Week4_SimonsenHomework

Steven Simonsen

2024-09-22

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
setwd("C:\\Users\\steve\\OneDrive\\Documents\\School\\DSE6211\\Week4")

data <- read.csv("lab_4_data.csv")

set.seed(42)
training_ind <- createDataPartition(data$lodgepole_pine,
                                    p = 0.75,
                                    list = FALSE,
                                    times = 1)

training_set <- data[training_ind, ]
test_set <- data[-training_ind, ]

unique(training_set$wilderness_area)
```

```
## [1] "wilderness_area_1" "wilderness_area_3" "wilderness_area_4"
## [4] "wilderness_area_2"
```

```
unique(training_set$soil_type)
```

```
##  [1] "soil_type_18" "soil_type_30" "soil_type_12" "soil_type_29" "soil_type_20"
##  [6] "soil_type_23" "soil_type_22" "soil_type_10" "soil_type_11" "soil_type_5"
## [11] "soil_type_17" "soil_type_13" "soil_type_31" "soil_type_2"  "soil_type_33"
## [16] "soil_type_32" "soil_type_6"  "soil_type_14" "soil_type_39" "soil_type_3"
## [21] "soil_type_16" "soil_type_40" "soil_type_4"  "soil_type_38" "soil_type_24"
## [26] "soil_type_35" "soil_type_27" "soil_type_1"  "soil_type_19" "soil_type_8"
## [31] "soil_type_9"  "soil_type_28" "soil_type_34" "soil_type_37" "soil_type_21"
```

```
## [36] "soil_type_36" "soil_type_26" "soil_type_25"
```

```r
top_20_soil_types <- training_set %>%
  group_by(soil_type) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  select(soil_type) %>%
  top_n(20)
```

```
## Selecting by soil_type
```

```r
training_set$soil_type <- ifelse(training_set$soil_type %in% top_20_soil_types$soil_type,
                                 training_set$soil_type,
                                 "other")

training_set$wilderness_area <- factor(training_set$wilderness_area)
training_set$soil_type <- factor(training_set$soil_type)

class(training_set$wilderness_area)
```

```
## [1] "factor"
```

```r
class(training_set$soil_type)
```

```
## [1] "factor"
```

```r
levels(training_set$wilderness_area)
```

```
## [1] "wilderness_area_1" "wilderness_area_2" "wilderness_area_3"
## [4] "wilderness_area_4"
```

```r
levels(training_set$soil_type)
```

```
##  [1] "other"        "soil_type_27" "soil_type_28" "soil_type_29" "soil_type_3"
##  [6] "soil_type_30" "soil_type_31" "soil_type_32" "soil_type_33" "soil_type_34"
## [11] "soil_type_35" "soil_type_36" "soil_type_37" "soil_type_38" "soil_type_39"
## [16] "soil_type_4"  "soil_type_40" "soil_type_5"  "soil_type_6"  "soil_type_8"
## [21] "soil_type_9"
```

```r
onehot_encoder <- dummyVars(~ wilderness_area + soil_type,
                            training_set[, c("wilderness_area", "soil_type")],
                            levelsOnly = TRUE,
                            fullRank = TRUE)


onehot_enc_training <- predict(onehot_encoder,
                               training_set[, c("wilderness_area", "soil_type")])


training_set <- cbind(training_set, onehot_enc_training)



test_set$soil_type <- ifelse(test_set$soil_type %in% top_20_soil_types$soil_type,
                             test_set$soil_type,
                             "other")
```

```r
test_set$wilderness_area <- factor(test_set$wilderness_area)
test_set$soil_type <- factor(test_set$soil_type)

onehot_enc_test <- predict(onehot_encoder, test_set[, c("wilderness_area", "soil_type")])

test_set <- cbind(test_set, onehot_enc_test)

test_set[, -c(11:13)] <- scale(test_set[, -c(11:13)],
                               center = apply(training_set[, -c(11:13)], 2, mean),
                               scale = apply(training_set[, -c(11:13)], 2, sd))

training_set[, -c(11:13)] <- scale(training_set[, -c(11:13)])


training_features <- array(data = unlist(training_set[, -c(11:13)]),
                           dim = c(nrow(training_set), 33))

training_labels <- array(data = unlist(training_set[, 13]),
                         dim = c(nrow(training_set)))

test_features <- array(data = unlist(test_set[, -c(11:13)]),
                       dim = c(nrow(test_set), 33))

test_labels <- array(data = unlist(test_set[, 13]),
                     dim = c(nrow(test_set)))

library(reticulate)
library(tensorflow)
```

```
##
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':
##
##     train
```

```r
library(keras3)
```

```
##
## Attaching package: 'keras3'

