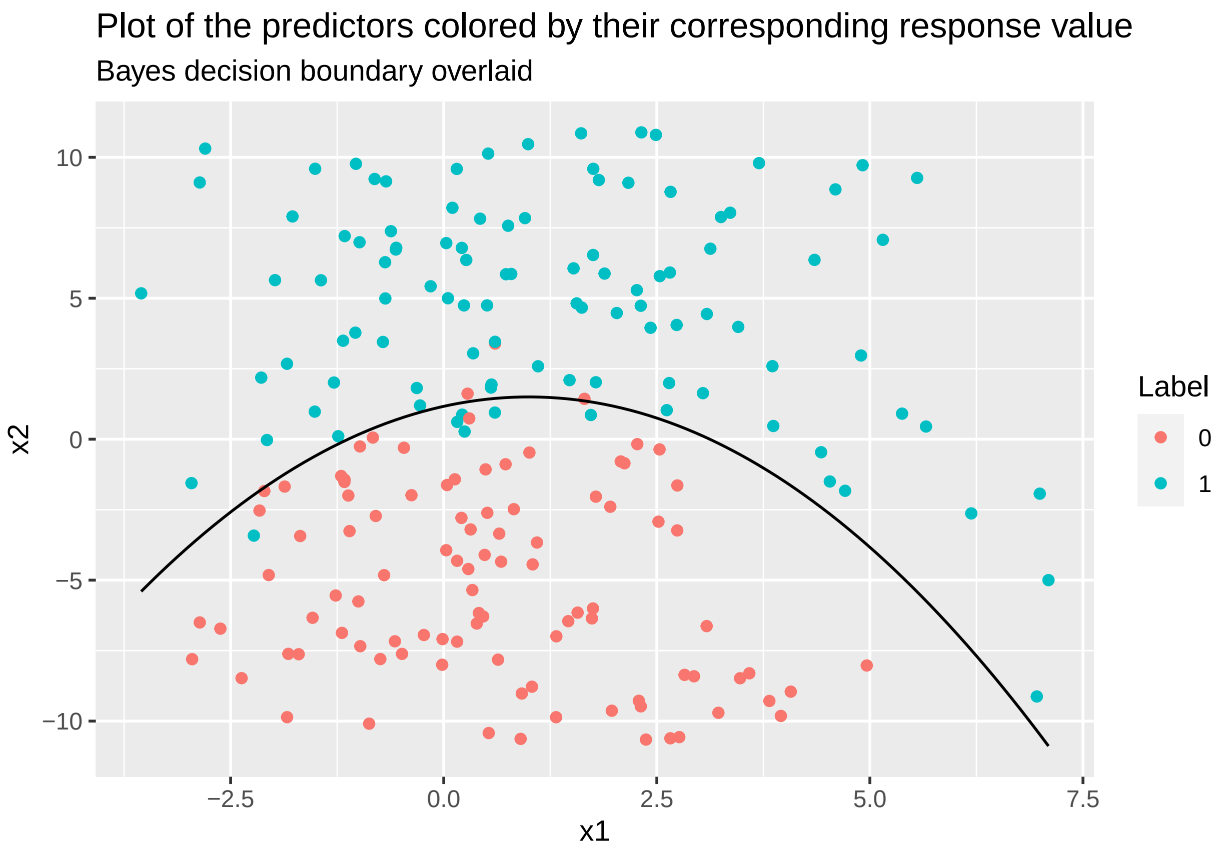
Now that we have computed the decision boundary and labeled our sample data points, we will now ensure that we have an approximately equal number of points for $y = 0$ and $y = 1$. Based on the given sample and decision boundary, we obtained 103 points labeled $y = 1$, and 97 points labeled $y = 0$. Based on the fact that a (80,120) split is passable, we have an approximately equal split.

If we wanted to generate data with a linear decision boundary we could modify Equation (1) in many ways. One way we could modify Equation 1 is by simply removing the square from the second term, resulting in the following linear equation: $f(x_1, x_2) = x_2 + \frac{1}{3}(x_1 - 1) - \frac{1}{2} + \epsilon$. Another way we could modify Equation 1 is by simply dropping the squared term from the equation, resulting in the following linear equation that only depends on x_2: $f(x_1, x_2) = x_2 - \frac{1}{2} + \epsilon$. In general, as long as we drop the quadratic terms from Equation 1, the resulting equation will be a linear decision boundary. Now, if we wanted to modify Equation 2 to allow for more than two classes, all we would need to do is add more conditional statements to the equation. One example of modifying Equation 2 to allow for more than 2 classes is as follows

$$y_i = \begin{cases} 2, & f(x_{i1}, x_{i2}) > 2 \\ 1, & 2 \geq f(x_{i1}, x_{i2}) > 1 \\ 0, & \text{else} \end{cases}$$

The above equation allows for the classification of three classes, however, there are infinitely many ways we could've chosen these three classes, and the same logic extends up to k classes. In this specific example, we kept the first case as is, and then we modified Equation 2 by simply adding to the case when $f(x_{i1}, x_{i2}) > 1$. Namely, we added an intermediate condition, which checks if $2 \geq f(x_{i1}, x_{i2}) > 1$, as well as a condition which checks if $f(x_{i1}, x_{i2}) > 2$.

Part b
In this part of the exercise, we will plot the data points $\{(x_{i1}, x_{i2})\}$. In particular, we will color code each point based on the response y_i, as well as plot the decision boundary. All of this will be presented in a single plot. This is done in R below using `ggplot`.



As expected, without the noise term, our classifier is slightly off, but for the most part, the points fall on the correct side of the decision boundary.

Part c
In this part of the exercise, we will write a function `knn(x_train, y_train, x_new, k)` that takes in training data, labels for the training data, a new data point, and the number of neighbors k. This function will perform k-nearest neighbors and return the label for the new data point. Since `knn` is already a function in R, I will call this function `knn_2`. The R code for this function is below.

```r
knn_2 <- function(x_train, y_train, x_new, k) {
  # Create an empty vector to store the distances of x_new with each point
  dists <- rep(NA, times=nrow(x_train))

  # Loop over the number of rows/data points in the dataframe/matrix
  for (n in 1:nrow(x_train)) {

    # Initialize the total distance
    dist = 0
    # Loop over the number of columns/predictors in the dataframe/matrix
    for (p in 1:ncol(x_train)) {

      # Compute the part of the Euclidean distance under the square root
      dist = dist + (x_new[p]-x_train[n,p])^2
    }

    # Add the total Euclidean distance to the dists vector
    dists[n] = sqrt(dist)
  }
  # If x_train was a dataframe, we need to unlist the resulting dists list
  dists_new <- unlist(dists)

  # Use the order function to obtain the indices of the nearest neighbors
  indices <- order(dists_new)[1:k]

  # Obtain the labels from the nearest neighbors
  labels <- y_train[indices]

  # Obtain the most frequent label
  prediction <- names(sort(table(labels), decreasing=TRUE)[1])

  # Return this label as the prediction
  return(prediction)
}
```

This function assumes that `x_train` is either a matrix or a dataframe, `y_train` and `x_new` are vectors, and `k` is a positive integer. Furthermore, since there were no instructions on how to settle tiebreakers, this function will break ties in whatever fashion R uses to settle tiebreakers. In particular, if there is a tie between the closest neighbors, however the `order` function breaks ties will be used. Furthermore, if there is a tie between the chosen label, however the `sort` function breaks ties will be used.

