Exercie 1: Linear Decision Boundaries

Part A

Lets load and important the iris data set

```
iris_data <- read.csv("./irisdata.csv")
summary(iris_data)
```

```
##   sepal_length    sepal_width     petal_length    petal_width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# Lets look at the head of the iris data set
head(iris_data)
```

```
##   sepal_length sepal_width petal_length petal_width species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# Number of rows of the iris dataset (number of individual observations)
nrow(iris_data)
```

```
## [1] 150
```

```
# All of the columns are numeric except for the species column
# They are continuous because they correspond to length and width (except for species) # The species ar
sapply(iris_data, class)
```

```
## sepal_length  sepal_width petal_length  petal_width      species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

```
# The only categorical variable is species in this iris dataset
sapply(iris_data, class)
```

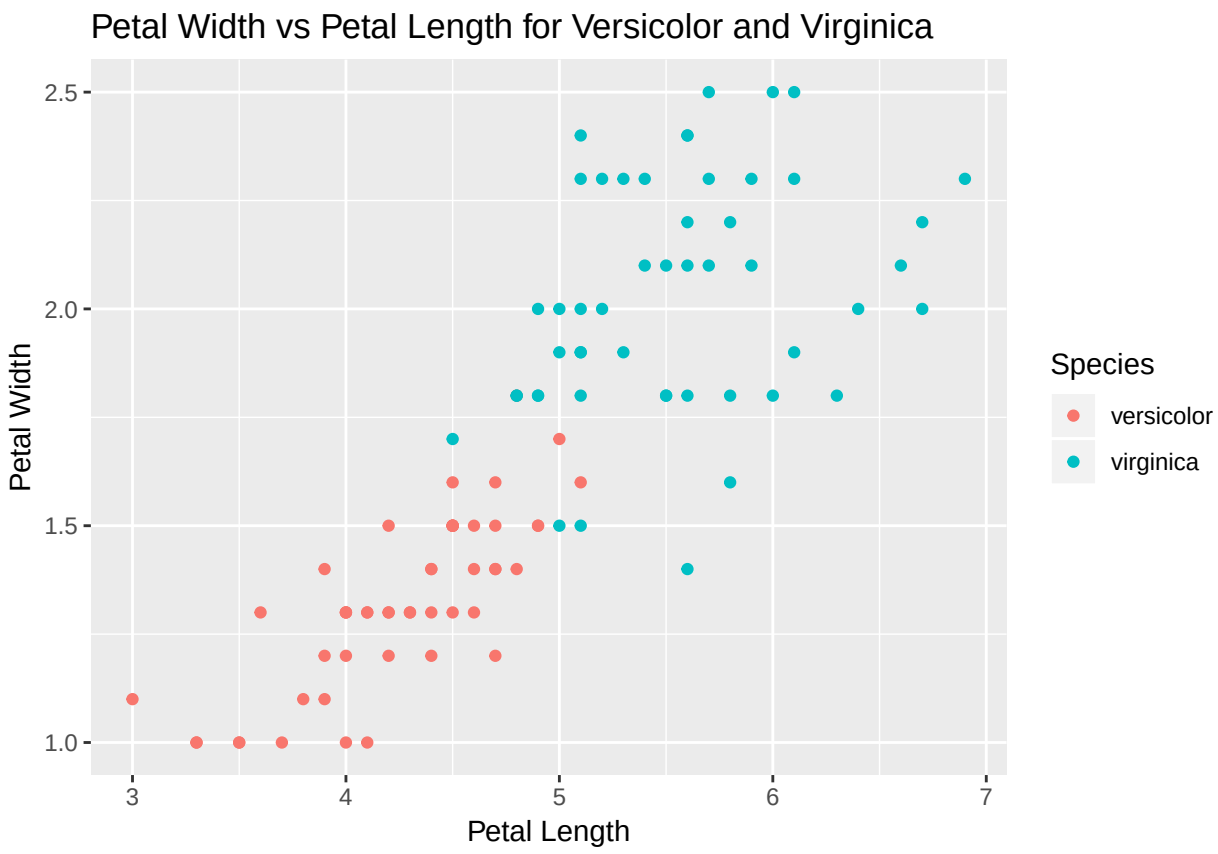
```
## sepal_length  sepal_width petal_length  petal_width      species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

From our observations it looks like there are 50 samples of three species: setosa, versicolor and virginica. Each of the samples has measurements of sepal lenght, sepal width and petal length.

