Preliminary Results_Simonsen

Steven Simonsen

2024-09-29

Below are the preliminary results for the binary classification problem where the label is to determine hotel bookings for ABC Hotels that are at a high risk of cancellation. The workflow for creating a densely connected feed-forward neural network for resolving this problem is outlined below.

First, I imported the data, and applied feature engineering to exclude the BookingID to eliminate noise in the dataset, as noted in my Analytic Plan. Next, I converted the positive class (class that I'm interested in determining), which is cancelled bookings to "1", and the non_cancelled class to "0". As noted in my Analytic plan, I was able to extract features from the arrival_date predictor, including both the year and the season. From there, I split the data into a training set (75%) and test set (25%), and created a recipe (booking_recipe) to apply one hot encoding to all categorical variables, center the variables, scale each variable, and prep the data on the training set to use going forward.

Next, the preparation to create a neural network was performed by converting the labels and features on both the training and test sets to arrays. After that, I was able to generate my models. I created five neural network models for this supervised classification problem, and each is outlined below:

Model 1 creation: I considered this model to be my baseline, and specified a model with three layers. I chose three layers because this seemed to be a number where I could appropriately include a sufficient number of units in the model, without adding too much complexity. The first dense layer, or hidden layer, consisted of 87 units, while the second hidden layer consisted of 42 units. I chose these numbers because the total number of features, or predictors, in the dataset was 29 after baking the data. I wanted to be sure to include enough units to capture the interactions and complexities within the data, so I chose three times the amount in the first layer to do this. In the second layer, I chose 42 as this is approximately half of the original 87 units. Finally, the output layer had 1 layer so an overall probability could be generated. Within the hidden layers, I chose RELU (Rectified Linear Unit) as my activation function for a few reasons. First, this activation function introduces non-linearity into the model, allowing to help learn complex patterns within the data. Second, this activation function is effective, yet simple because it involves simple thresholding at 0. For my output layer, I chose sigmoid as my activation function for two main reasons. First, this function outputs a value between 0 and 1, which is ideal because the objective of this problem is to determine a probability between 0 and 1 in which bookings are at high risk for cancellation. Second, sigmoid is ideal for binary classification problems such as this because it ouputs a value between 0 and 1, thus making for simple decision making between two classes.

Compiling Model 1: To compile model 1, and all models, I chose rmsprop as the optimizer. I chose this because rmsprop has an adaptive learning rate for each parameter, which helps in faster convergence and better performance. Additionally, this also helps in dealing with the sparse gradients and ensures more effective updates. I chose binary_crossentropy as my loss function because this particular algorithm measures the performance of a classification model whose output is a probability value between 0 and 1. Since our model uses sigmoid in our output layer, this was a good choice. Also, this loss function is differentiable, making it suitable for gradient-based optimization methods.

Additional model creation: Over the course of the code below, I made singular adjustments from the baseline to each sequential model. In model 2, I used more units in each hidden layer, with 174 in the first layer (six times the amount of predictors) and 87 in the second hidden layer. In model 3, I kept the original amount of units in the hidden layers, but increased the number of epochs from 250 to 750 as a means of iterating

through the training set more in an attempt to capture additional data intricacies. In model 4, I increased the number of hidden layers overall as another means of capturing patterns and complex relationships within the data. Finally, in model 5 I combined the increased number of hidden layers with the increased number of epochs to see if the two of these modifications combined would produce better results.

Evaluation of the models: To evaluate the various models, I created plots of each of the history variables to determine the accuracy and loss for the training data, and held out validation data. Looking at each of the accuracy/loss plots, a few general conclusions can be made. First, none of these models appear to underfit to the data. I can tell this is the case because underfitting typically comes with the appearance of at least a few of the following visual indicators: 1) High training loss curve 2) Low training accuracy 3) Similar training and validation performance. None of these are occurring. However, there is some overfitting happening. Typical signs of overfitting on the various training and validation plots include the following: 1) Low training loss with high validation loss 2) High training accuracy with low validation accuracy 3) Diverging training and validation curves. In the models I've created below, overfitting occurs and is illustrated by reasons 1) and 3) above. Model 5 seems to overfit the most, which is to be expected because it is the most complex model due to the highest number of epochs and highest number of hidden layers.

Next Steps: In the next phase of the project, I would like to dial in the various models below to find the optimal, or champion, result. This will provide me with the best means of predicting whether a booking is of high risk for cancellation for ABC Hotels. Achieving overfitting is a good thing, because now I can apply methods to reduce overfitting, while working to effectively generalize on new data (test set). Some of the methods I would like to use to do this in the final project phase include earlier stopping on the training data, simplification of the final model, or application of L1/L2 regularization. As an additional quick means of validation, I also generated ROC/AUC data on the held out test set. Model 2 produced the highest AUC (area under the curve) score, so it may be beneficial to take a closer look at the qualities that helped this model perform so well.

```
library(tidymodels)
```

```
## -- Attaching packages ------------------------------------ tidymodels 1.2.0 --
## v broom        1.0.6     v recipes      1.1.0
## v dials        1.3.0     v rsample      1.2.1
## v dplyr        1.1.4     v tibble       3.2.1
## v ggplot2      3.5.1     v tidyr        1.3.1
## v infer        1.0.7     v tune         1.2.1
## v modeldata    1.4.0     v workflows    1.1.4
## v parsnip      1.2.1     v workflowsets 1.1.0
## v purrr        1.0.2     v yardstick    1.3.1
## -- Conflicts --------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(reticulate)
library(tensorflow)
library(keras3)
```

```
##
## Attaching package: 'keras3'
## The following objects are masked from 'package:tensorflow':
##
##     set_random_seed, shape
```

```
## The following object is masked from 'package:yardstick':
##
##     get_weights
```

```
library(MESS)

getwd()
```

```
## [1] "C:/Users/steve/OneDrive/Documents/School/DSE6211/Project"
```

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

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

data <- data[, -1]
data$booking_status <- ifelse(data$booking_status == "canceled", 1, 0)