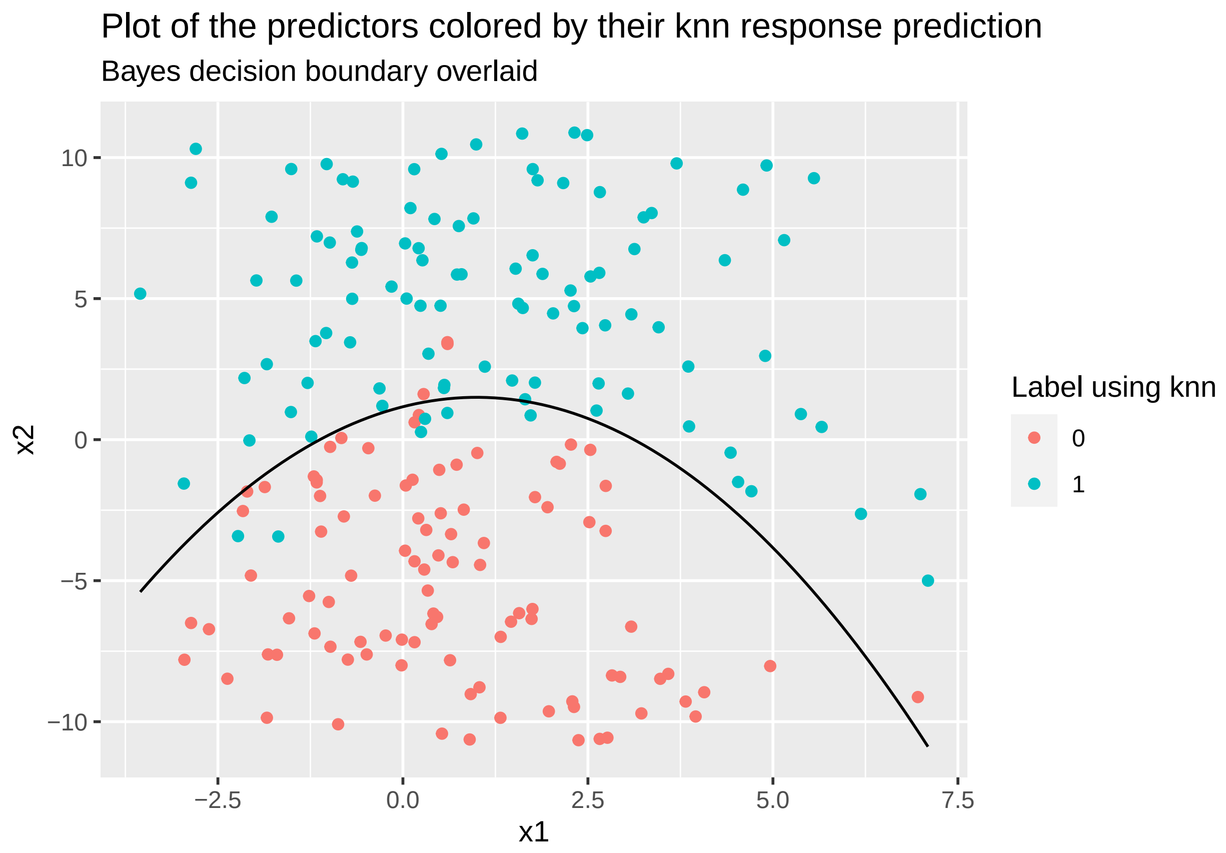
Part d

In this part of the exercise, we will perform k-nearest neighbors on the 200 training data points for $k = 1, 2, 3, \dots, 50$, using the `knn` function from the `class` package. For each k we will compute the training error. We will also determine for which k the lowest training error occurs. Furthermore, we will present a single plot where all the training data points are predicted using this k with the true Bayes decision boundary overlaid. Lastly, we will present a 2×2 confusion matrix. All of this is done below.

It is important to note that since we are using our training data to test the accuracy, we will need to use $k + 1$ in the `knn` function. If we don't do thus, when $k = 1$, for each point in our "testing set" the nearest neighbor will always be the given data point, leading to a training accuracy of 100% (that is, a training error rate of 0%.) With that being said, from performing k-nearest neighbors on the 200 training data points for $k = 1, 2, 3, \dots, 50$, using the `knn` function from the `class` package, the training errors for each k were as follows

```
##  [1] 0.030 0.040 0.035 0.040 0.045 0.050 0.050 0.045 0.050 0.050 0.055 0.055
## [13] 0.045 0.060 0.060 0.055 0.055 0.045 0.050 0.060 0.050 0.065 0.065 0.070
## [25] 0.070 0.075 0.075 0.070 0.070 0.075 0.075 0.075 0.075 0.075 0.075 0.080
## [37] 0.070 0.080 0.080 0.080 0.075 0.075 0.075 0.075 0.075 0.075 0.075 0.075
## [49] 0.080 0.080
```

As can be seen from above, the value of k that resulted in the smallest training error was 1, which resulted in a training error of 0.03. We will now present a single plot where all the training data points are predicted using this k, as well as the true decision boundary.



As can be seen from the above plot, the predictions using 1 as the value of k looks very similar to the plot in part (b). This makes sense intuitively because this value of k was selected with the smallest training error. We will now present the 2×2 confusion matrix to further emphasize these results. It is important to note that, since R uses randomness to break the tiebreakers in the `knn` function, the error from this trial might not be the same as the error found when computing the training error for all values of k in the range $[1, 50]$. This is because we recomputed the k-nearest neighbors to obtain the predictions once we found the "best" value of k.

```
##          yi
## knn_k_low  0  1
##         0 94  4
##         1  3 99
```

As can be seen by the above confusion matrix, there were 4 times when the true class was actually 1, but our model instead said it was in class 0. Furthermore, there were 3 times when the true class was actually 0, but our model instead said it was in class 1. Therefore, out of 200 data points, we had 7 errors, for an error rate of $7/200 = 0.035$ (which, as expected, differs slightly from our original error rate).

Part e
In this part of the exercise, using the same data generating process from part (a), we will generate 5000 new data points, which will constitute as our test data set. Again, we will perform k-nearest neighbors on these data points for $k = 1, 2, 3, \ldots, 50$ and compute the test error for each k. At the end we will report the smallest test error and the corresponding k. This is done below.

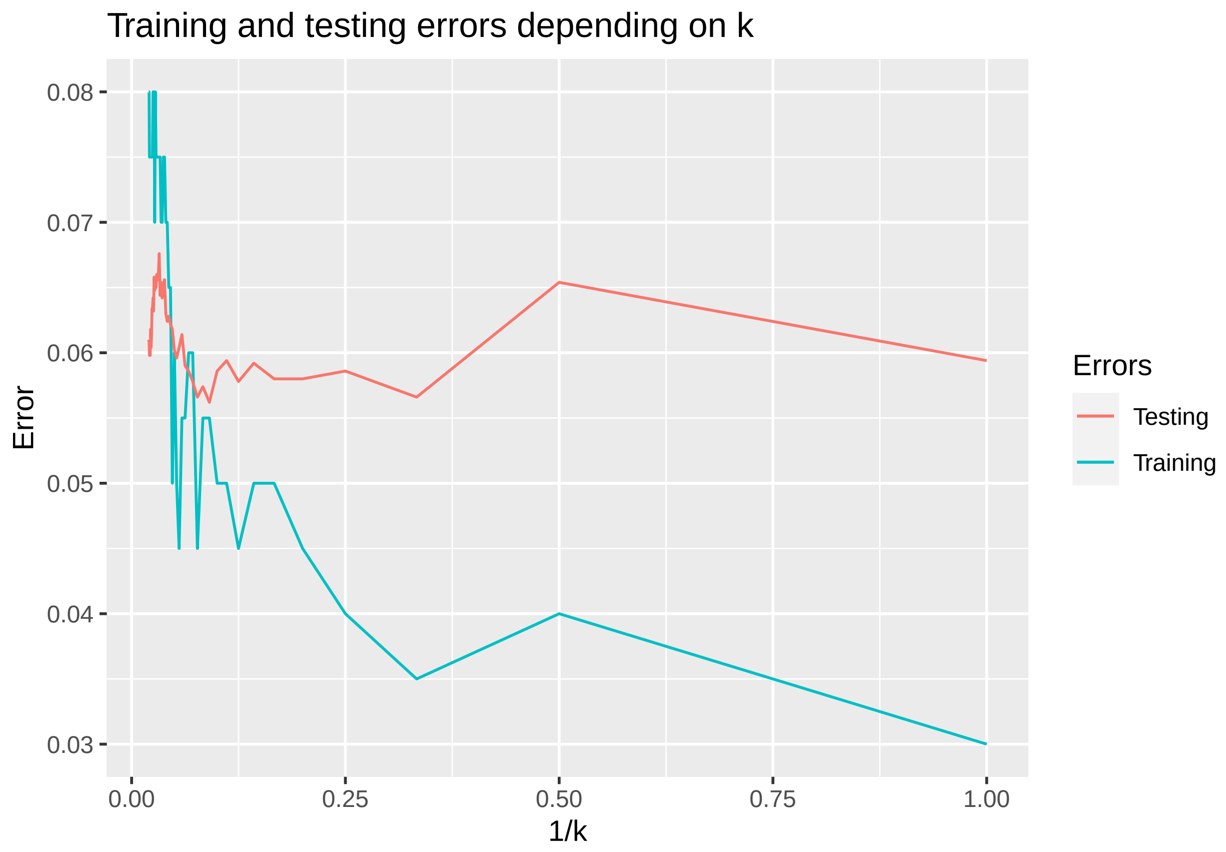
Before moving on, we will ensure that we have approximately equal number of points for $y = 0$ and $y = 1$. Based on the given sample and decision boundary, we obtained 2396 points labeled $y = 1$, and 2604 points labeled $y = 0$. Thus, as seen above, we have an approximately equal split.

