````
---
title: "CS 422 - Homework 7"
author: "Imaduddin Sheikh"
output:
  html_document:
    toc: yes
    df_print: paged
  pdf_document:
    toc: yes
  html_notebook:
    toc: yes
    toc_float: yes
---

###### Due Date: Sunday, April 10 2022 11:59:59 PM Chicago Time

******

### 2. Practicum Problems

******

### 2.1. Feed Forward Neural Networks

******

```{r}
library(keras)
library(dplyr)
library(caret)
library(tidyverse)

rm(list=ls())
setwd("~/Homework 7")

df <- read.csv("activity-small.csv")
```

******

### Part 2.1(a)

******

```{r}
set.seed(1122)
df <- df[sample(nrow(df)), ]

indx <- sample(1:nrow(df), 0.20*nrow(df))
test.df  <- df[indx, ]
train.df <- df[-indx, ]

label.test <- test.df$label
test.df$label <- NULL
test.df <- as.data.frame(scale(test.df))
test.df$label <- label.test
rm(label.test)

label.train <- train.df$label
````
````

````
train.df$label <- NULL
train.df <- as.data.frame(scale(train.df))
train.df$label <- label.train
rm(label.train)
rm(indx)
```

```{r}
x.train.df <- select(train.df, -label)
x.test.df <- select(test.df, -label)
```

```{r}
y.train.df <- train.df$label
y.train.df.ohe <- to_categorical(y.train.df)
y.test.df <- test.df$label
y.test.df.ohe <- to_categorical(y.test.df)
```

```{r}
# number <- c(0)
# loss <- c(0)
# accuracy <- c(0)
# evaluation.df <- data.frame(number, loss, accuracy)

# for (i in c(3:8)) {
#   model <- keras_model_sequential() %>%
#   layer_dense(units = i, activation="relu", input_shape=c(3)) %>%
#   layer_dense(units = 4, activation="softmax")
#
#   model %>% compile(loss = "categorical_crossentropy",
#           optimizer="adam",
#           metrics=c("accuracy"))
#
#   model %>% fit(
#     data.matrix(x.train.df),
#     y.train.df,
#     epochs=100,
#     batch_size=1,
#     validation_split=0.20)
#
#   c(loss, accuracy) %<-% (model %>% evaluate(as.matrix(x.test.df), y.test.df))
#
#   evaluation.df <- (evaluation.df %>% add_row(number = i, loss = loss, accuracy =
loss))
#   rm(model)
#
# }

# write.csv(evaluation.df, "evaluation_table.csv)
```
````

******

After running through iterations from 4 to 9 for the number of neurons in the hidden layer numerous times, I found 8 to be giving out the highest accuracy. I iterated over these number of neurons and then recorded the accuracies into a "evaluation table" for one of the instances when I trained the models. The table can be found as "evaluation_table.csv" file.
````

******

```{r}
model <- keras_model_sequential() %>%
  layer_dense(units = 8, activation="relu", input_shape=c(3)) %>%
  layer_dense(units = 4, activation="softmax")

model

model %>% compile(loss = "categorical_crossentropy",
          optimizer="adam",
          metrics=c("accuracy"))

model %>% fit(
    data.matrix(x.train.df),
    y.train.df.ohe,
    epochs=100,
    batch_size=1,
    validation_split=0.20)

c(loss, accuracy) %<-% (model %>% evaluate(as.matrix(x.test.df), y.test.df.ohe))
```

```{r}
pred.prob <- predict(model, as.matrix(x.test.df))
```

```{r}
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
```

```{r}
confusion.matrix <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

Part 2.1(a)(i)

```{r}
paste0("Overall Accuracy: ", signif(confusion.matrix[["overall"]][["Accuracy"]],
4))
```

******

Part 2.1(a)(ii)

******

```{r}
paste0('Batch Gradient Descent')
paste0("Overall Accuracy: ", signif(confusion.matrix[["overall"]][["Accuracy"]],
2))
paste0('   ', 'Class 0: Sensitivity = ', signif(confusion.matrix[["byClass"]][1,1],
2), ", Specificity = ", signif(confusion.matrix[["byClass"]][1,2], 2), ", Balanced
Accuracy = ", signif(confusion.matrix[["byClass"]][1,11], 2))
paste0('   ', 'Class 1: Sensitivity = ', signif(confusion.matrix[["byClass"]][2,1],
2), ", Specificity = ", signif(confusion.matrix[["byClass"]][2,2], 2), ", Balanced
Accuracy = ", signif(confusion.matrix[["byClass"]][2,11], 2))
```

paste0('   ', 'Class 2: Sensitivity = ', signif(confusion.matrix[["byClass"]][3,1], 2), ", Specificity = ", signif(confusion.matrix[["byClass"]][3,2], 2), ", Balanced Accuracy = ", signif(confusion.matrix[["byClass"]][3,11], 2))
paste0('   ', 'Class 3: Sensitivity = ', signif(confusion.matrix[["byClass"]][4,1], 2), ", Specificity = ", signif(confusion.matrix[["byClass"]][4,2], 2), ", Balanced Accuracy = ", signif(confusion.matrix[["byClass"]][4,11], 2))
```