#Function creation to extract seasons from arrival date
data$arrival_date <- as.Date(data$arrival_date)
data$month <- format(data$arrival_date, "%m")
data$year <- format(data$arrival_date, "%Y")

get_season <- function(month) {
  month <- as.numeric(month)
  if (month %in% c(12, 1, 2)) {
    return("Winter")
  } else if (month %in% c(3, 4, 5)) {
    return("Spring")
  } else if (month %in% c(6, 7, 8)) {
    return("Summer")
  } else if (month %in% c(9, 10, 11)) {
    return("Autumn")
  }
}

data$season <- sapply(data$month, get_season)

data <- data[, -9]
data <- data[, -16]

set.seed(42)
data_split <- initial_split(data, strata = "booking_status", prop = 0.75)

training_set <- training(data_split)
test_set  <- testing(data_split)

booking_recipe <-
  recipe(
    booking_status ~ .,
    data = training_set
  ) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  prep(training = training_set, retain = TRUE)
```

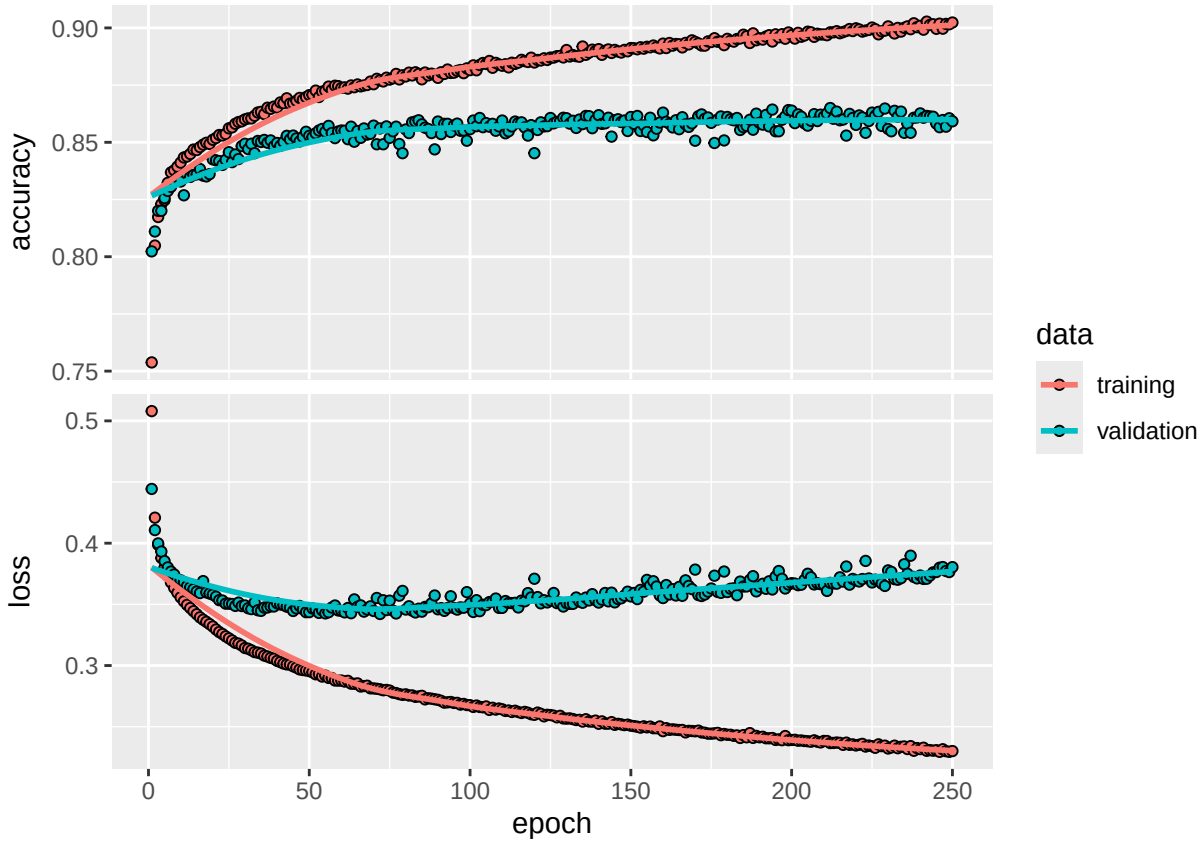
```
training_set_baked <- bake(booking_recipe, new_data = training_set)
test_set_baked <- bake(booking_recipe, new_data = test_set)



training_features <- array(data = unlist(training_set_baked[, -12]),
                           dim = c(nrow(training_set_baked), 28))
training_labels <- array(data = unlist(training_set_baked[, 12]),
                         dim = c(nrow(training_set_baked)))

test_features <- array(data = unlist(test_set_baked[, -12]),
                       dim = c(nrow(test_set_baked), 28))
test_labels <- array(data = unlist(test_set_baked[, 12]),
                     dim = c(nrow(test_set_baked)))

use_virtualenv("my_tf_workspace")
```

Model 1 - Baseline model

```
set.seed(42)
model1 <- keras_model_sequential() %>%
  layer_dense(units = 87, activation = "relu") %>%
  layer_dense(units = 42, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

set.seed(42)
compile(model1,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")

set.seed(42)
random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
```

```
set.seed(42)
history1 <- fit(model1, training_features, training_labels,
               epochs = 250, batch_size = 512, validation_split = 0.33)
```