From performing k-nearest neighbors on the 5000 training data points for $k = 1, 2, 3, \ldots, 50$, using the `knn` function from the `class` package, the testing errors for each k were as follows

```
##  [1] 0.0594 0.0654 0.0566 0.0586 0.0580 0.0580 0.0592 0.0578 0.0594 0.0586
## [11] 0.0562 0.0574 0.0566 0.0578 0.0586 0.0590 0.0614 0.0604 0.0596 0.0602
## [21] 0.0618 0.0622 0.0628 0.0624 0.0630 0.0656 0.0652 0.0642 0.0654 0.0644
## [31] 0.0676 0.0660 0.0656 0.0660 0.0650 0.0650 0.0648 0.0658 0.0632 0.0642
## [41] 0.0634 0.0634 0.0612 0.0604 0.0618 0.0598 0.0600 0.0598 0.0608 0.0610
```

As can be seen from above, the value of k that resulted in the smallest training error was 11, which resulted in a training error of 0.0562.

Part f
In this part of the exercise, we will make a single plot with $1/k$ on the horizontal axis and the error on the vertical axis. In this plot, we will have the training error and testing error be two separate lines in the same plot. For the scale on the axis, we will choose the scale so that the trends can be seen clearly. Furthermore, we will comment on what we see and if anything surprises us. Lastly, based on our results, we will explain what value of k we should choose. All of this is done below.



When looking at the above plot, none of the results are surprising. With that being said, one thing that stands out right away when looking at the plot, is that as $1/k$ gets closer and closer to 1, the training error decreases, while the highest training errors seem to happen as $1/k$ approaches 0. In terms of the testing error, as $1/k$ increases, the testing error decreases rapidly and then stays relatively constant, until increasing again and then staying relatively constant as $1/k$ begins to approach 1. Furthermore, when k is large, the difference between the training and testing error is not that big. Lastly, the training error as a whole is smaller than the testing error for low values of k, which makes sense because overfitting can become a concern for low values of k.

In order to find which value of k to choose, we want to find a value that minimizes both the training error and the testing error. This value of k appears to be where both points hit a local minimum between the $1/k$ values of 0.25 and 0.5. I estimate this $1/k$ value to be around 0.35, which corresponds to a k value of around 3. Therefore, $k = 3$ is the value of k that I would choose.

Exercise 2
In this exercise, we will generate binary data using a simple probabilistic model. After this, we will fit a logistic regression model and attempt to recover the true parameters.

Part a
In this exercise, we will first generate our responses, in which we will use the logit model/logistic distribution to help us. In particular, we will sample a vector x of length $n = 100$ from a $Uniform(-9, 9)$ distribution.

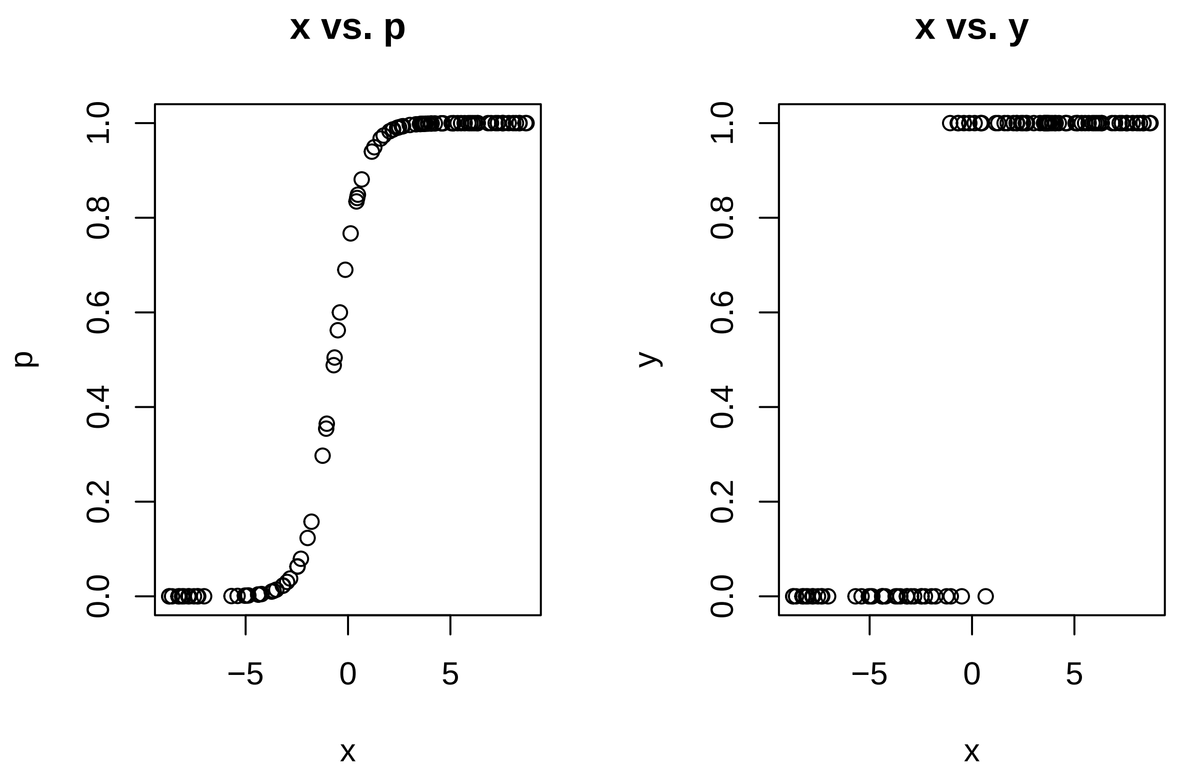
Under the logistic distribution, the probability of success is given as

$$P(Y = 1|X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{3}$$

Taking $[\beta_0, \beta_1] = [1, 1.5]$, and using Equation 3, we will now compute p, the vector of success probabilities.

Furthermore, noting that $y|p_i \sim Bernoulli(p_i)$, we can simulate the outcomes using a Bernoulli trial. We will now generate y_i with the `rbinom(1, size = 1, prob = p i)` function, where p_i is $P(Y = 1|X = x_i)$, for each of the 100 x_i values. This vector of outcomes will be named y

We will now present two plots. On the first, we will plot x vs. p, and on the second, we will plot x vs. y. This is done below.



The x vs. y plot only had values at $y = 0$ and $y = 1$, which make sense because y was simulated from a Bernoulli. On the other hand, the x vs. p plot looks more like a logistic distribution, which makes sense because that's where we simulated the values of p from.

Part b
In this part of the exercise, we will start by fitting a logistic regression model on this data in R using the `glm` function. We will then report the estimated coefficients and standard errors. This is done below.

```
##
```

```
## Call:
## glm(formula = as.factor(y) ~ x, family = binomial(link = "logit"))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.5124     0.7885   1.918  0.05511 .
## x             1.9895     0.7047   2.823  0.00476 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 132.813  on 99  degrees of freedom
## Residual deviance:  16.904  on 98  degrees of freedom
## AIC: 20.904
##
## Number of Fisher Scoring iterations: 9
```

As can be seen from the above output, $\hat{\beta}_0 = 1.5124$, with a standard error of $SE(\hat{\beta}_0) = 0.7885$. Also, we can see that, $\hat{\beta}_1 = 1.9895$, with a standard error of $SE(\hat{\beta}_1) = 0.7047$.

We will now interpret the $\hat{\beta}_1$ coefficient. For every unit increase in x_1, we estimate that the log odds of $P(Y = 1|X = x)$ will increase by $\hat{\beta}_1 = 1.9895$, on average.

Since the true value of β_0 was 1 and our estimate $\hat{\beta}_0$ was 1.5124, we can see that the estimate was 0.5124 units larger than the true value. Furthermore, since the true value of β_1 was 1.5 and our estimate $\hat{\beta}_1$ was 1.9895, we can see that the estimate was 0.4895 units larger than the true value. Although these values aren't too far off, one would like these estimates to be quite a bit closer. One reason why this may have occurred is because the two groups $y = 1$ and $y = 0$ are quite distinct, and logistic regression parameter estimates in R break down/are unstable when the groups are too distinct. However, since there is still quite a bit of overlap, this may suggest why our estimates are still relatively close to there actual values. Another reason for these differences may be due to the relatively small sample size of 100.