Let's plot the second and third iris classses:

```r
# Lets pick the 2nd and 3rd classes: versicolor and virginica and let's plot the subset
plot1 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("Petal Width vs Petal Length for Versicolor and Virginica")

plot1
```



Part B

The following function will take in five inputs (c0, c1, c2) which are constants that will be estimated later in this exercise and (x1, x2) which correspond to petal width and petal length. The output will return a value between a probability value between 0 and 1 where all values above 0.5 will correspond to the 3rd iris class and values below 0.5 to the 2nd iris class.

```r
# inputs: c0, c1, c2, petal_width, petal_length
# outputs: value between 0 and 1 representing the probability

one_layer_neural_network <- function(c0, c1, c2, petal_width, petal_length) {

  # Linear function
  z <- c0 + c1*petal_length + c2*petal_width

  #result
  result <- 1-1/(1 + exp(-z))
```

```
  return (result)
}
```

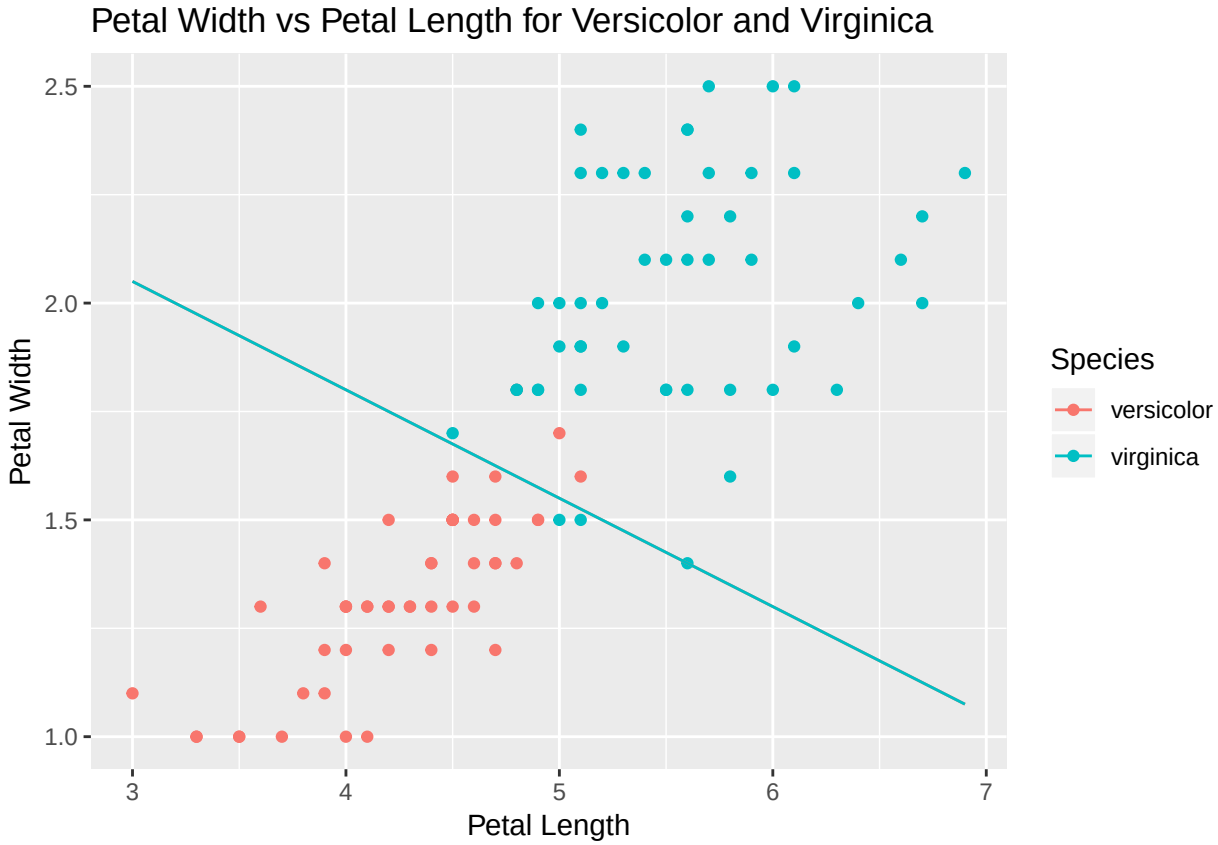
Part C

We are going to estimate the constant values such as c0 = 2.8, c1 = -1/4 and c2 = 1.

```
z_function <- function(x) (2.8 + (-1/4)*x)

# Lets pick the 2nd and 3rd classes: versicolor and virginica and let's plot the subset
plot2 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("Petal Width vs Petal Length for Versicolor and Virginica") + stat_function(fun = z_function)

plot2
```



Part D

UGH 3D???

Part E

Now lets see how are function from question b performs:

```r
# Versicolor
data1 <- iris_data %>% filter (species == 'versicolor')

for (i in 1:50){

  if (one_layer_neural_network(2.8, -1/4, -1, data1$petal_width[i], data1$petal_length[i]) < 0.5){

     resulting_flower <- "versicolor"
  } else {
    resulting_flower <- "virginica"
  }
  print (one_layer_neural_network(2.8, -1/4, -1, data1$petal_width[i], data1$petal_length[i]))
  print(paste("actual: versicolor, predicted:", resulting_flower))
}
```

```
## [1] 0.4439861
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4563613
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4812588
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3775407
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4625702
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4073334
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4937503
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.273885
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4133824
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3953209
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.2839402
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4378235
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3100255
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4439861
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3543437
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4255575
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4563613
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3153985
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4563613
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3262929
```

```
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.549834
## [1] "actual: versicolor, predicted: virginica"
## [1] 0.3775407
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4812588
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3953209
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3953209
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4255575
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.450166
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.5374298
## [1] "actual: versicolor, predicted: virginica"
## [1] 0.4563613
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.2839402
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3208213
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.294215
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3486451
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.5187412
## [1] "actual: versicolor, predicted: virginica"
## [1] 0.4563613
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4812588
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4687906
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4013123
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3834335