******

### Part 2.1(b)

******

```{r}
batch_sizes <- c(1, 32, 64, 128, 256)
```

```{r}
create_model <- function(bs) {
  model <- keras_model_sequential() %>%
    layer_dense(units = 8, activation="relu", input_shape=c(3)) %>%
    layer_dense(units = 4, activation="softmax")

  model %>% compile(loss = "categorical_crossentropy",
          optimizer="adam",
          metrics=c("accuracy"))

  model %>% fit(
    data.matrix(x.train.df),
    y.train.df.ohe,
    epochs=100,
    batch_size=bs,
    validation_split=0.20)

  return(model)
}

```

```{r}
# Batch 1
model.1 <- NULL
begin <- Sys.time()
model.1 <- create_model(1)
end <- Sys.time()

runtime.1 <- round(as.numeric(difftime(time1 = end, time2 = begin, units = "secs")), 2)

pred.prob <- predict(model.1, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.1 <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# Batch 32
model.32 <- NULL
begin <- Sys.time()

````
model.32 <- create_model(32)
end <- Sys.time()

runtime.32 <- round(as.numeric(difftime(time1 = end, time2 = begin, units =
"secs")), 2)

pred.prob <- predict(model.32, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.32 <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# Batch 64
model.64 <- NULL
begin <- Sys.time()
model.64 <- create_model(64)
end <- Sys.time()

runtime.64 <- round(as.numeric(difftime(time1 = end, time2 = begin, units =
"secs")), 2)

pred.prob <- predict(model.64, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.64 <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# Batch 128
model.128 <- NULL
begin <- Sys.time()
model.128 <- create_model(128)
end <- Sys.time()

runtime.128 <- round(as.numeric(difftime(time1 = end, time2 = begin, units =
"secs")), 2)

pred.prob <- predict(model.128, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.128 <- confusionMatrix(as.factor(pred.class),
as.factor(y.test.df))
```

```{r}
# Batch 256
model.256 <- NULL
begin <- Sys.time()
model.256 <- create_model(256)
end <- Sys.time()

runtime.256 <- round(as.numeric(difftime(time1 = end, time2 = begin, units =
"secs")), 2)

pred.prob <- predict(model.256, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.256 <- confusionMatrix(as.factor(pred.class),
as.factor(y.test.df))
```

```{r}
````

````
  print(paste('Batch size : ', 1))
  print(paste("Time taken to train neural network: ", runtime.1, " (seconds)"))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.1[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.1[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.1[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.1[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.1[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.1[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.1[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.1[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.1[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.1[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.1[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.1[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.1[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Batch size : ', 32))
  print(paste("Time taken to train neural network: ", runtime.32, " (seconds)"))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.32[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.32[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.32[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.32[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.32[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.32[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.32[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.32[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.32[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.32[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.32[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.32[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.32[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Batch size : ', 64))
  print(paste("Time taken to train neural network: ", runtime.64, " (seconds)"))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.64[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.64[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.64[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.64[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.64[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.64[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.64[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
````

signif(confusion.matrix.64[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.64[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.64[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.64[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.64[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.64[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Batch size : ', 128))
  print(paste("Time taken to train neural network: ", runtime.128, " (seconds)"))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.128[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.128[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.128[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.128[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.128[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.128[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.128[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.128[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.128[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.128[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.128[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.128[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.128[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Batch size : ', 256))
  print(paste("Time taken to train neural network: ", runtime.256, " (seconds)"))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.256[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.256[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.256[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.256[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.256[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.256[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.256[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.256[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.256[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.256[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.256[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.256[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.256[["byClass"]][4,11], 2)))
```