Part c

In this part of the exercise, we will compute the predicted probabilities $\hat{p}$ using the `predict` function with the argument `type = response`.

Using these predicted probabilities, we will declare the predicted outcome as $\hat{y}_i = 1$ if $\hat{p}_i \geq 0.5$ and $\hat{y}_i = 0$ otherwise. With these predicted outcomes in hand, we will generate and present a 2×2 confusion matrix, as well as compute the prediction error using the confusion matrix. This is done below.

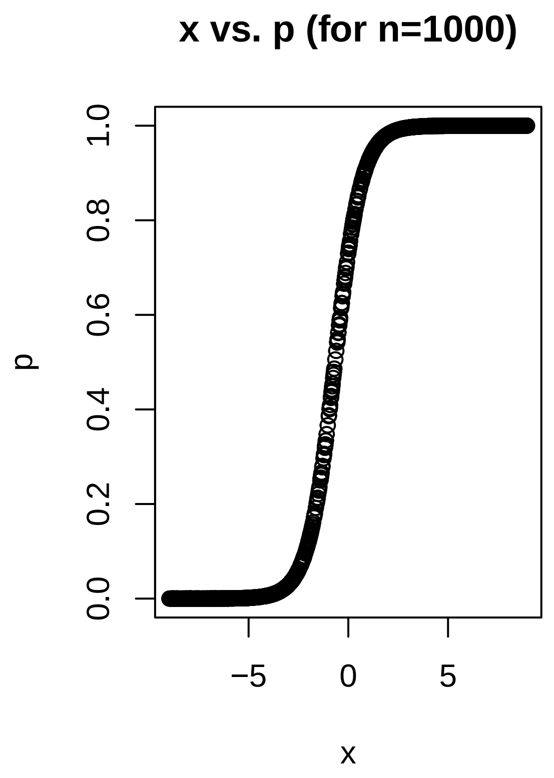
```
##     y
## yhat  0  1
##    0 36  1
##    1  2 61
```

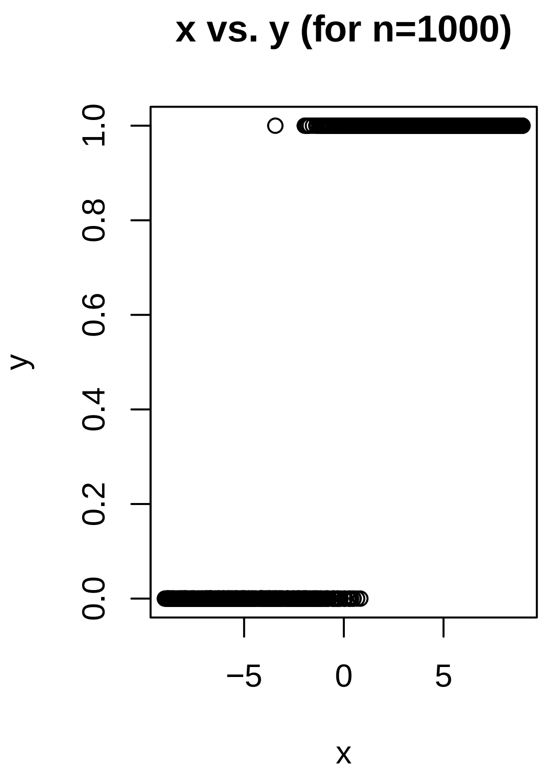
As can be seen by the above confusion matrix, there was 1 time when the true class was actually 1, but our model instead said it was in class 0. Furthermore, there were 2 times when the true class was actually 0, but our model instead said it was in class 1. Therefore, out of 100 data points, we had 3 errors, for an error rate of $3/100 = 0.03$.

Part d

In this part of the exercise, we will repeat parts (a) and (b) with $n = 10^3$ and $n = 10^4$. In each case we will compare the estimated coefficients for the different values of n against the true parameters and comment on what we see.

Starting with $n = 10^3$, we will generate n observations from a $Uniform(-9, 9)$ distribution, compute p_i for each x_i using the logistic distribution, and then compute y_i for each x_i using the Bernoulli distribution with $p = p_i$. Once this is done we will plot x vs. p and x vs. y. This is done below.





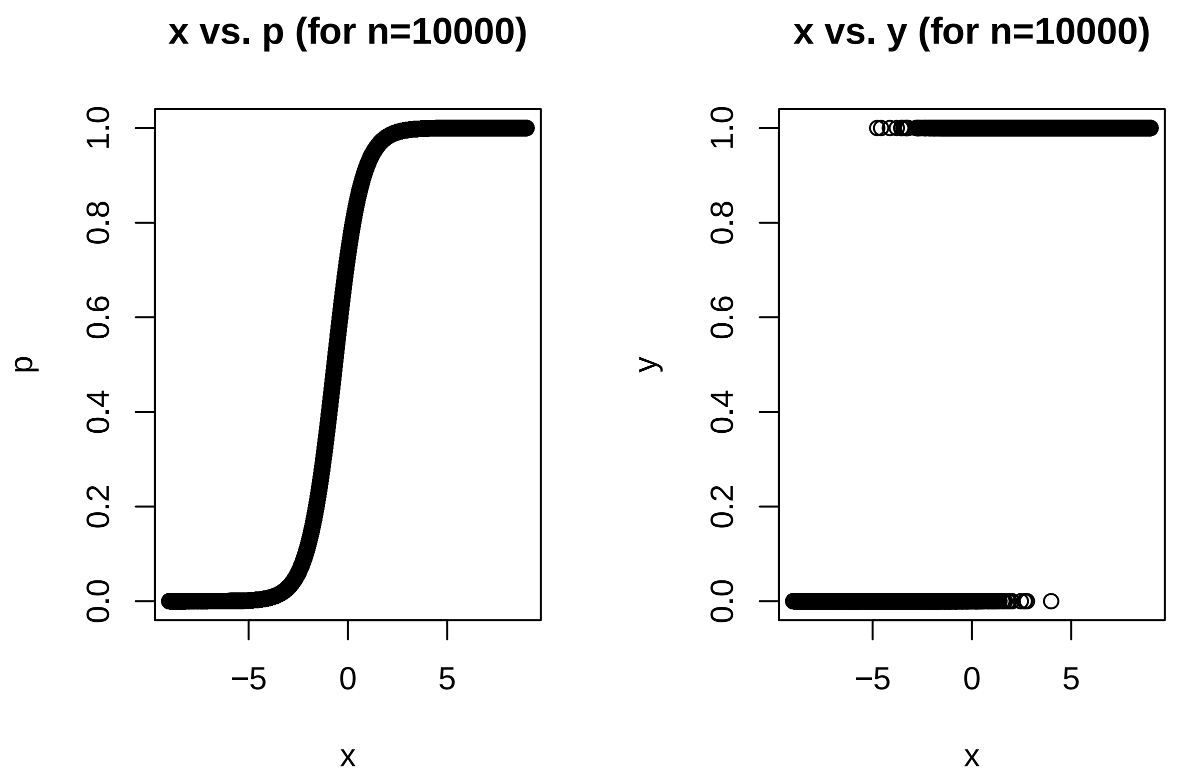
As noted previously, the x vs. y plot only had values at $y = 0$ and $y = 1$, which make sense because y was simulated from a Bernoulli. On the other hand, the x vs. p plot looks more like a logistic distribution, which makes sense because that's where we simulated the values of p from. We will now move on to repeating part (b) for $n = 10^3$. Fitting the logistic regression model and comparing coefficients is done below.

```
##
## Call:
## glm(formula = as.factor(y_103) ~ x_103, family = binomial(link = "logit"))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.9273     0.2039   4.547 5.45e-06 ***
## x_103         1.6830     0.1648  10.213  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.10  on 999  degrees of freedom
## Residual deviance:  203.31  on 998  degrees of freedom
## AIC: 207.31
##
## Number of Fisher Scoring iterations: 9
```

As can be seen from the above output, $\hat{\beta}_0 = 0.9273$, with a standard error of $SE(\hat{\beta}_0) = 0.2039$. Also, we can see that, $\hat{\beta}_1 = 1.6830$, with a standard error of $SE(\hat{\beta}_1) = 0.1648$.

Since the true value of β_0 was 1 and our estimate $\hat{\beta}_0$ was 0.9273, we can see that the estimate was 0.0727 units smaller than the true value. Furthermore, since the true value of β_1 was 1.5 and our estimate $\hat{\beta}_1$ was 1.6830, we can see that the estimate was 0.183 units larger than the true value. In terms of absolute distance from the true values of the parameters, when $n = 1000$, the parameters are much closer to their true values as compared to $n = 100$, but still not as close as we would want.