```
plot(history1)
```

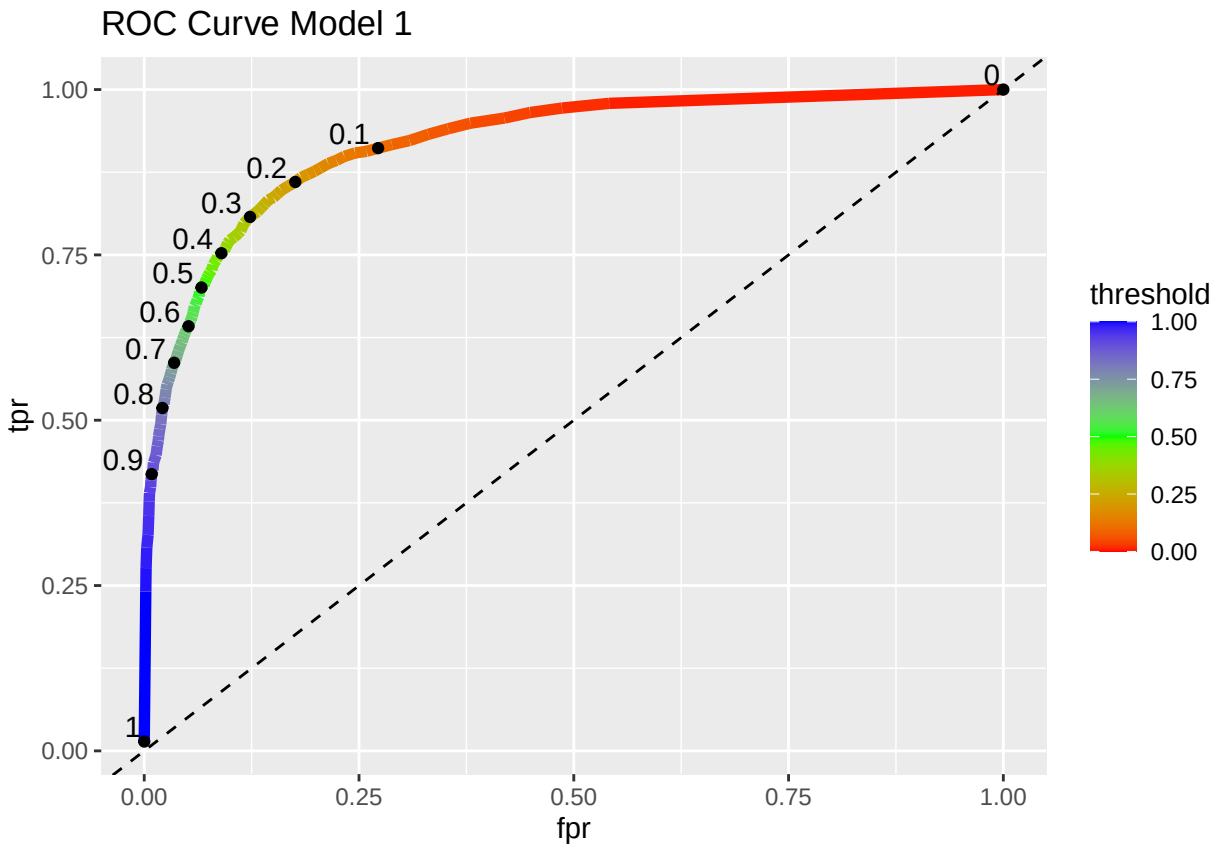



```
set.seed(42)
predictions1 <- predict(model1, test_features)
```

```
## 284/284 - 0s - 1ms/step
```

```
test_results1 <-
  test_set %>%
  select(booking_status) %>%
  bind_cols(
    data.frame(p_1 = predictions1)
  )

roc_data <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data$threshold) {

  over_threshold <- test_results1[test_results1$p_1 >= i, ]

  fpr <- sum(over_threshold$booking_status==0)/sum(test_results1$booking_status==0)
  roc_data[roc_data$threshold==i, "fpr"] <- fpr

  tpr <- sum(over_threshold$booking_status==1)/sum(test_results1$booking_status==1)
  roc_data[roc_data$threshold==i, "tpr"] <- tpr

}

ggplot() +
  geom_line(data = roc_data, aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +
```

```
  scale_color_gradientn(colors = rainbow(3)) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(data = roc_data[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +
  geom_text(data = roc_data[seq(1, 101, 10), ],
            aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2)) +
  labs(title = "ROC Curve Model 1")
```




```
auc1 <- auc(x = roc_data$fpr, y = roc_data$tpr, type = "spline")
auc1
```

```
## [1] 0.9210874
```

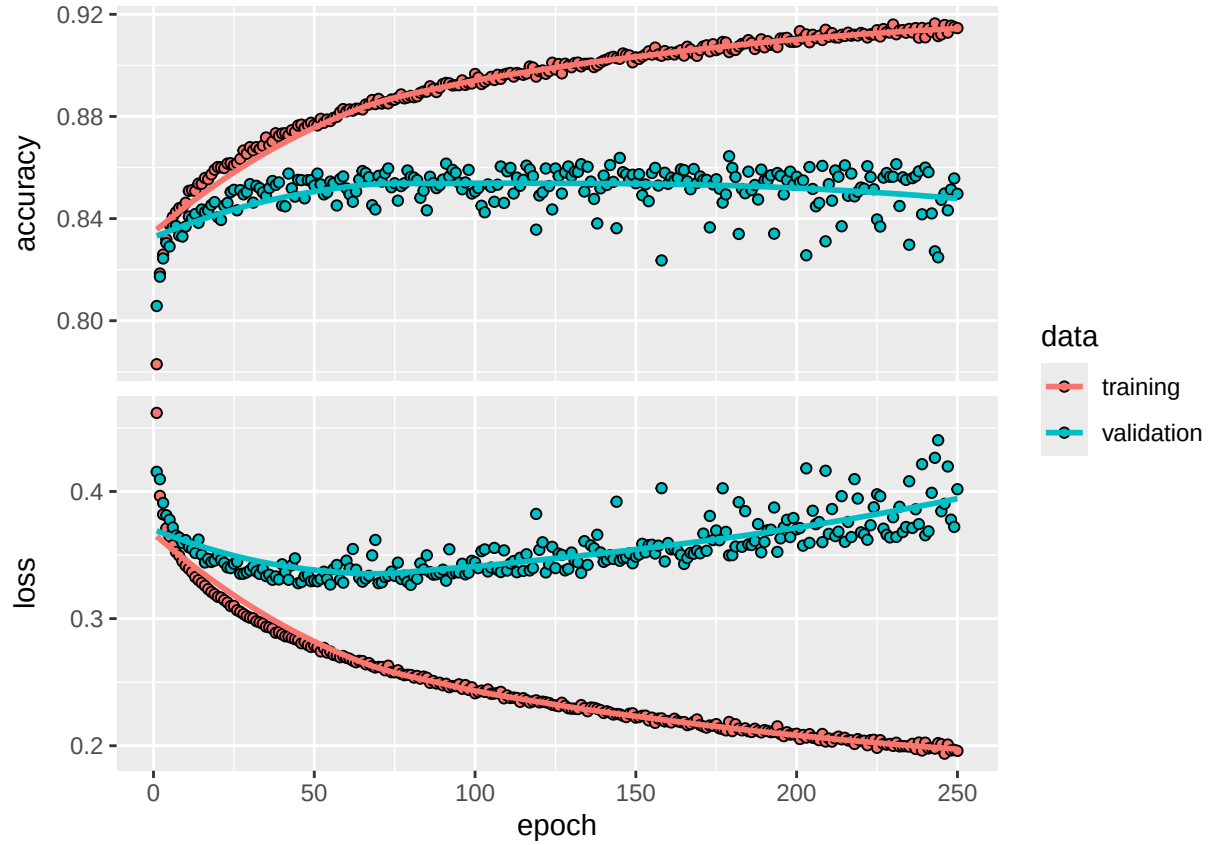
Model 2 - Additional units in hidden layer