## The following objects are masked from 'package:tensorflow':
##
##     set_random_seed, shape
```

```r
use_virtualenv("my_tf_workspace")

model1 <- keras_model_sequential() %>%
  layer_dense(units = 20, activation = "relu") %>%
  layer_dense(units = 10, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

compile(model1,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")
```

```r
set.seed(42)
history1 <- fit(model1, training_features, training_labels,
               epochs = 100, batch_size = 512, validation_split = 0.33)
```

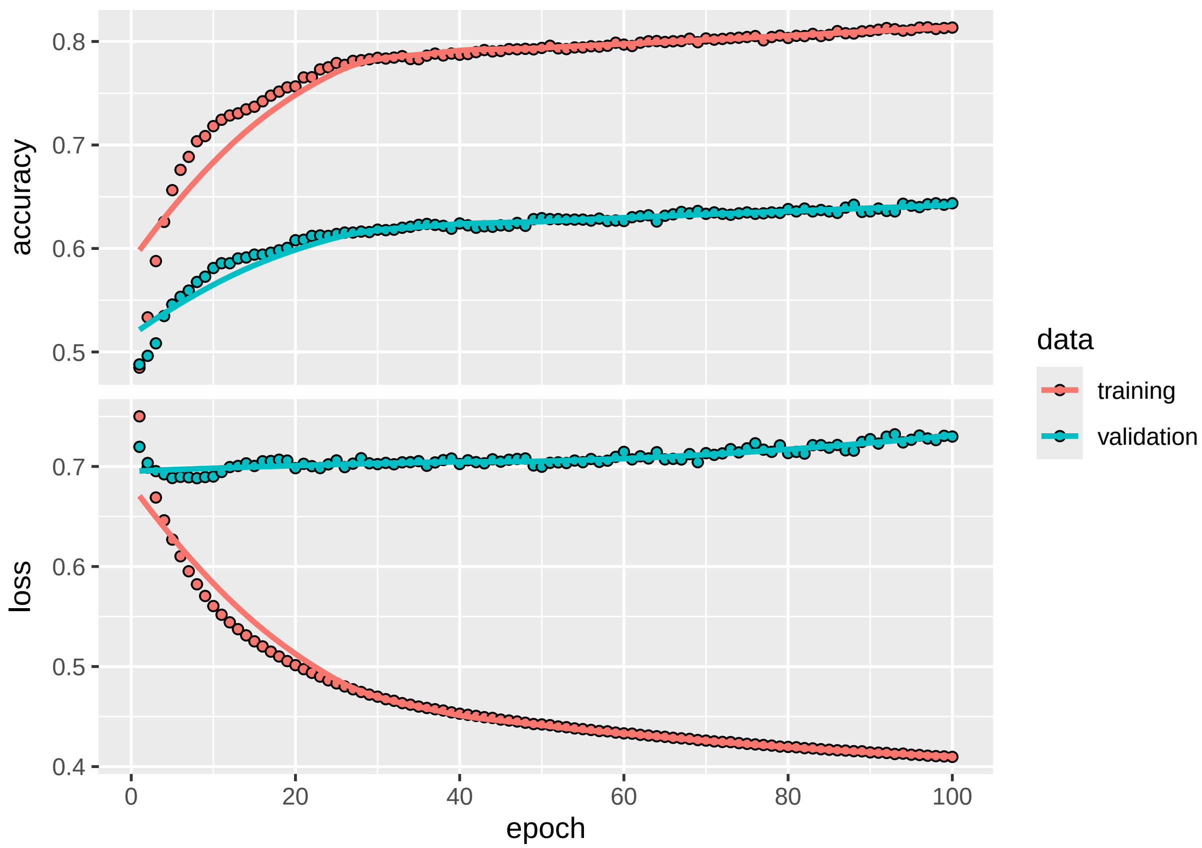
```
## Epoch 1/100
## 9/9 - 1s - 72ms/step - accuracy: 0.4848 - loss: 0.7501 - val_accuracy: 0.4880 - val_loss: 0.7196
## Epoch 2/100
## 9/9 - 0s - 5ms/step - accuracy: 0.5335 - loss: 0.6989 - val_accuracy: 0.4963 - val_loss: 0.7035
## Epoch 3/100
## 9/9 - 0s - 5ms/step - accuracy: 0.5878 - loss: 0.6690 - val_accuracy: 0.5083 - val_loss: 0.6953
## Epoch 4/100
## 9/9 - 0s - 5ms/step - accuracy: 0.6257 - loss: 0.6462 - val_accuracy: 0.5348 - val_loss: 0.6922
## Epoch 5/100
## 9/9 - 0s - 4ms/step - accuracy: 0.6563 - loss: 0.6269 - val_accuracy: 0.5459 - val_loss: 0.6885
## Epoch 6/100
## 9/9 - 0s - 5ms/step - accuracy: 0.6760 - loss: 0.6102 - val_accuracy: 0.5533 - val_loss: 0.6896
## Epoch 7/100
## 9/9 - 0s - 5ms/step - accuracy: 0.6885 - loss: 0.5952 - val_accuracy: 0.5593 - val_loss: 0.6892
## Epoch 8/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7036 - loss: 0.5823 - val_accuracy: 0.5677 - val_loss: 0.6883
## Epoch 9/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7086 - loss: 0.5706 - val_accuracy: 0.5728 - val_loss: 0.6894
## Epoch 10/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7182 - loss: 0.5605 - val_accuracy: 0.5811 - val_loss: 0.6900
## Epoch 11/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7244 - loss: 0.5518 - val_accuracy: 0.5857 - val_loss: 0.6945
## Epoch 12/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7285 - loss: 0.5442 - val_accuracy: 0.5857 - val_loss: 0.6994
## Epoch 13/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7305 - loss: 0.5374 - val_accuracy: 0.5904 - val_loss: 0.7005
## Epoch 14/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7344 - loss: 0.5312 - val_accuracy: 0.5913 - val_loss: 0.7031
## Epoch 15/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7369 - loss: 0.5252 - val_accuracy: 0.5941 - val_loss: 0.7005
## Epoch 16/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7422 - loss: 0.5201 - val_accuracy: 0.5941 - val_loss: 0.7052
## Epoch 17/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7477 - loss: 0.5149 - val_accuracy: 0.5959 - val_loss: 0.7055
## Epoch 18/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7515 - loss: 0.5101 - val_accuracy: 0.5982 - val_loss: 0.7068
## Epoch 19/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7557 - loss: 0.5054 - val_accuracy: 0.6006 - val_loss: 0.7059
## Epoch 20/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7566 - loss: 0.5014 - val_accuracy: 0.6080 - val_loss: 0.6983
## Epoch 21/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7652 - loss: 0.4974 - val_accuracy: 0.6084 - val_loss: 0.7027
## Epoch 22/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7655 - loss: 0.4937 - val_accuracy: 0.6121 - val_loss: 0.7003
## Epoch 23/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7730 - loss: 0.4899 - val_accuracy: 0.6126 - val_loss: 0.6984
## Epoch 24/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7751 - loss: 0.4862 - val_accuracy: 0.6121 - val_loss: 0.7021
## Epoch 25/100
```