Moving onto $n = 10^4$, we will generate n observations from a $Uniform(-9, 9)$ distribution, compute p_i for each x_i using the logistic distribution, and then compute y_i for each x_i using the Bernoulli distribution with $p = p_i$. Once this is done we will plot x vs. p and x vs. y. This is done below.



Again, as noted previously, the x vs. y plot only had values at $y = 0$ and $y = 1$, which make sense because y was simulated from a Bernoulli. On the other hand, the x vs. p plot looks more like a logistic distribution, which makes sense because that's where we simulated the values of p from. We will now move on to repeating part (b) for $n = 10^4$. Fitting the logistic model and comparing coefficients is done below.

```
##
## Call:
## glm(formula = as.factor(y_104) ~ x_104, family = binomial(link = "logit"))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.01699    0.05940   17.12   <2e-16 ***
## x_104        1.51602    0.04375   34.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13774.5  on 9999  degrees of freedom
## Residual deviance:  2394.8  on 9998  degrees of freedom
## AIC: 2398.8
##
## Number of Fisher Scoring iterations: 8
```

As can be seen from the above output, $\hat{\beta}_0 = 1.01699$, with a standard error of $SE(\hat{\beta}_0) = 0.05940$. Also, we can see that, $\hat{\beta}_1 = 1.51602$, with a standard error of $SE(\hat{\beta}_1) = 0.04375$.

Since the true value of β_0 was 1 and our estimate $\hat{\beta}_0$ was 1.01699, we can see that the estimate was 0.01699 units larger than the true value. Furthermore, since the true value of β_1 was 1.5 and our estimate $\hat{\beta}_1$ was 1.51602, we can see that the estimate was 0.01602 units larger than the true value. In terms of absolute distance from the true values of the parameters, when $n = 10000$, the parameters are much closer to their true values as compared to $n = 100$, and even $n = 1000$. It seems like, in this case, the sample size n does have quite a big role in determining how close the parameters are to their true values.

Part e
In this part of the exercise, we will go back to the $n = 100$ case and attempt to fit a linear model instead. In particular, we will fit a linear model of the form $y = a + bx$.

We will now compute the predicted probabilities $\hat{p}$ using the `predict` function in R. We will call these predicted values ℓ.

Using these predicted probabilities, we will declare the predicted outcome as $\hat{y}_i = 1$ if $\ell_i \geq 0.5$ and $\hat{y}_i = 0$ otherwise. With these predicted outcomes in hand, we will generate and present a 2×2 confusion matrix, as well as compute the prediction error using the confusion matrix. This is done below.

```
##     y
## lhat  0  1
##    0 36  1
##    1  2 61
```

As can be seen by the above confusion matrix, there was 1 time when the true class was actually 1, but our model instead said it was in class 0. Furthermore, there were 2 times when the true class was actually 0, but our model instead said it was in class 1. Therefore, out of 100 data points, we had 3 errors, for an error rate of $3/100 = 0.03$. This is actually exactly the same confusion matrix that we obtained in part (c) for the logistic regression model.

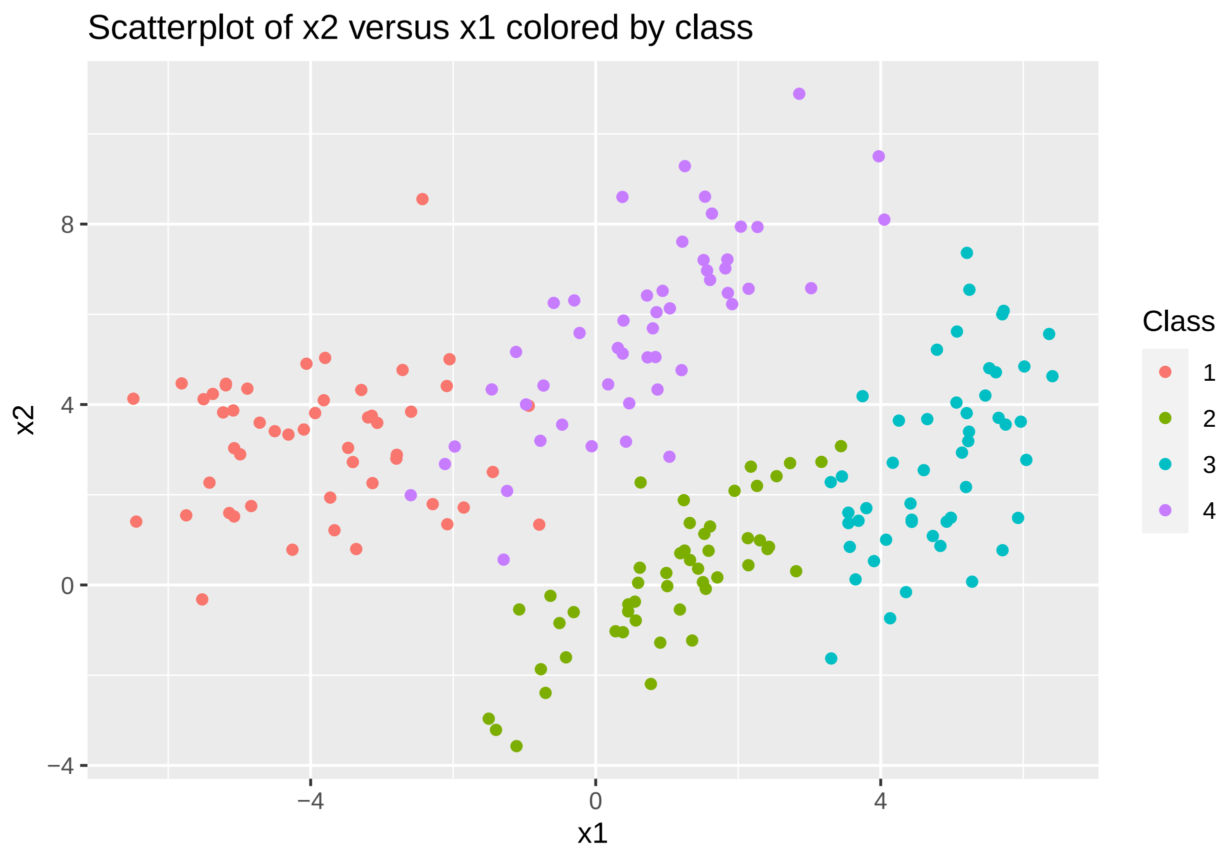
One reason why we have seen the exact same results in both cases, as hinted at in the given hint, is because we drew from a $Uniform(-a, a)$, $a > 0$ distribution. This means that we expect approximately half of the values to be positive, and approximately half of the values to be negative. However, as hinted at in the hint, if we were to draw samples from a Uniform distribution that was skewed on one side, we would expect that the number of negative and positive values are not approximately equal to each other, thus skewing the models and leading to different results. Thus, in summary, as talked about in office hours, if the data are not symmetric around zero, the linear regression "classifier" will predict one of the classes more often than the other, and thus lead to different predictions than an actual classifier handled to output values between 0 and 1. However, since our data was symmetric around 0, the line from the linear regression output followed closely to the center of the logistic distribution curve, thus leading to similar predictions.

Exercise 3
In this exercise, we will simulate new data and explore linear and quadratic discriminant analysis techniques. In particular, we will use $p = 2$ features, $K = 4$ classes, and $n = 50 \times K = 200$ total data points (50 per class).

Part a
In this part of the exercise, we will fix the mean vectors as $\mu_1 = [-4, 3]$, $\mu_2 = [1, 0]$, $\mu_3 = [5, 3]$, and $\mu_1 = [1, 6]$. Similarly, we will fix the covariance matrices as

$$\Sigma_1 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$
$$\Sigma_2 = \Sigma_4 = \begin{bmatrix} 2 & 2 \\ 2 & 3 \end{bmatrix}$$
$$\Sigma_3 = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}$$

Using these fixed values, we will generate 50 data points for each class $k = 1, 2, 3, 4$ that follow a $N(\mu_k, \Sigma_k)$ distribution using the `mvnorm` function from the `MASS` package. After this done we will generate a single plot that shows a scatter plot of all 200 data points color coded by class. This is done below.



As can be seen above, all four of these classes are quite distinct. In this case a multinomial regression model might not work due to the instability of the parameter estimates that this clear class distinction will cause (hence why we are doing LDA and QDA.)