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3775407
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3775407
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.4378235
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3543437
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.273885
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3893608
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3658644
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3893608
```

```
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3953209
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.2788848
## [1] "actual: versicolor, predicted: versicolor"
## [1] 0.3834335
## [1] "actual: versicolor, predicted: versicolor"
```

```
# Virginica
data2 <- iris_data %>% filter (species == 'virginica')

for (j in 1:50){
  if (one_layer_neural_network(2.8, -1/4, -1, data2$petal_width[j], data2$petal_length[j]) < 0.5){

     resulting_flower <- "versicolor"
  } else {
    resulting_flower <- "virginica"
  }
  print (one_layer_neural_network(2.8, -1/4, -1, data2$petal_width[j], data2$petal_length[j]))
  print(paste("actual: virginica, predicted: ", resulting_flower))
}
```

```
## [1] 0.7685248
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5926666
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6846015
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5986877
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7005671
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7211152
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5062497
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6399161
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6106392
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7729423
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6165665
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6046791
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6626218
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6106392
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.705785
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6952967
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5926666
```

```
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7455466
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7729423
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.4875026
## [1] "actual: virginica, predicted:  versicolor"
## [1] 0.7160598
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6046791
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.705785
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5560139
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6737071
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6224593
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.549834
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5560139
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6681878
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5621765
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6513549
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6899745
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6899745
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.4937503
## [1] "actual: virginica, predicted:  versicolor"
## [1] 0.5
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7359454
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7310586
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5926666
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.549834
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6570105
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7310586
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6846015
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5926666
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.726115
```

```
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.754915
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6899745
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.5866176
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.6224593
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.7005671
## [1] "actual: virginica, predicted:  virginica"
## [1] 0.56832
## [1] "actual: virginica, predicted:  virginica"
```

Results match to what we have seen in the graph above.