```
set.seed(42)
model2 <- keras_model_sequential() %>%
  layer_dense(units = 174, activation = "relu") %>%
  layer_dense(units = 87, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

set.seed(42)
compile(model2,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")
```

```
set.seed(42)
random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
```

```
set.seed(42)
history2 <- fit(model2, training_features, training_labels,
                epochs = 250, batch_size = 512, validation_split = 0.33)
```

```
plot(history2)
```

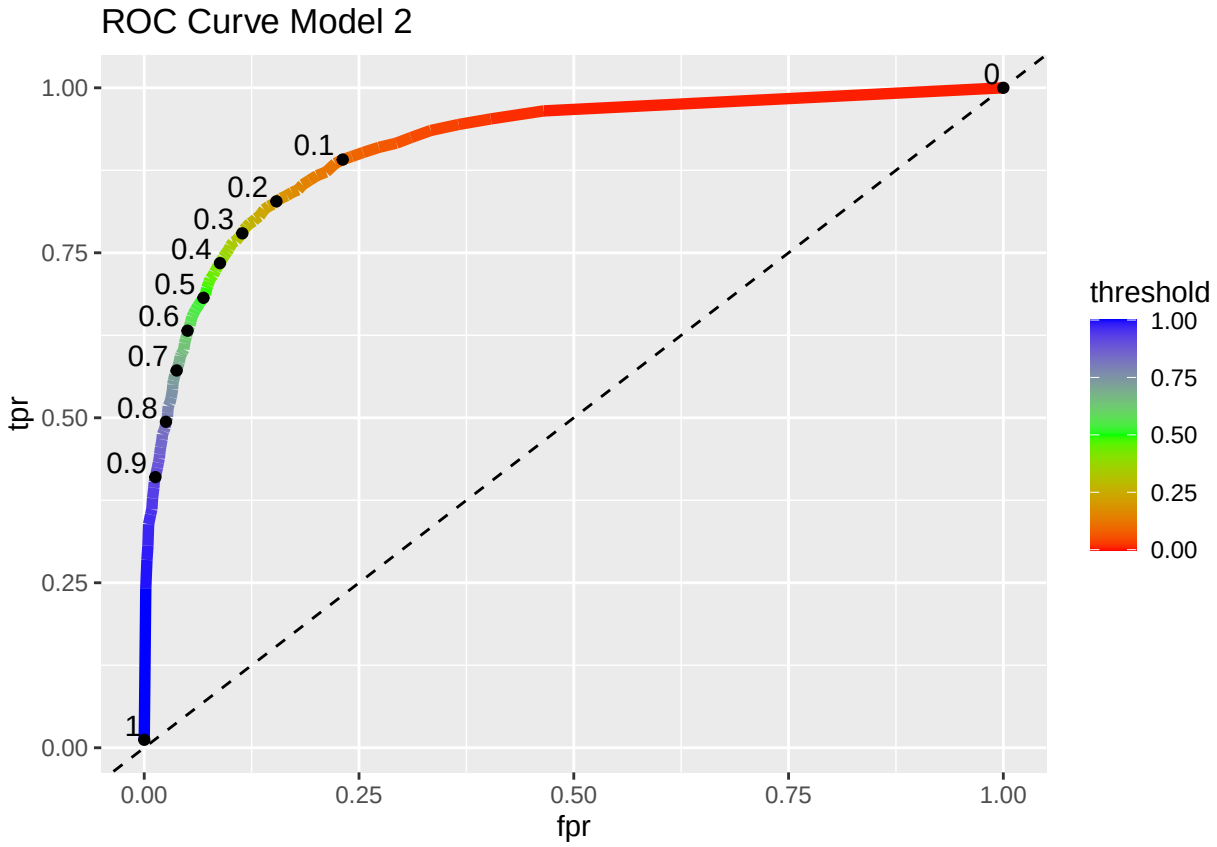



```
set.seed(42)
predictions2 <- predict(model2, test_features)
```

```
## 284/284 - 1s - 2ms/step
```

```
test_results2 <-
  test_set %>%
  select(booking_status) %>%
  bind_cols(
    data.frame(p_1 = predictions2)
  )

roc_data2 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data2$threshold) {

  over_threshold <- test_results2[test_results2$p_1 >= i, ]
```

```
  fpr <- sum(over_threshold$booking_status==0)/sum(test_results2$booking_status==0)
  roc_data2[roc_data2$threshold==i, "fpr"] <- fpr

  tpr <- sum(over_threshold$booking_status==1)/sum(test_results2$booking_status==1)
  roc_data2[roc_data2$threshold==i, "tpr"] <- tpr

}

ggplot() +
  geom_line(data = roc_data2, aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +
  scale_color_gradientn(colors = rainbow(3)) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(data = roc_data2[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +
  geom_text(data = roc_data2[seq(1, 101, 10), ],
            aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2)) +
  labs(title = "ROC Curve Model 2")
```