Part b
In this part of the exercise, we will fit a linear discriminant analysis model on the simulated data. Once this is done we will report the training error and 4×4 confusion matrix. This is done below.

As computed after fitting an LDA model on the data, the training error of the LDA model was calculated as 0.055. Below we will now display the corresponding 4×4 confusion matrix.

```
##          true_label
## lda_label  1  2  3  4
##         1 46  0  0  3
##         2  1 48  0  2
##         3  0  2 50  0
##         4  3  0  0 45
```

As can be seen above, there were a total of 11 errors (the sum of the off diagonal entries), for an error rate of, as expected, $11/200 = 0.055$.

Part c
In this part of the exercise, we will fit a quadratic discriminant analysis model on the simulated data. Once this is done we will report the training error and 4×4 confusion matrix. This is done below.

As computed after fitting an LDA model on the data, the training error of the LDA model was calculated as 0.04. Below we will now display the corresponding 4×4 confusion matrix.

```
##          true_label
## qda_label  1  2  3  4
##         1 47  0  0  3
##         2  0 49  1  0
##         3  0  0 49  0
##         4  3  1  0 47
```

As can be seen above, there were a total of 8 errors (the sum of the off diagonal entries), for an error rate of, as expected, $8/200 = 0.04$.

Part d
In this part of the exercise, we will generate $n = 500$ test observations for each class using the same procedure in part (a). We will then use this data to calculate the testing errors of our LDA and QDA models from parts (b) and (c). After this is done we will compare these with the training errors and decide which model we prefer. This is done below.

After running our LDA and QDA models on the testing data set, the LDA model yielded a testing error of 0.0635, and our QDA model yielded a testing error of 0.0565. Since the error in the QDA model was lower for both testing and training, we prefer this model (which makes sense because the covariance matrices are not identical in each class.) However, since the errors in both training and testing were similar in both LDA and QDA, one could make the case that LDA should be chosen since it is the simpler model.

Part e
In this part of the exercise we will speak upon the differences of QDA and LDA in terms of which one has more parameters to predict, which one is simpler, and which one is more generalizable. This discussion is done below.

The key difference between LDA and QDA is that, for $p = 1$, LDA assumes that $\sigma_k = \sigma$. That is, the variance in each class is equal to each other. On the other hand, QDA assumes that $\sigma_k \neq \sigma$. That is, the variance in each class are not all equal to each other. Similarly, for $p > 1$, LDA assumes that $\Sigma_k = \Sigma$. That is, the covariance matrix in each class is equal to each other. On the other hand, QDA assumes that $\Sigma_k \neq \Sigma$. That is, the covariance matrix in each class are not all equal to each other. Since QDA does not assume that all classes have the same variance/covariance matrix, it follows that QDA has more parameters to estimate than LDA. Furthermore, since LDA has less parameters and stronger assumptions about each class it would be the simpler model. On that same token, since QDA does not make the strong assumptions about the variance/covariance of each class like LDA does, it is thus a more generalizable method to a wider amount of data sets (as most of the time assuming that the data in different classes has the same variance is not true/justifiable.)

Part f
In this part of the exercise, based on the true data generation process in part (a) and our answer to part (e), we will conclude if we prefer LDA or QDA. Furthermore, if necessary, we will reconcile any discrepancies with our answer from part (d).

Based on the data generation process in part (a), since the covariance matrices in each class were not the same, QDA is the more appropriate model, and is thus preferred over LDA in this case. Since we came to the same conclusion in part (d) there are no discrepancies to reconcile.

Part g
In this part of the exercise, we will answer the following question: "True/False: Since QDA is a more flexible model than LDA, we always prefer QDA over LDA?" We will answer and justify our selection below.

I believe this statement is **false**. In order to justify this claim, we will look at two cases: theoretical and practical. For the theoretical case, whenever we know from the true data generation process that the variance/covariance matrix in each class is the same or approximately the same, LDA is the preferred and more appropriate model. Furthermore, in cases where we know the true Bayes decision boundary, and it's linear, LDA makes much more sense to approximate this decision boundary then QDA. On the other hand, moving onto the practical case, even though most of the time we don't know the true data generation process, if there is evidence of equal or approximately equal variances/covariance matrices between the different classes, LDA would again be the preferred and more appropriate model. Lastly, when n is small, due to the larger number of parameters needed to be estimated in a QDA model, LDA would be the preferred model.

Exercise 4
In this exercise, we will analytically work out a Bayes classifier for a binary classification problem. Suppose we have data (X, Y) where $X \in \mathbb{R}$ and $Y \in \{1, 2\}$. We assume that the observations X from class $Y = 1$ are drawn from a $N(\mu, \sigma^2)$ distribution, and the observations X from class $Y = 2$ are drawn from a $Uniform(0, \theta)$ distribution. We also assume that the prior probability that a random observation belongs to class $Y = 1$ is $P(Y = 1) = \pi_1$, where $\pi_1 \in [0, 1]$, and thus $P(Y = 2) = 1 - \pi_1$.

Part a
In this part of the exercise, we will start by writing down $P(X = x|Y = 1)$ and $P(X = x|Y = 2)$. In this case since the distributions of $X|Y = 1$ and $X|Y = 2$ are continuous, the PDF will only serve as an approximation of these probabilities. With this being said we can say that

$$P(X = x|Y = 1) \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},\ -\infty < x < \infty$$
$$P(X = x|Y = 2) \approx \frac{1}{\theta},\ 0 \leq x \leq \theta$$

Now, we will use Bayes' rule to write down $P(Y = 1|X = x)$ and $P(Y = 2|X = x)$. We will start with defining $P(X = x)$. By the law of total probability, and the fact that $Y \in \{1, 2\}$, it follows that $P(X = x) = P(X = x|Y = 1) \cdot P(Y = 1) + P(X = x|Y = 2) \cdot P(Y = 2)$. Now, since $P(X = x|Y = 1) \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$, $P(X = x|Y = 2) = \frac{1}{\theta}$, $P(Y = 1) = \pi_1$, and $P(Y = 2) = 1 - \pi_1$, it follows that we can write $P(Y = 1|X = x)$ as follows

$$\begin{aligned}
P(Y = 1|X = x) &= \frac{P(X = x|Y = 1) \cdot P(Y = 1)}{P(X = x)} \quad \text{Bayes' rule} \\
&= \frac{P(X = x|Y = 1) \cdot P(Y = 1)}{P(X = x|Y = 1) \cdot P(Y = 1) + P(X = x|Y = 2) \cdot P(Y = 2)} \quad \text{Law of total probability} \\
&= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \cdot \pi_1}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \cdot \pi_1 + \frac{1}{\theta} \cdot (1 - \pi_1)}
\end{aligned}$$

Similarly, since $P(X = x|Y = 1) \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$, $P(X = x|Y = 2) = \frac{1}{\theta}$, $P(Y = 1) = \pi_1$, and $P(Y = 2) = 1 - \pi_1$, it follows that we can write $P(Y = 2|X = x)$ as follows

$$\begin{aligned}
P(Y = 2|X = x) &= \frac{P(X = x|Y = 2) \cdot P(Y = 2)}{P(X = x)} \quad \text{Bayes' rule} \\
&= \frac{P(X = x|Y = 2) \cdot P(Y = 2)}{P(X = x|Y = 1) \cdot P(Y = 1) + P(X = x|Y = 2) \cdot P(Y = 2)} \quad \text{Law of total probability} \\
&= \frac{\frac{1}{\theta} \cdot (1 - \pi_1)}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \cdot \pi_1 + \frac{1}{\theta} \cdot (1 - \pi_1)}
\end{aligned}$$

Obviously, these probabilities will be on the interval $[0, 1]$.