```
## 9/9 - 0s - 6ms/step - accuracy: 0.7792 - loss: 0.4832 - val_accuracy: 0.6140 - val_loss: 0.7061
## Epoch 26/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7773 - loss: 0.4801 - val_accuracy: 0.6154 - val_loss: 0.6994
## Epoch 27/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7812 - loss: 0.4772 - val_accuracy: 0.6154 - val_loss: 0.7028
## Epoch 28/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7819 - loss: 0.4746 - val_accuracy: 0.6163 - val_loss: 0.7083
## Epoch 29/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7828 - loss: 0.4720 - val_accuracy: 0.6158 - val_loss: 0.7032
## Epoch 30/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7842 - loss: 0.4699 - val_accuracy: 0.6182 - val_loss: 0.7023
## Epoch 31/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7835 - loss: 0.4674 - val_accuracy: 0.6177 - val_loss: 0.7036
## Epoch 32/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7844 - loss: 0.4658 - val_accuracy: 0.6182 - val_loss: 0.7022
## Epoch 33/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7858 - loss: 0.4634 - val_accuracy: 0.6200 - val_loss: 0.7042
## Epoch 34/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7831 - loss: 0.4619 - val_accuracy: 0.6209 - val_loss: 0.7043
## Epoch 35/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7828 - loss: 0.4600 - val_accuracy: 0.6228 - val_loss: 0.7053
## Epoch 36/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7863 - loss: 0.4586 - val_accuracy: 0.6237 - val_loss: 0.7007
## Epoch 37/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7883 - loss: 0.4573 - val_accuracy: 0.6228 - val_loss: 0.7041
## Epoch 38/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7865 - loss: 0.4560 - val_accuracy: 0.6219 - val_loss: 0.7064
## Epoch 39/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7883 - loss: 0.4542 - val_accuracy: 0.6191 - val_loss: 0.7080
## Epoch 40/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7872 - loss: 0.4529 - val_accuracy: 0.6242 - val_loss: 0.7025
## Epoch 41/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7879 - loss: 0.4517 - val_accuracy: 0.6223 - val_loss: 0.7062
## Epoch 42/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7897 - loss: 0.4507 - val_accuracy: 0.6200 - val_loss: 0.7043
## Epoch 43/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7917 - loss: 0.4494 - val_accuracy: 0.6214 - val_loss: 0.7032
## Epoch 44/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7904 - loss: 0.4486 - val_accuracy: 0.6209 - val_loss: 0.7072
## Epoch 45/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7908 - loss: 0.4470 - val_accuracy: 0.6223 - val_loss: 0.7048
## Epoch 46/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7926 - loss: 0.4461 - val_accuracy: 0.6219 - val_loss: 0.7067
## Epoch 47/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7924 - loss: 0.4451 - val_accuracy: 0.6247 - val_loss: 0.7075
## Epoch 48/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7929 - loss: 0.4438 - val_accuracy: 0.6219 - val_loss: 0.7080
## Epoch 49/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7924 - loss: 0.4425 - val_accuracy: 0.6284 - val_loss: 0.7010
## Epoch 50/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7938 - loss: 0.4421 - val_accuracy: 0.6293 - val_loss: 0.6998
## Epoch 51/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7958 - loss: 0.4414 - val_accuracy: 0.6284 - val_loss: 0.7036
## Epoch 52/100
```