```
auc2 <- auc(x = roc_data2$fpr, y = roc_data2$tpr, type = "spline")
auc2
```

```
## [1] 0.9162546
```

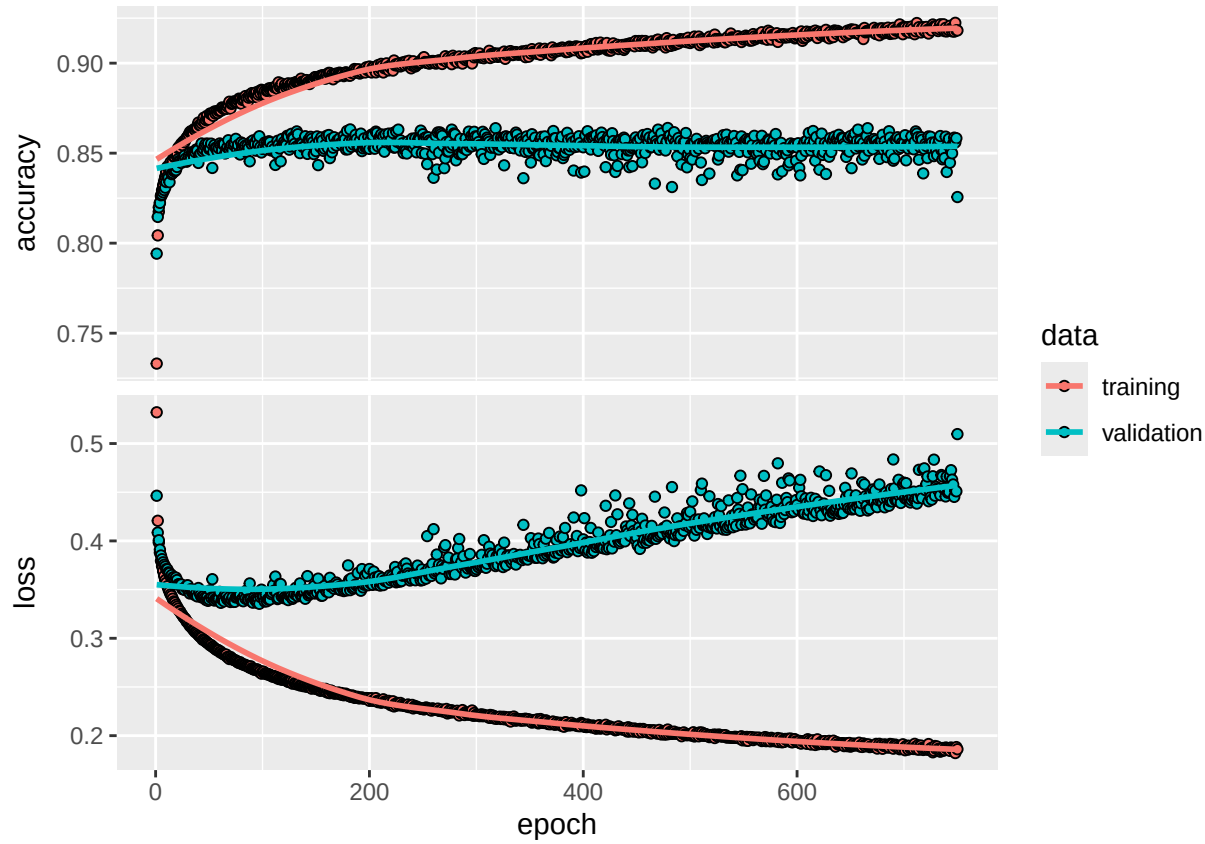
#Model 3 - More epochs, same units in hidden layer as original

```
set.seed(42)
model3 <- keras_model_sequential() %>%
  layer_dense(units = 87, activation = "relu") %>%
```

```
  layer_dense(units = 42, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

set.seed(42)
compile(model3,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")


set.seed(42)
random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
```

```
set.seed(42)
history3 <- fit(model3, training_features, training_labels,
                epochs = 750, batch_size = 512, validation_split = 0.33)
```

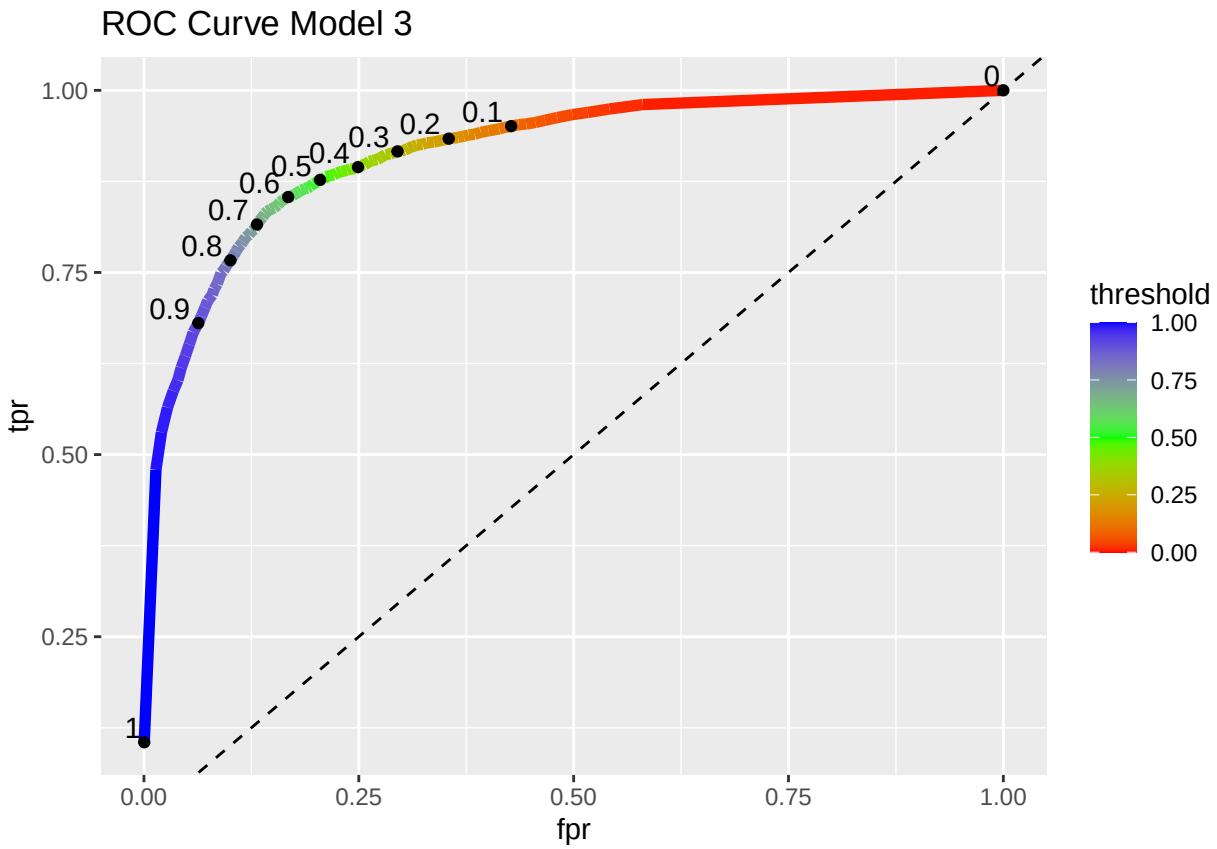
```
plot(history3)
```