******

Part 2.1(c)
```

******

Part 2.1(c)(i)

******

<span style="color: blue;">The increasing number of batch sizes results in more training samples picked up in a batch, and hence lesser computing time will be taken to compute errors and average errors compared to a batch size of 1 where each sample is treated as batch for a constant central processing prowess.<span>

******

Part 2.1(c)(ii)

******

<span style="color: blue;">As batch size increased, balanced accuracy decreased except for batch size 128. Balanced Accuracy of batch size 1 is the highest. The specificity, sensitivity and balanced accuracy of Class 0 in all batch sizes remain above 0.90. However, Sensitivity of Class 3 declined per increasing batch size. The specificity, Sensitivity and Balanced accuracy across batch sizes 32, 64 and 128 appear to be fluctuating across all 3 classes however their fluctuations are highlighted by their overall accuracies. The model with batch size 256 appear to have given the worst performance. Models with lower batch sizes appear to have performed better due to more examples being trained on them per iterations.<span>

******

Part 2.1(d)

******
```{r}
create_model.n <- function(batch.size, neurons) {
  model <- keras_model_sequential() %>%
    layer_dense(units = 8, activation="relu", input_shape=c(3)) %>%
    layer_dense(units = neurons, activation="relu") %>%
    layer_dense(units = 4, activation="softmax")

  model %>% compile(loss = "categorical_crossentropy",
          optimizer="adam",
          metrics=c("accuracy"))

  model %>% fit(
    data.matrix(x.train.df),
    y.train.df.ohe,
    epochs=100,
    batch_size=batch.size,
    validation_split=0.20)

  return(model)
}
```

```{r}
# New Hidden Layer [number of neurons = 6] & Batch Size = 1
model.6n <- create_model.n(1, 6)

pred.prob <- predict(model.6n, as.matrix(x.test.df))
```

````
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.6n <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# New Hidden Layer [number of neurons = 7] & Batch Size = 1
model.7n <- create_model.n(1, 7)

pred.prob <- predict(model.7n, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.7n <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# New Hidden Layer [number of neurons = 8] & Batch Size = 1
model.8n <- create_model.n(1, 8)

pred.prob <- predict(model.8n, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.8n <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

```{r}
# New Hidden Layer [number of neurons = 9] & Batch Size = 1
model.9n <- create_model.n(1, 9)

pred.prob <- predict(model.9n, as.matrix(x.test.df))
pred.class <- apply(pred.prob, 1, function(x) which.max(x)-1)
confusion.matrix.9n <- confusionMatrix(as.factor(pred.class), as.factor(y.test.df))
```

******

### Part 2.1(d)(i) & Part 2.1(d)(ii)

******

```{r}
  print(paste('Second Hidden Layer of Neurons : ', 6))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.6n[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.6n[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.6n[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.6n[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.6n[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.6n[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.6n[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.6n[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.6n[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.6n[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.6n[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.6n[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.6n[["byClass"]][4,11], 2)))
```
````
````

```{r}
  print(paste('Second Hidden Layer of Neurons : ', 7))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.7n[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.7n[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.7n[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.7n[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.7n[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.7n[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.7n[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.7n[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.7n[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.7n[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.7n[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.7n[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.7n[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Second Hidden Layer of Neurons : ', 8))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.8n[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.8n[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.8n[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.8n[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.8n[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.8n[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.8n[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.8n[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.8n[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.8n[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.8n[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.8n[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.8n[["byClass"]][4,11], 2)))
```

```{r}
  print(paste('Second Hidden Layer of Neurons : ', 9))
  print(paste("Overall Accuracy: ", signif(confusion.matrix.9n[["overall"]]
[["Accuracy"]], 2)))
  print(paste('   ', 'Class 0: Sensitivity = ',
signif(confusion.matrix.9n[["byClass"]][1,1], 2), ", Specificity = ",
signif(confusion.matrix.9n[["byClass"]][1,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.9n[["byClass"]][1,11], 2)))
  print(paste('   ', 'Class 1: Sensitivity = ',
signif(confusion.matrix.9n[["byClass"]][2,1], 2), ", Specificity = ",
signif(confusion.matrix.9n[["byClass"]][2,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.9n[["byClass"]][2,11], 2)))
  print(paste('   ', 'Class 2: Sensitivity = ',
signif(confusion.matrix.9n[["byClass"]][3,1], 2), ", Specificity = ",
signif(confusion.matrix.9n[["byClass"]][3,2], 2), ", Balanced Accuracy = ",
```

```
signif(confusion.matrix.9n[["byClass"]][3,11], 2)))
  print(paste('   ', 'Class 3: Sensitivity = ',
signif(confusion.matrix.9n[["byClass"]][4,1], 2), ", Specificity = ",
signif(confusion.matrix.9n[["byClass"]][4,2], 2), ", Balanced Accuracy = ",
signif(confusion.matrix.9n[["byClass"]][4,11], 2)))
```

******

Part(2.1)(d)(ii)(a)

******

I took between 6 and 9 the numbers of neurons that I thought would give me the best chance of improving my accuracy. After iterating through numerous number of neurons, adding another hidden layer doesn't necessarily mean that your model will improve. In my case, I found the number 7 of the secondary hidden layer to be good as it significantly improved my model's overall accuracy to 0.83.

******