```
## 9/9 - 0s - 6ms/step - accuracy: 0.7933 - loss: 0.4401 - val_accuracy: 0.6284 - val_loss: 0.7041
## Epoch 53/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7926 - loss: 0.4394 - val_accuracy: 0.6279 - val_loss: 0.7034
## Epoch 54/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7942 - loss: 0.4382 - val_accuracy: 0.6279 - val_loss: 0.7059
## Epoch 55/100
## 9/9 - 0s - 5ms/step - accuracy: 0.7942 - loss: 0.4374 - val_accuracy: 0.6279 - val_loss: 0.7043
## Epoch 56/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7954 - loss: 0.4367 - val_accuracy: 0.6270 - val_loss: 0.7075
## Epoch 57/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7949 - loss: 0.4356 - val_accuracy: 0.6288 - val_loss: 0.7046
## Epoch 58/100
## 9/9 - 0s - 8ms/step - accuracy: 0.7958 - loss: 0.4352 - val_accuracy: 0.6265 - val_loss: 0.7058
## Epoch 59/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7988 - loss: 0.4340 - val_accuracy: 0.6270 - val_loss: 0.7096
## Epoch 60/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7970 - loss: 0.4332 - val_accuracy: 0.6265 - val_loss: 0.7146
## Epoch 61/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7956 - loss: 0.4329 - val_accuracy: 0.6302 - val_loss: 0.7071
## Epoch 62/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7988 - loss: 0.4317 - val_accuracy: 0.6311 - val_loss: 0.7102
## Epoch 63/100
## 9/9 - 0s - 5ms/step - accuracy: 0.8002 - loss: 0.4311 - val_accuracy: 0.6321 - val_loss: 0.7080
## Epoch 64/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8004 - loss: 0.4303 - val_accuracy: 0.6260 - val_loss: 0.7143
## Epoch 65/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7995 - loss: 0.4298 - val_accuracy: 0.6316 - val_loss: 0.7071
## Epoch 66/100
## 9/9 - 0s - 5ms/step - accuracy: 0.8002 - loss: 0.4288 - val_accuracy: 0.6330 - val_loss: 0.7076
## Epoch 67/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8004 - loss: 0.4282 - val_accuracy: 0.6353 - val_loss: 0.7070
## Epoch 68/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8027 - loss: 0.4277 - val_accuracy: 0.6339 - val_loss: 0.7124
## Epoch 69/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7993 - loss: 0.4265 - val_accuracy: 0.6362 - val_loss: 0.7043
## Epoch 70/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8029 - loss: 0.4260 - val_accuracy: 0.6335 - val_loss: 0.7133
## Epoch 71/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8018 - loss: 0.4252 - val_accuracy: 0.6348 - val_loss: 0.7116
## Epoch 72/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8025 - loss: 0.4246 - val_accuracy: 0.6335 - val_loss: 0.7131
## Epoch 73/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8032 - loss: 0.4245 - val_accuracy: 0.6325 - val_loss: 0.7173
## Epoch 74/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8036 - loss: 0.4235 - val_accuracy: 0.6339 - val_loss: 0.7141
## Epoch 75/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8043 - loss: 0.4227 - val_accuracy: 0.6348 - val_loss: 0.7181
## Epoch 76/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8052 - loss: 0.4222 - val_accuracy: 0.6335 - val_loss: 0.7232
## Epoch 77/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8011 - loss: 0.4216 - val_accuracy: 0.6339 - val_loss: 0.7168
## Epoch 78/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8043 - loss: 0.4210 - val_accuracy: 0.6348 - val_loss: 0.7146
## Epoch 79/100
```

```
## 9/9 - 0s - 7ms/step - accuracy: 0.8057 - loss: 0.4200 - val_accuracy: 0.6344 - val_loss: 0.7211
## Epoch 80/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8034 - loss: 0.4196 - val_accuracy: 0.6381 - val_loss: 0.7134
## Epoch 81/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8054 - loss: 0.4193 - val_accuracy: 0.6358 - val_loss: 0.7146
## Epoch 82/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8052 - loss: 0.4184 - val_accuracy: 0.6386 - val_loss: 0.7129
## Epoch 83/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8073 - loss: 0.4181 - val_accuracy: 0.6358 - val_loss: 0.7213
## Epoch 84/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8052 - loss: 0.4174 - val_accuracy: 0.6372 - val_loss: 0.7213
## Epoch 85/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8063 - loss: 0.4168 - val_accuracy: 0.6358 - val_loss: 0.7188
## Epoch 86/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8100 - loss: 0.4163 - val_accuracy: 0.6344 - val_loss: 0.7215
## Epoch 87/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8079 - loss: 0.4160 - val_accuracy: 0.6395 - val_loss: 0.7162
## Epoch 88/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8077 - loss: 0.4154 - val_accuracy: 0.6423 - val_loss: 0.7158
## Epoch 89/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8098 - loss: 0.4153 - val_accuracy: 0.6353 - val_loss: 0.7246
## Epoch 90/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8102 - loss: 0.4142 - val_accuracy: 0.6358 - val_loss: 0.7273
## Epoch 91/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8114 - loss: 0.4139 - val_accuracy: 0.6386 - val_loss: 0.7229
## Epoch 92/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8130 - loss: 0.4136 - val_accuracy: 0.6362 - val_loss: 0.7298
## Epoch 93/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8118 - loss: 0.4126 - val_accuracy: 0.6358 - val_loss: 0.7322
## Epoch 94/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8105 - loss: 0.4129 - val_accuracy: 0.6432 - val_loss: 0.7240
## Epoch 95/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8111 - loss: 0.4119 - val_accuracy: 0.6413 - val_loss: 0.7267
## Epoch 96/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8134 - loss: 0.4116 - val_accuracy: 0.6399 - val_loss: 0.7309
## Epoch 97/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8137 - loss: 0.4109 - val_accuracy: 0.6427 - val_loss: 0.7280
## Epoch 98/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8121 - loss: 0.4105 - val_accuracy: 0.6437 - val_loss: 0.7264
## Epoch 99/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8130 - loss: 0.4102 - val_accuracy: 0.6423 - val_loss: 0.7307
## Epoch 100/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8134 - loss: 0.4097 - val_accuracy: 0.6437 - val_loss: 0.7299
```