```
set.seed(42)
predictions3 <- predict(model3, test_features)
```

```
## 284/284 - 0s - 752us/step
```

```r
test_results3 <-
  test_set %>%
  select(booking_status) %>%
  bind_cols(
    data.frame(p_1 = predictions3)
  )

roc_data3 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data3$threshold) {

  over_threshold <- test_results3[test_results3$p_1 >= i, ]

  fpr <- sum(over_threshold$booking_status==0)/sum(test_results3$booking_status==0)
  roc_data3[roc_data3$threshold==i, "fpr"] <- fpr

  tpr <- sum(over_threshold$booking_status==1)/sum(test_results3$booking_status==1)
  roc_data3[roc_data3$threshold==i, "tpr"] <- tpr

}

ggplot() +
  geom_line(data = roc_data3, aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +
  scale_color_gradientn(colors = rainbow(3)) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(data = roc_data3[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +
  geom_text(data = roc_data3[seq(1, 101, 10), ],
            aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2)) +
  labs(title = "ROC Curve Model 3")
```




```
auc3 <- auc(x = roc_data3$fpr, y = roc_data3$tpr, type = "spline")
auc3
```

```
## [1] 0.9177974
```

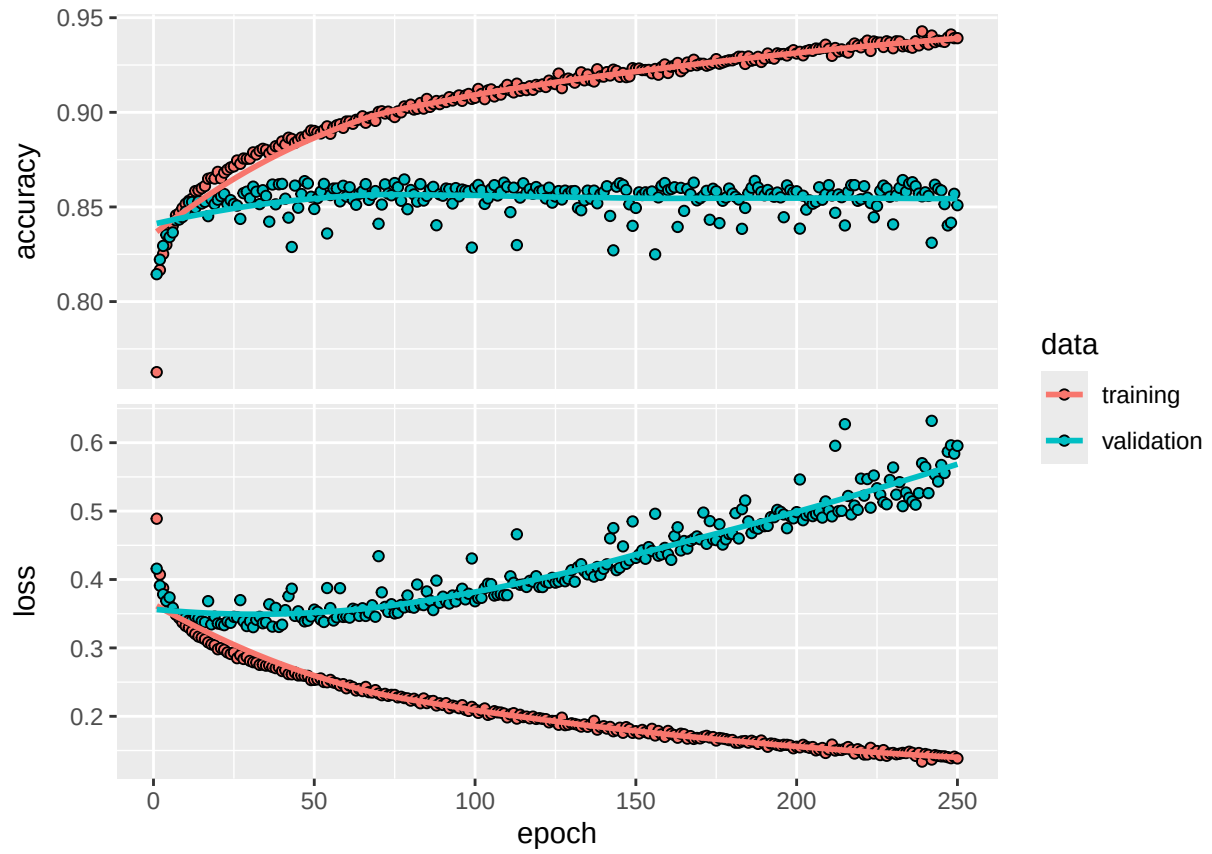
#Model 4 - Add more layers, 250 epochs, same # of units in hidden layer as original

```
set.seed(42)
model4 <- keras_model_sequential() %>%
  layer_dense(units = 87, activation = "relu") %>%
  layer_dense(units = 70, activation = "relu") %>%
  layer_dense(units = 53, activation = "relu") %>%
  layer_dense(units = 36, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

set.seed(42)
compile(model4,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")