Part b
In this part of the exercise, we will derive an expression for the Bayes decision boundary. That is, the set of values of x such that $P(Y = 1|X = x) = P(Y = 2|X = x)$. We will derive this expression by starting with $P(Y = 1|X = x) = P(Y = 2|X = x)$ and manipulating the equality. Since both of these probabilities have the same denominator we will only focus on equating the numerators, which amounts to solving for x in the

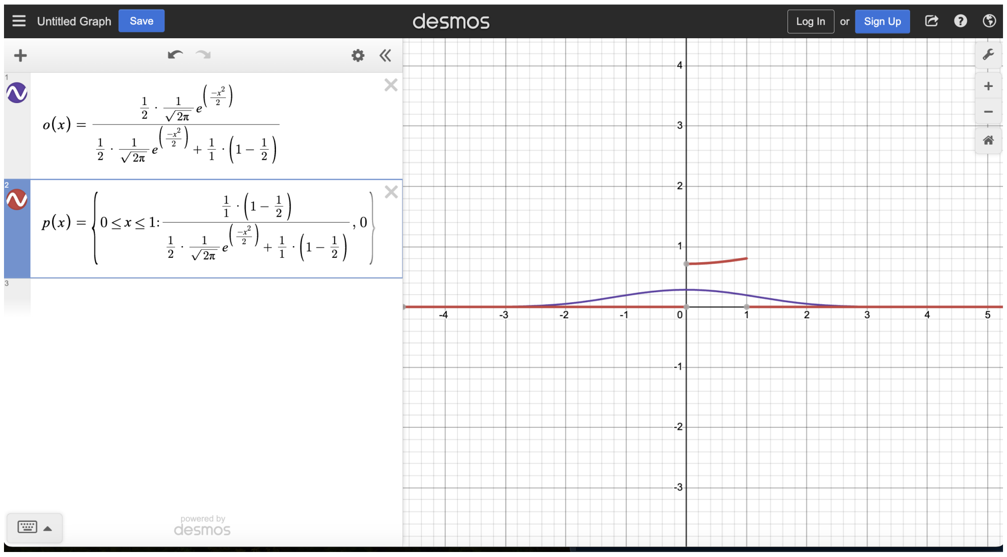
following equality

$$\begin{aligned}
\frac{\pi_1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} &= \frac{1}{\theta}(1-\pi_1) \\
\log\left(\frac{\pi_1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\right) &= \log\left(\frac{1}{\theta}(1-\pi_1)\right) \\
\log(\pi_1) + \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(e^{\frac{-(x-\mu)^2}{2\sigma^2}}\right) &= \log\left(\frac{1}{\theta}\right) + \log((1-\pi_1)) \\
\log(\pi_1) - \log(\sqrt{2\pi\sigma^2}) - \frac{(x-\mu)^2}{2\sigma^2} &= -\log(\theta) + \log((1-\pi_1)) \\
-\log(\pi_1) + \frac{1}{2}\log(2\pi\sigma^2) + \frac{(x-\mu)^2}{2\sigma^2} &= \log(\theta) - \log((1-\pi_1)) \\
\frac{(x-\mu)^2}{2\sigma^2} &= \log(\theta) + \log(\pi_1) - \log((1-\pi_1)) - \frac{1}{2}\log(2\pi\sigma^2) \\
\frac{(x-\mu)^2}{2\sigma^2} &= \log\left(\frac{\theta\pi_1}{1-\pi_1}\right) - \frac{1}{2}\log(2\pi\sigma^2) \\
(x-\mu)^2 &= 2\sigma^2\left(\log\left(\frac{\theta\pi_1}{1-\pi_1}\right) - \frac{1}{2}\log(2\pi\sigma^2)\right) \\
x-\mu &= \pm\sqrt{2\sigma^2\left(\log\left(\frac{\theta\pi_1}{1-\pi_1}\right) - \frac{1}{2}\log(2\pi\sigma^2)\right)} \\
x &= \mu \pm \sigma\sqrt{2\log\left(\frac{\theta\pi_1}{1-\pi_1}\right) - \log(2\pi\sigma^2)}
\end{aligned}$$

Therefore we have shown that $x = \mu \pm \sigma\sqrt{2\log\left(\frac{\theta\pi_1}{1-\pi_1}\right) - \log(2\pi\sigma^2)}$ is the Bayes decision boundary.

Part c
In this part of the exercise, we will suppose that we know that $\mu = 0$, $\sigma^2 = \sigma = 1$, $\theta = 1$, and $\pi = 0.5$. In particular, we will use the Bayes decision boundary from part (b), to find the range of x values that will get assigned to class 1 and class 2. Once these ranges are found, we will draw a picture of the real line (describing the values x can take) and identify the intervals that get classified to class 1 and class 2. This is done below.

As defined in the problem statement, observations of x from the class $Y = 2$ are drawn from a $Uniform(0, \theta)$ distribution, where in this case $\theta = 1$. From the definition of a uniform distribution, x can only take certain values. Namely, x can only take values between 0 and $\theta = 1$. As mentioned in office hours, the uniform distribution can be thought of as having an indicator function attached to it, indicating whether or not $x \in [0, \theta = 1]$. Which means, for any value of x, above 1 or below 0, the following probability holds true: $P(Y = 2|X = x) = 0$. The following plot of $P(Y = 2|X = x)$ and $P(Y = 1|X = x)$, as requested in the hint, will show this occurrence.

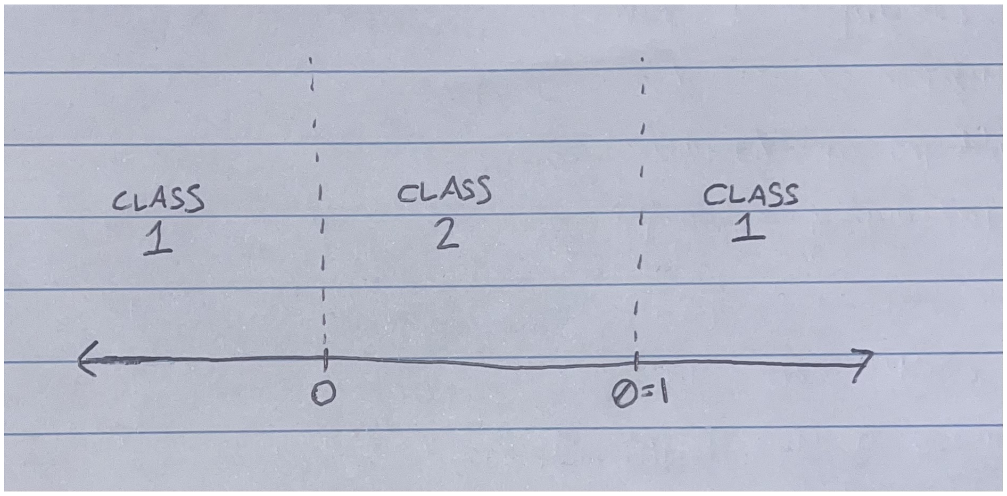


$$o(x) = \frac{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} e^{\left(\frac{-x^2}{2}\right)}}{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} e^{\left(\frac{-x^2}{2}\right)} + \frac{1}{1} \cdot \left(1 - \frac{1}{2}\right)}$$

$$p(x) = \left\{ 0 \le x \le 1 : \frac{\frac{1}{1} \cdot \left(1 - \frac{1}{2}\right)}{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} e^{\left(\frac{-x^2}{2}\right)} + \frac{1}{1} \cdot \left(1 - \frac{1}{2}\right)}, 0 \right\}$$

The above indicator function/restriction on our x values is what leads to our Bayes decision boundary being undefined when plugging in our parameter values. Therefore, we must use our knowledge of the problem to find out which values of x get labeled to each class.

As discussed above, since the uniform distribution isn't defined for values of x below 0 or greater than 1, in the ranges of $x < 0$ and $x > 1$, the probability of $Y = 2$ is zero, and thus in these ranges, x is labeled class 1. However, as seen in the above plot, whenever x is between 0 and 1, the probability of $Y = 2$ is always greater than the probability of $Y = 1$. Therefore, when $0 \le x \le 1$, the data point will be labeled class 2. The figure below will summarize this discussion using the real line.



Part d

In this part of the exercise, we will write down an expression to estimate $\pi = P(Y = 1)$ from the data. One way to estimate $\pi = P(Y = 1)$ from the data is $\hat{\pi} = \frac{n_{i:y_i=1}}{n}$, where n is the total number of data points, and $n_{i:y_i=1}$ is the total number of data points that are part of class 1. In other words the estimator is the proportion of the training data that falls into class 1. Another way to estimate $\pi = P(Y = 1)$ from the data is $\frac{\sum_{i=1}^{n} I(y_i=1)}{n}$. The numerator, in this case, is just a more concrete way of writing $n_{i:y_i=1}$. In particular, I is the indicator function that gives 1 if $y_i = 1$ and 0 otherwise.

Part e

In this part of the exercise, we will assume that $X_1, \dots, X_n \overset{iid}{\sim} N(\mu, \sigma^2)$. The goal of this part of the exercise is to show how we can use X_i to estimate the mean μ as well as the variance σ^2. One way we can estimate the mean μ is the sample mean of the data

$$\hat{\mu} = \bar{X} = \sum_{i=1}^{n} \frac{X_i}{n}.$$

Similarly, we can estimate the variance σ^2 with the sample variance of the data

$$\hat{\sigma}^2 = S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{\mu})^2$$

In our case, $\hat{\mu}$ is the MLE of μ in the normal distribution, but $\hat{\sigma}^2$ is not. Instead, the MLE for σ^2 in the normal distribution is the adjusted sample variance, defined as $\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu})^2$.