```r
plot(history1)
```




```r
set.seed(42)
predictions1 <- predict(model1, test_features)
```

```
## 69/69 - 0s - 2ms/step
```

```r
head(predictions1, 10)
```

```
##            [,1]
##  [1,] 0.8892217
##  [2,] 0.7098093
##  [3,] 0.9327543
##  [4,] 0.5875900
##  [5,] 0.3151655
##  [6,] 0.7449991
##  [7,] 0.8573032
##  [8,] 0.5073557
##  [9,] 0.8812863
## [10,] 0.8300340
```

```r
predicted_class1 <- (predictions1[, 1] >= 0.5) * 1
head(predicted_class1, 10)
```

```
##  [1] 1 1 1 1 0 1 1 1 1 1
```

#Exercises

1) Copy and paste the loss and accuracy curves obtained from running the code above (note, the curves will be slightly different than those shown in this lab).

See above plot(history) graphs.

2) Change the hidden layers to have 50 units and 25 units, respectively, and re-run the code. Copy and paste the new loss and accuracy curves.

See output from the plot(history) code below.

```
model2 <- keras_model_sequential() %>%
  layer_dense(units = 50, activation = "relu") %>%
  layer_dense(units = 25, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

compile(model2,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")

set.seed(42)
history2 <- fit(model2, training_features, training_labels,
               epochs = 100, batch_size = 512, validation_split = 0.33)
```