set.seed(42)
random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
```

```
set.seed(42)
history4 <- fit(model4, training_features, training_labels,
```

```
                epochs = 250, batch_size = 512, validation_split = 0.33)
```

```
plot(history4)
```

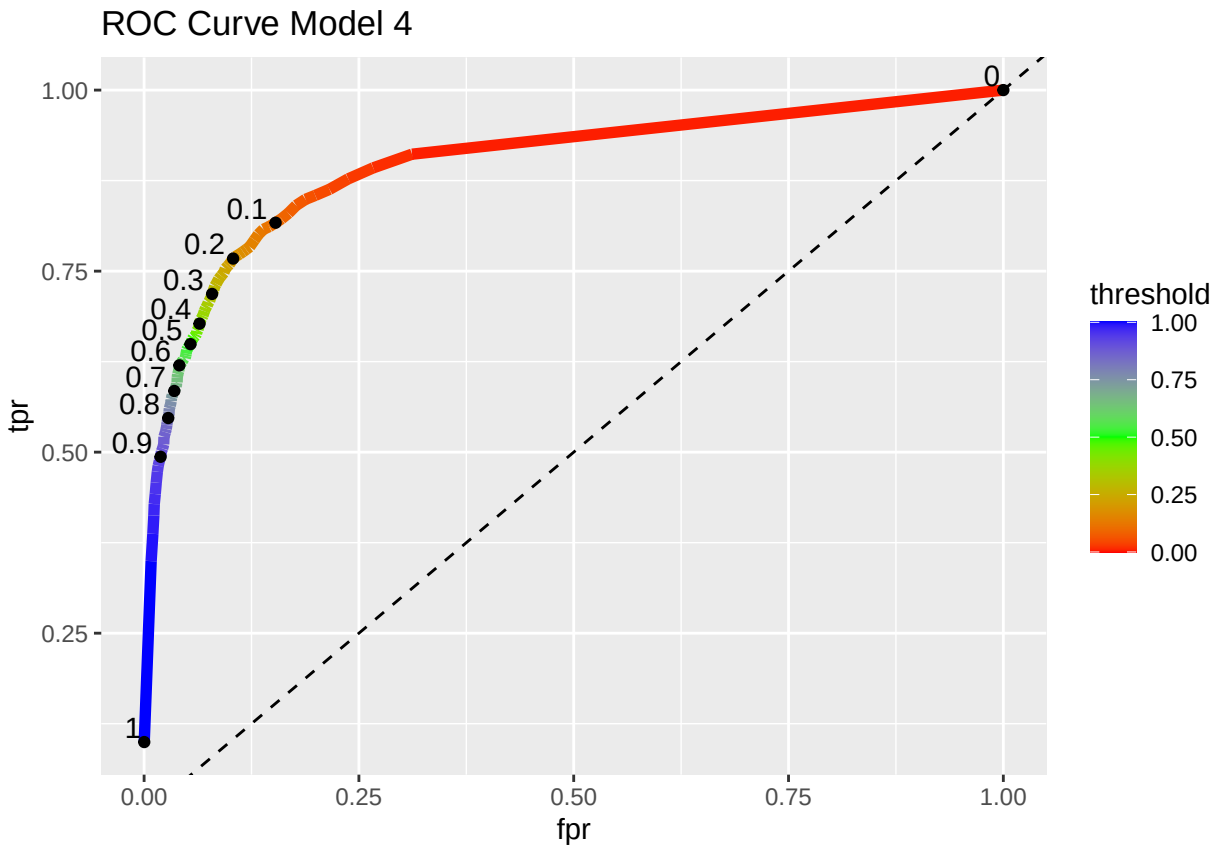



```
set.seed(42)
predictions4 <- predict(model4, test_features)
```

```
## 284/284 - 0s - 894us/step
```

```
test_results4 <-
  test_set %>%
  select(booking_status) %>%
  bind_cols(
    data.frame(p_1 = predictions4)
  )

roc_data4 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data4$threshold) {

  over_threshold <- test_results4[test_results4$p_1 >= i, ]

  fpr <- sum(over_threshold$booking_status==0)/sum(test_results4$booking_status==0)
  roc_data4[roc_data4$threshold==i, "fpr"] <- fpr

  tpr <- sum(over_threshold$booking_status==1)/sum(test_results4$booking_status==1)
  roc_data4[roc_data4$threshold==i, "tpr"] <- tpr
```

```
}

ggplot() +
  geom_line(data = roc_data4, aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +
  scale_color_gradientn(colors = rainbow(3)) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(data = roc_data4[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +
  geom_text(data = roc_data4[seq(1, 101, 10), ],
            aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2)) +
  labs(title = "ROC Curve Model 4")
```




```
auc4 <- auc(x = roc_data4$fpr, y = roc_data4$tpr, type = "spline")
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique
## 'x' values
```

```
auc4
```

```
## [1] 0.9131366
```

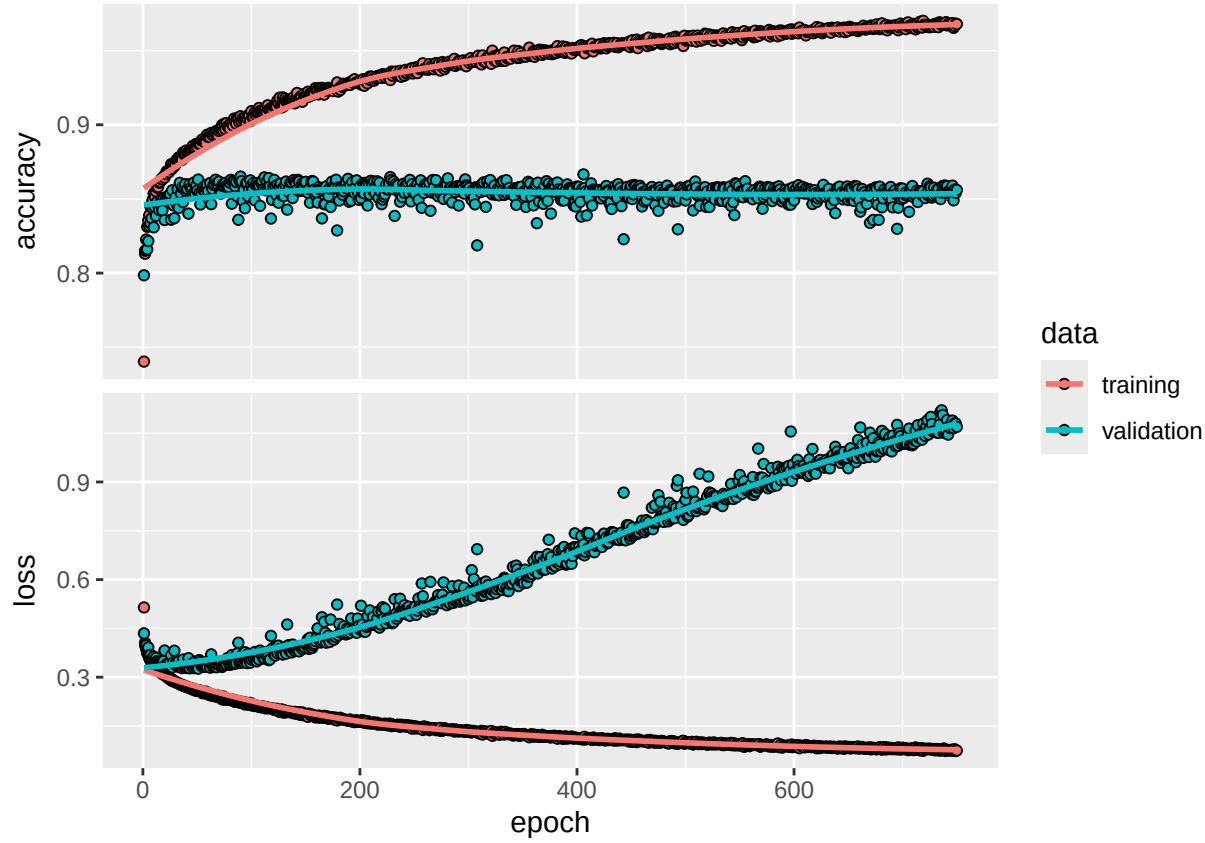
#Model 5 - Combining the two most meaningful models. More hidden layers, and 750 epochs. The goal is to overfit