Exercie 2: Neural networks

Part A

Mean squared error calculations. The inputs are in order: the data vectors (as the iris dataset that the petal length and petal width information is taken from), the parameters defining the neural network (w0, w1, w2). The pattern classes are computed inside the algorithm.

```
mean_squared_error <- function(data, w0, w1, w2) {

  # Lets setup a vector for the pattern classes
  pattern_classes <- rep(NA, 100)

  for (i in 1:100){
    pattern_classes[i] = one_layer_neural_network(w0, w1, w2, data$petal_width[i+50],
                                                  data$petal_length[i+50])
  }

  # what is returned by the program
  resulting_sum <- 0

  # Now lets compute the sum of all the errors
  for (j in 1:100){
    if (j <= 50){
      resulting_sum = resulting_sum + (pattern_classes[j] - 0)^2
    } else {
      resulting_sum = resulting_sum + (pattern_classes[j] - 1)^2
    }
  }
  return(resulting_sum/100)
}
```

Part B

I have decided to look at three different values. First, I computed the mean square error of the original and got a mean square error of 0.1489:

```r
# Mean squared error for the one we graphed before
paste0("Previous Mean Square Error: ", mean_squared_error(iris_data, 2.8, -1/4, -1))
```

```
## [1] "Previous Mean Square Error: 0.148888121336409"
```

Then I made a completely incorrect estimate such that the boundary is not even touching any of the data points and got a mean square error of 0.3225.

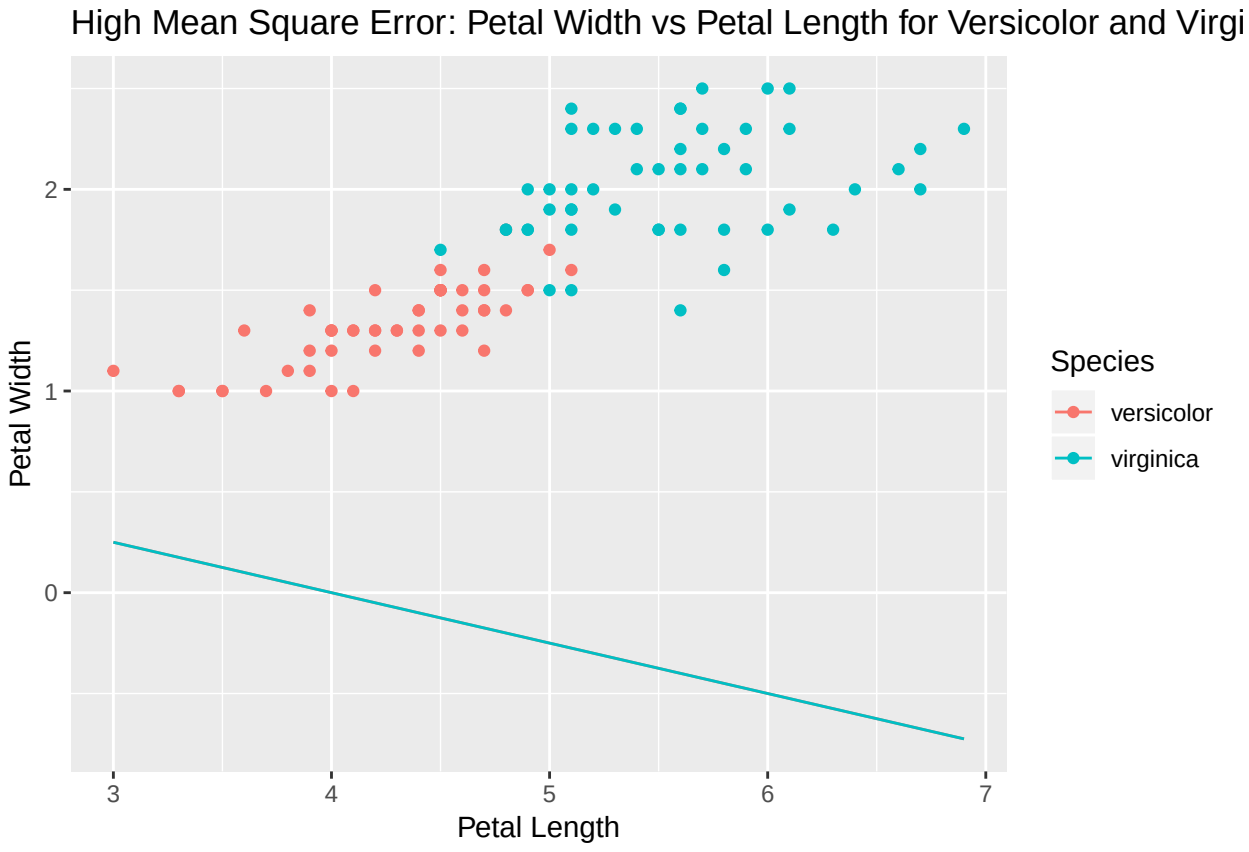
```r
# High error
paste0("High Mean Square Error: ", mean_squared_error(iris_data, 1, -1/4, -1))
```

```
## [1] "High Mean Square Error: 0.322508245611475"
```

```r
z_function2 <- function(x) (1 + (-1/4)*x)

plot3 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("High Mean Square Error: Petal Width vs Petal Length for Versicolor and Virginica") + stat_func

plot3
```