```
## Epoch 1/100
## 9/9 - 1s - 116ms/step - accuracy: 0.5627 - loss: 0.7135 - val_accuracy: 0.5394 - val_loss: 0.6843
## Epoch 2/100
## 9/9 - 0s - 9ms/step - accuracy: 0.6399 - loss: 0.6388 - val_accuracy: 0.5348 - val_loss: 0.6833
## Epoch 3/100
## 9/9 - 0s - 8ms/step - accuracy: 0.6899 - loss: 0.6051 - val_accuracy: 0.5283 - val_loss: 0.6832
## Epoch 4/100
## 9/9 - 0s - 8ms/step - accuracy: 0.7084 - loss: 0.5802 - val_accuracy: 0.5412 - val_loss: 0.6725
## Epoch 5/100
## 9/9 - 0s - 8ms/step - accuracy: 0.7212 - loss: 0.5610 - val_accuracy: 0.5565 - val_loss: 0.6697
## Epoch 6/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7349 - loss: 0.5452 - val_accuracy: 0.5607 - val_loss: 0.6742
## Epoch 7/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7447 - loss: 0.5316 - val_accuracy: 0.5700 - val_loss: 0.6776
## Epoch 8/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7531 - loss: 0.5202 - val_accuracy: 0.5788 - val_loss: 0.6808
## Epoch 9/100
## 9/9 - 0s - 10ms/step - accuracy: 0.7600 - loss: 0.5097 - val_accuracy: 0.5992 - val_loss: 0.6719
## Epoch 10/100
## 9/9 - 0s - 8ms/step - accuracy: 0.7712 - loss: 0.5001 - val_accuracy: 0.6070 - val_loss: 0.6722
## Epoch 11/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7760 - loss: 0.4923 - val_accuracy: 0.6237 - val_loss: 0.6659
## Epoch 12/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7780 - loss: 0.4842 - val_accuracy: 0.6172 - val_loss: 0.6739
## Epoch 13/100
## 9/9 - 0s - 10ms/step - accuracy: 0.7826 - loss: 0.4777 - val_accuracy: 0.6196 - val_loss: 0.6776
## Epoch 14/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7840 - loss: 0.4729 - val_accuracy: 0.6219 - val_loss: 0.6790
## Epoch 15/100
## 9/9 - 0s - 10ms/step - accuracy: 0.7865 - loss: 0.4675 - val_accuracy: 0.6344 - val_loss: 0.6686
## Epoch 16/100
## 9/9 - 0s - 10ms/step - accuracy: 0.7851 - loss: 0.4640 - val_accuracy: 0.6335 - val_loss: 0.6683
## Epoch 17/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7876 - loss: 0.4603 - val_accuracy: 0.6432 - val_loss: 0.6634
## Epoch 18/100
```

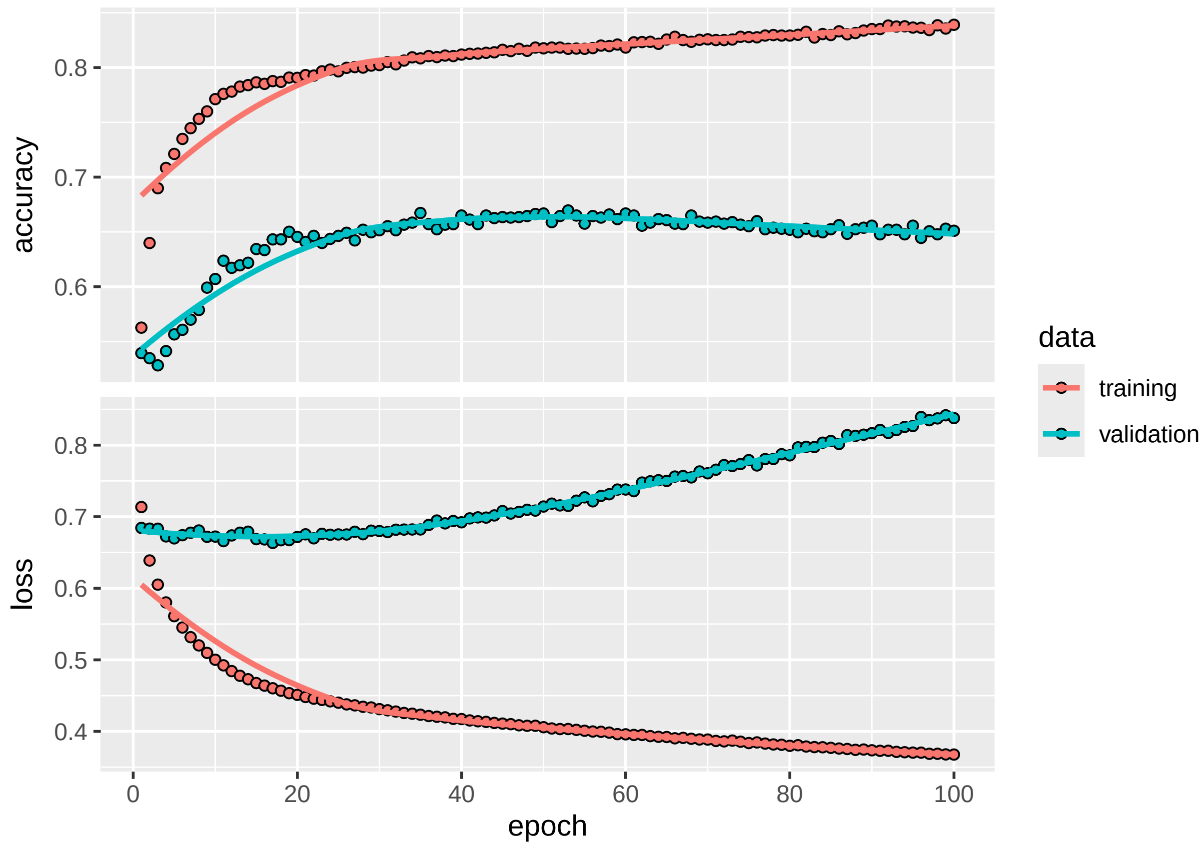
```
## 9/9 - 0s - 11ms/step - accuracy: 0.7869 - loss: 0.4569 - val_accuracy: 0.6432 - val_loss: 0.6672
## Epoch 19/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7908 - loss: 0.4534 - val_accuracy: 0.6501 - val_loss: 0.6673
## Epoch 20/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7906 - loss: 0.4511 - val_accuracy: 0.6455 - val_loss: 0.6717
## Epoch 21/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7931 - loss: 0.4481 - val_accuracy: 0.6409 - val_loss: 0.6754
## Epoch 22/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7926 - loss: 0.4457 - val_accuracy: 0.6464 - val_loss: 0.6698
## Epoch 23/100
## 9/9 - 0s - 8ms/step - accuracy: 0.7965 - loss: 0.4440 - val_accuracy: 0.6399 - val_loss: 0.6761
## Epoch 24/100
## 9/9 - 0s - 9ms/step - accuracy: 0.7981 - loss: 0.4420 - val_accuracy: 0.6437 - val_loss: 0.6748
## Epoch 25/100
## 9/9 - 0s - 7ms/step - accuracy: 0.7965 - loss: 0.4401 - val_accuracy: 0.6464 - val_loss: 0.6752
## Epoch 26/100
## 9/9 - 0s - 6ms/step - accuracy: 0.7997 - loss: 0.4377 - val_accuracy: 0.6492 - val_loss: 0.6752
## Epoch 27/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8004 - loss: 0.4364 - val_accuracy: 0.6423 - val_loss: 0.6790
## Epoch 28/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8000 - loss: 0.4344 - val_accuracy: 0.6520 - val_loss: 0.6755
## Epoch 29/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8018 - loss: 0.4334 - val_accuracy: 0.6497 - val_loss: 0.6805
## Epoch 30/100
## 9/9 - 0s - 21ms/step - accuracy: 0.8022 - loss: 0.4311 - val_accuracy: 0.6515 - val_loss: 0.6800
## Epoch 31/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8050 - loss: 0.4295 - val_accuracy: 0.6552 - val_loss: 0.6785
## Epoch 32/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8029 - loss: 0.4277 - val_accuracy: 0.6515 - val_loss: 0.6818
## Epoch 33/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8063 - loss: 0.4258 - val_accuracy: 0.6566 - val_loss: 0.6820
## Epoch 34/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8093 - loss: 0.4247 - val_accuracy: 0.6585 - val_loss: 0.6822
## Epoch 35/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8084 - loss: 0.4233 - val_accuracy: 0.6673 - val_loss: 0.6822
## Epoch 36/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8105 - loss: 0.4215 - val_accuracy: 0.6571 - val_loss: 0.6885
## Epoch 37/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8095 - loss: 0.4204 - val_accuracy: 0.6525 - val_loss: 0.6947
## Epoch 38/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8109 - loss: 0.4196 - val_accuracy: 0.6566 - val_loss: 0.6907
## Epoch 39/100
## 9/9 - 0s - 13ms/step - accuracy: 0.8105 - loss: 0.4175 - val_accuracy: 0.6571 - val_loss: 0.6940
## Epoch 40/100
## 9/9 - 0s - 17ms/step - accuracy: 0.8118 - loss: 0.4172 - val_accuracy: 0.6650 - val_loss: 0.6922
## Epoch 41/100
## 9/9 - 0s - 10ms/step - accuracy: 0.8125 - loss: 0.4153 - val_accuracy: 0.6613 - val_loss: 0.6977
## Epoch 42/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8127 - loss: 0.4142 - val_accuracy: 0.6571 - val_loss: 0.6991
## Epoch 43/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8134 - loss: 0.4133 - val_accuracy: 0.6650 - val_loss: 0.6988
## Epoch 44/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8139 - loss: 0.4118 - val_accuracy: 0.6627 - val_loss: 0.7018
## Epoch 45/100
```