Part f
In this part of the exercise, we will now assume that $X_1, \ldots, X_n \overset{iid}{\sim} Uniform(0, \theta)$. The goal of this part of the exercise is to show how we can use X_i to estimate θ. Since the Uniform distribution requires all of its values to be in between 0 and θ, one way we can crudely estimate θ is by taking the maximum value of the X_i's, that is

$$\hat{\theta} = \max\{X_1, \ldots, X_n\}$$

If we let $\hat{\theta} = \max\{X_1, \ldots, X_n\}$, then we can be sure that all of the X_i's are less than or equal to $\hat{\theta}$, which would then satisfy that property of the Uniform distribution. This is in fact the MLE estimate for θ in the Uniform distribution.

part g
In this part of the exercise, we will suppose that the predicted classification intervals are

$$\hat{y} = \begin{cases} 1, & x \in (\infty, 5) \cup [30, 40] \cup (100, \infty) \\ 2, & x \in [5, 30) \cup (40, 100] \end{cases}$$

We will also assume that the true classification intervals for the two classes are

$$y = \begin{cases} 1, & x \in (\infty, 10) \cup (20, 40] \cup (110, \infty) \\ 2, & x \in [10, 20] \cup (40, 110] \end{cases}$$

The goal of this part of the exercise is to identify the intervals that will be misclassified as class 1 by the estimated classifier which are

$$x \in (100, 110]$$

As well as the intervals that will be misclassified as class 2 by the estimated classifier which are

$$x \in [5, 10) \cup (20, 30)$$

All of the other intervals are correctly classified by the estimated classifier.

Exercise 5
In this exercise, we will consider the logistic model $\log \frac{p}{1-p} = mx$ and $Y|X = x \sim Bernouli(p)$.

Part a
In this part of the exercise, we will show that the probability mass function of $Y|X = x$ takes the form

$$f(y|x) = \exp\left\{\frac{y\theta - b(\theta)}{\alpha} + c(y,\alpha)\right\}.$$

We will express this PMF in terms of x and not p. We will also decide what the values of θ, $b(\theta)$, α, and $c(y,\alpha)$ are in our case. Since we will be writing the PMF of $Y|X = x$ in terms of x and not p, we will solve for p in the expression $\log \frac{p}{1-p} = mx$. Doing this, we can see that p can be written as

$$\begin{aligned}
\log \frac{p}{1-p} &= mx \\
\frac{p}{1-p} &= e^{mx} \\
p &= (1-p)e^{mx} \\
p &= e^{mx} - pe^{mx} \\
p + pe^{mx} &= e^{mx} \\
p(1+e^{mx}) &= e^{mx} \\
p &= \frac{e^{mx}}{1+e^{mx}}
\end{aligned}$$

Furthermore, since $Y|X = x \sim Bernouli(p)$, it follows that Y is a Bernoulli random variable with success probability p, thus we can write its PMF as

$$f(y) = P(Y = y) = p^y(1-p)^{1-y},\ y = 0, 1$$

Therefore, rewriting the PMF of Y in terms of x we obtain the following

$$f(y|x) = P(Y = y|X = x) = \left(\frac{e^{mx}}{1+e^{mx}}\right)^y \left(1 - \frac{e^{mx}}{1+e^{mx}}\right)^{1-y},\ y = 0, 1$$

However, with the PMF in this form it will be difficult to manipulate the terms to get it in the required form, thus we will take the log of both sides and manipulate the resulting expression as follows

$$\begin{aligned}
\log\left(\left(\frac{e^{mx}}{1+e^{mx}}\right)^y \left(1 - \frac{e^{mx}}{1+e^{mx}}\right)^{1-y}\right) &= \log\left(\left(\frac{e^{mx}}{1+e^{mx}}\right)^y\right) + \log\left(\left(1 - \frac{e^{mx}}{1+e^{mx}}\right)^{1-y}\right) \\
&= y\log\left(\frac{e^{mx}}{1+e^{mx}}\right) + (1-y)\log\left(1 - \frac{e^{mx}}{1+e^{mx}}\right) \\
&= y\left(\log(e^{mx}) - \log(1+e^{mx})\right) + (1-y)\log\left(\frac{1}{1+e^{mx}}\right) \\
&= y\log(e^{mx}) - y\log(1+e^{mx}) + (1-y)\left(\log(1) - \log(1+e^{mx})\right) \\
&= ymx - y\log(1+e^{mx}) + (y-1)\log(1+e^{mx}) \\
&= ymx - y\log(1+e^{mx}) + y\log(1+e^{mx}) - \log(1+e^{mx}) \\
&= ymx - \log(1+e^{mx})
\end{aligned}$$

Now that we have an expression in the form that we want we will take the exp of both sides and obtain the following

$$\begin{aligned} f(y|x) &= \exp\{ymx - \log(1+e^{mx})\} \\ &= \exp\left\{\frac{ymx - \log(1+e^{mx})}{1} + 0\right\} \\ &= \exp\left\{\frac{y\theta - \log\left(1+e^{\theta}\right)}{1} + 0\right\} \end{aligned}$$

Thus, as seen above, $\theta = mx$, $b(\theta) = \log\left(1+e^{\theta}\right)$, $\alpha = 1$, and $c(y,\alpha) = 0$.

Part b
In this part of the exercise, we are tasked with showing that $E[Y|X=x] = b'(\theta)$ or, equivalently, $\theta = (b')^{-1}(E[Y|X=x])$. To start off this proof we will take the derivative of $b(\theta)$ which turns out to be the following

$$b'(\theta) = \frac{d}{d\theta}b(\theta) = \frac{d}{d\theta}\log\left(1+e^{\theta}\right) = \frac{e^{\theta}}{1+e^{\theta}}$$

Now we must take the expected value of $Y|X=x$. However, we know that $Y|X=x \sim \mathit{Bernoulli}(p)$ with $p = \frac{e^{mx}}{1+e^{mx}}$. Furthermore, it is well known that the expected value of a Bernoulli random variable with success probability p is simply p. Thus, in our case, $E[Y|X=x] = p = \frac{e^{mx}}{1+e^{mx}} = \frac{e^{\theta}}{1+e^{\theta}}$. Therefore we have shown that, in our case, $E[Y|X=x] = b'(\theta)$.

Part c
In this part of the exercise, we are tasked with showing that $\text{Var}[Y|X=x] = \alpha b''(\theta)$. To start off this proof we will take the second derivative of $b(\theta)$ which turns out to be the following

$$b''(\theta) = \frac{d^2}{d\theta^2}b(\theta) = \frac{d}{d\theta}b'(\theta) = \frac{d}{d\theta}\frac{e^{\theta}}{1+e^{\theta}} = \frac{(1+e^{\theta})\frac{d}{d\theta}e^{\theta} - e^{\theta}\frac{d}{d\theta}(1+e^{\theta})}{(1+e^{\theta})^2} = \frac{(1+e^{\theta})e^{\theta} - e^{\theta}e^{\theta}}{(1+e^{\theta})^2} = \frac{e^{\theta}(1+e^{\theta}-e^{\theta})}{(1+e^{\theta})^2}$$

Therefore, we can see from above that $b''(\theta) = \frac{e^{\theta}}{(1+e^{\theta})^2}$. Hence, since $\alpha = 1$ in our case, it follows that $\alpha b''(\theta) = b''(\theta)$. Now we must take the variance of $Y|X=x$. However, we know that $Y|X=x \sim \mathit{Bernoulli}(p)$ with $p = \frac{e^{mx}}{1+e^{mx}}$. Furthermore, it is well known that the variance of a Bernoulli random variable with success probability p is simply $p(1-p)$. Thus, in our case, the variance of $Y|X=x$ can be written as follows

$$\begin{aligned} \text{Var}[Y|X=x] &= p(1-p) \\ &= \frac{e^{mx}}{1+e^{mx}}\left(1 - \frac{e^{mx}}{1+e^{mx}}\right) \\ &= \frac{e^{mx}}{1+e^{mx}}\left(\frac{1}{1+e^{mx}}\right) \\ &= \frac{e^{mx}}{(1+e^{mx})^2} \\ &= \frac{e^{\theta}}{(1+e^{\theta})^2} \end{aligned}$$

Therefore we have shown that, in our case, $\text{Var}[Y|X=x] = \alpha b''(\theta)$.