Finally, I found a slighly better boundary line that made only three points dislocated and decreased the mean square error slightly to 0.1423.

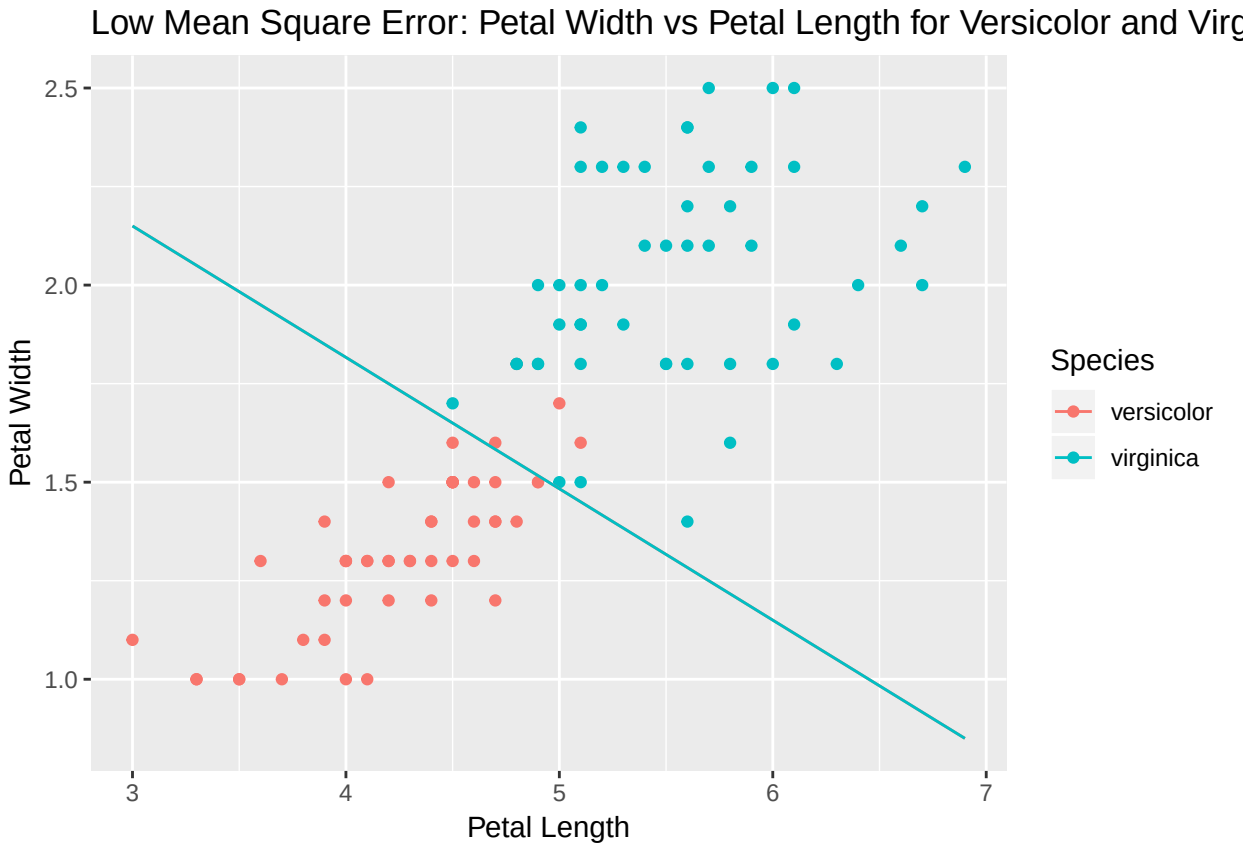
```r
# Low error
paste0("Low Mean Square Error: ", mean_squared_error(iris_data, 3.15, -1/3, -1))
```

```
## [1] "Low Mean Square Error: 0.142297154674927"
```

```r
z_function3 <- function(x) (3.15 + (-1/3)*x)

plot4 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("Low Mean Square Error: Petal Width vs Petal Length for Versicolor and Virginica") + stat_functi

plot4
```