```
## 9/9 - 0s - 8ms/step - accuracy: 0.8162 - loss: 0.4109 - val_accuracy: 0.6636 - val_loss: 0.7079
## Epoch 46/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8150 - loss: 0.4101 - val_accuracy: 0.6631 - val_loss: 0.7044
## Epoch 47/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8171 - loss: 0.4085 - val_accuracy: 0.6636 - val_loss: 0.7067
## Epoch 48/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8153 - loss: 0.4078 - val_accuracy: 0.6645 - val_loss: 0.7096
## Epoch 49/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8182 - loss: 0.4079 - val_accuracy: 0.6664 - val_loss: 0.7085
## Epoch 50/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8175 - loss: 0.4059 - val_accuracy: 0.6668 - val_loss: 0.7142
## Epoch 51/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8182 - loss: 0.4041 - val_accuracy: 0.6589 - val_loss: 0.7182
## Epoch 52/100
## 9/9 - 0s - 11ms/step - accuracy: 0.8182 - loss: 0.4033 - val_accuracy: 0.6645 - val_loss: 0.7158
## Epoch 53/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8171 - loss: 0.4033 - val_accuracy: 0.6696 - val_loss: 0.7151
## Epoch 54/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8173 - loss: 0.4022 - val_accuracy: 0.6650 - val_loss: 0.7224
## Epoch 55/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8171 - loss: 0.4009 - val_accuracy: 0.6576 - val_loss: 0.7270
## Epoch 56/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8178 - loss: 0.3998 - val_accuracy: 0.6645 - val_loss: 0.7215
## Epoch 57/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8201 - loss: 0.3995 - val_accuracy: 0.6631 - val_loss: 0.7290
## Epoch 58/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8196 - loss: 0.3983 - val_accuracy: 0.6659 - val_loss: 0.7314
## Epoch 59/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8210 - loss: 0.3962 - val_accuracy: 0.6617 - val_loss: 0.7379
## Epoch 60/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8182 - loss: 0.3959 - val_accuracy: 0.6668 - val_loss: 0.7380
## Epoch 61/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8230 - loss: 0.3950 - val_accuracy: 0.6650 - val_loss: 0.7357
## Epoch 62/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8232 - loss: 0.3950 - val_accuracy: 0.6557 - val_loss: 0.7477
## Epoch 63/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8235 - loss: 0.3934 - val_accuracy: 0.6585 - val_loss: 0.7494
## Epoch 64/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8216 - loss: 0.3925 - val_accuracy: 0.6617 - val_loss: 0.7510
## Epoch 65/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8255 - loss: 0.3922 - val_accuracy: 0.6608 - val_loss: 0.7500
## Epoch 66/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8280 - loss: 0.3903 - val_accuracy: 0.6576 - val_loss: 0.7561
## Epoch 67/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8251 - loss: 0.3910 - val_accuracy: 0.6571 - val_loss: 0.7569
## Epoch 68/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8235 - loss: 0.3897 - val_accuracy: 0.6650 - val_loss: 0.7549
## Epoch 69/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8253 - loss: 0.3887 - val_accuracy: 0.6594 - val_loss: 0.7632
## Epoch 70/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8258 - loss: 0.3886 - val_accuracy: 0.6585 - val_loss: 0.7607
## Epoch 71/100
## 9/9 - 0s - 6ms/step - accuracy: 0.8251 - loss: 0.3867 - val_accuracy: 0.6594 - val_loss: 0.7657
## Epoch 72/100
```

```
## 9/9 - 0s - 7ms/step - accuracy: 0.8251 - loss: 0.3861 - val_accuracy: 0.6576 - val_loss: 0.7721
## Epoch 73/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8255 - loss: 0.3871 - val_accuracy: 0.6589 - val_loss: 0.7708
## Epoch 74/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8280 - loss: 0.3856 - val_accuracy: 0.6566 - val_loss: 0.7736
## Epoch 75/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8278 - loss: 0.3837 - val_accuracy: 0.6552 - val_loss: 0.7791
## Epoch 76/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8276 - loss: 0.3846 - val_accuracy: 0.6599 - val_loss: 0.7716
## Epoch 77/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8292 - loss: 0.3830 - val_accuracy: 0.6525 - val_loss: 0.7805
## Epoch 78/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8296 - loss: 0.3818 - val_accuracy: 0.6538 - val_loss: 0.7806
## Epoch 79/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8292 - loss: 0.3813 - val_accuracy: 0.6529 - val_loss: 0.7873
## Epoch 80/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8292 - loss: 0.3798 - val_accuracy: 0.6520 - val_loss: 0.7857
## Epoch 81/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8299 - loss: 0.3807 - val_accuracy: 0.6497 - val_loss: 0.7969
## Epoch 82/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8326 - loss: 0.3789 - val_accuracy: 0.6529 - val_loss: 0.7976
## Epoch 83/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8274 - loss: 0.3781 - val_accuracy: 0.6506 - val_loss: 0.7973
## Epoch 84/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8306 - loss: 0.3778 - val_accuracy: 0.6497 - val_loss: 0.8034
## Epoch 85/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8296 - loss: 0.3770 - val_accuracy: 0.6525 - val_loss: 0.8058
## Epoch 86/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8331 - loss: 0.3761 - val_accuracy: 0.6562 - val_loss: 0.8016
## Epoch 87/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8303 - loss: 0.3754 - val_accuracy: 0.6483 - val_loss: 0.8140
## Epoch 88/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8315 - loss: 0.3742 - val_accuracy: 0.6525 - val_loss: 0.8130
## Epoch 89/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8338 - loss: 0.3747 - val_accuracy: 0.6538 - val_loss: 0.8146
## Epoch 90/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8351 - loss: 0.3736 - val_accuracy: 0.6557 - val_loss: 0.8167
## Epoch 91/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8351 - loss: 0.3729 - val_accuracy: 0.6478 - val_loss: 0.8212
## Epoch 92/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8383 - loss: 0.3729 - val_accuracy: 0.6520 - val_loss: 0.8171
## Epoch 93/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8374 - loss: 0.3714 - val_accuracy: 0.6520 - val_loss: 0.8211
## Epoch 94/100
## 9/9 - 0s - 7ms/step - accuracy: 0.8376 - loss: 0.3709 - val_accuracy: 0.6478 - val_loss: 0.8255
## Epoch 95/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8365 - loss: 0.3703 - val_accuracy: 0.6557 - val_loss: 0.8267
## Epoch 96/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8363 - loss: 0.3703 - val_accuracy: 0.6446 - val_loss: 0.8393
## Epoch 97/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8342 - loss: 0.3687 - val_accuracy: 0.6506 - val_loss: 0.8349
## Epoch 98/100
## 9/9 - 0s - 8ms/step - accuracy: 0.8385 - loss: 0.3688 - val_accuracy: 0.6478 - val_loss: 0.8373
## Epoch 99/100
```

```
## 9/9 - 0s - 9ms/step - accuracy: 0.8356 - loss: 0.3678 - val_accuracy: 0.6529 - val_loss: 0.8417
## Epoch 100/100
## 9/9 - 0s - 9ms/step - accuracy: 0.8390 - loss: 0.3676 - val_accuracy: 0.6511 - val_loss: 0.8378
```

```r
plot(history2)
```




```r
set.seed(42)
predictions2 <- predict(model2, test_features)
```

```
## 69/69 - 0s - 2ms/step
```

```r
head(predictions2, 10)
```

```
##            [,1]
##  [1,] 0.8905020
##  [2,] 0.8692546
##  [3,] 0.9195083
##  [4,] 0.7498678
##  [5,] 0.1221767
##  [6,] 0.6305349
##  [7,] 0.7876419
##  [8,] 0.7358160
##  [9,] 0.5480292
## [10,] 0.8271554
```

```r
predicted_class2 <- (predictions2[, 1] >= 0.5) * 1
head(predicted_class2, 10)
```

```
##  [1] 1 1 1 1 0 1 1 1 1 1
```

3) Compare the curves from 1) and 2) and discuss which architecture (i.e., number of nodes in the hidden layers) results in better performance.

Comparing the curves from 1) and 2), it appears that the architecture with less nodes in the hidden layers actually performs better. This is somewhat surprising to me, but after diving into possible explanations for this behavior, I believe it to be due to overfitting the data with more nodes in 2). One of the main reasons I believe overfitting to be an issue in 2) is because of the fact that the model continues to improve on the training data as the number of epochs increases. However, the model on the validation data appears to worsen, showing that it is not generalizing well to new data it hasn't "seen before". To resolve this, it may be worth decreasing the number of hidden layers in the model, or collecting additional data to help it generalize better.

4) Calculate the accuracy on the test set for the models in 1) and 2). Which accuracy is better?

```r
set.seed(42)
results1 <- model1 %>% evaluate(test_features, test_labels)
```

```
## 69/69 - 0s - 2ms/step - accuracy: 0.7480 - loss: 0.5464
```

```r
results1
```

```
## $accuracy
## [1] 0.7480496
##
## $loss
## [1] 0.5464011
```

```r
results2 <- model2 %>% evaluate(test_features, test_labels)
```

```
## 69/69 - 0s - 993us/step - accuracy: 0.7572 - loss: 0.5554
```

```r
results2
```

```
## $accuracy
## [1] 0.7572281
##
## $loss
## [1] 0.5554054
```

The accuracy for model2 used in 2) is better at .754 as opposed to model1 used in 1) at .747, although they are both very close together.