Part C

The equation that calculates the p function as I call it (officially the class probabilities) is listed below. This is also the output of the neural network.

$$p = \sigma(w_0 + w_1 * length_p + w_2 * width_p)$$

Where σ is the sigmoid activation function. From there we derived the calculations of the mean squared error as follow:

$$meanSquareError = \frac{1}{N}\sum_{n=1}^{N}(p - y_n)^2$$

Where y_n is the known label the value that is either 0 (for versicolor) or 1 (for verginica). N is the total number of data points aka 50 for versicolor and 50 virginica so the total is 100.

The derivative of the sigmoid function is:

$$\frac{d\sigma(x)}{dx} = \sigma(x) * (1 - \sigma(x))$$

For simplification the weight column vector is the following:

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

The shape of w is 3 by 1 for the three constants.

All of the obserations can be fitted into a matrix such as:

$$\mathbf{x} = \begin{pmatrix} 1 & L_1 & W_1 \\ 1 & L_2 & W_2 \\ \vdots & \vdots & \vdots \\ 1 & L_m & W_m \end{pmatrix}$$

The shape of x is m = 100 by 3.

And:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

The shape is n = 100 by 1.

Which means that we can rewrite the meansSquareError such as:

$$meanSquareError = \frac{1}{N}(\sigma(\mathbf{x} * \mathbf{w}) - \mathbf{y})^2$$

Now let us take a derivative of the above function with respect to the output of the neural network. Then let's set it to zero and minimize it using gradient descent. We will be using the vector form.

$$\frac{\partial(meanSquareError)}{\partial\sigma} = \frac{2}{N} * \mathbf{x} * (1 - \sigma(\mathbf{x} * \mathbf{w}) * (\sigma(\mathbf{x} * \mathbf{w}) - \mathbf{y})$$

Now to substitute the probability (p):

$$\frac{\partial(meanSquareError)}{\partial\sigma} = \frac{2}{N} * \mathbf{x} * (1 - \mathbf{p}) * (\mathbf{p} - \mathbf{y})$$

Part D

???? HELP PLS

Part E