```
set.seed(42)
model5 <- keras_model_sequential() %>%
  layer_dense(units = 87, activation = "relu") %>%
  layer_dense(units = 70, activation = "relu") %>%
  layer_dense(units = 53, activation = "relu") %>%
```

```
  layer_dense(units = 36, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

set.seed(42)
compile(model5,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")


set.seed(42)
random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
```

```
set.seed(42)
history5 <- fit(model5, training_features, training_labels,
                epochs = 750, batch_size = 512, validation_split = 0.33)
```

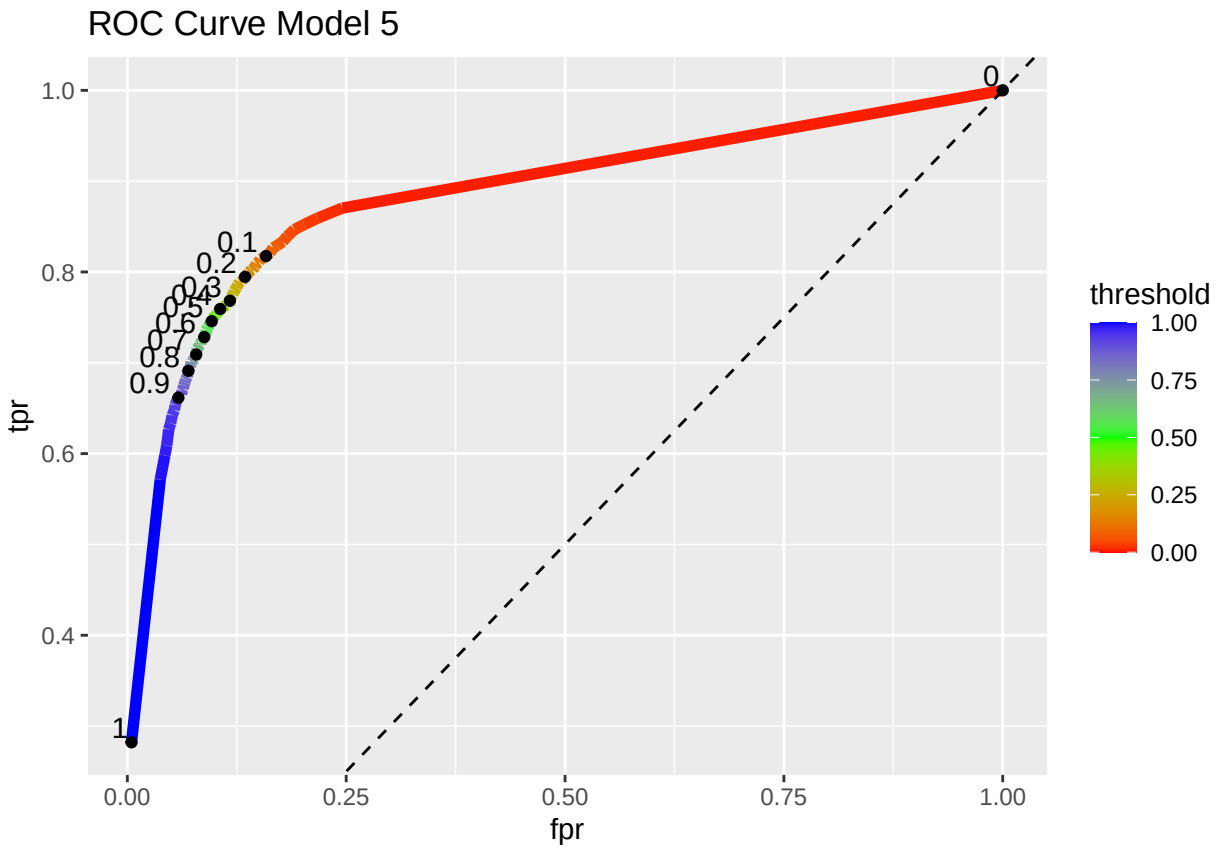
```
plot(history5)
```




```
set.seed(42)
predictions5 <- predict(model5, test_features)
```

```
## 284/284 - 1s - 2ms/step
```

```r
test_results5 <-
  test_set %>%
  select(booking_status) %>%
  bind_cols(
    data.frame(p_1 = predictions5)
  )

roc_data5 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data5$threshold) {

  over_threshold <- test_results5[test_results5$p_1 >= i, ]

  fpr <- sum(over_threshold$booking_status==0)/sum(test_results5$booking_status==0)
  roc_data5[roc_data5$threshold==i, "fpr"] <- fpr

  tpr <- sum(over_threshold$booking_status==1)/sum(test_results5$booking_status==1)
  roc_data5[roc_data5$threshold==i, "tpr"] <- tpr

}

ggplot() +
  geom_line(data = roc_data5, aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +
  scale_color_gradientn(colors = rainbow(3)) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(data = roc_data5[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +
  geom_text(data = roc_data5[seq(1, 101, 10), ],
            aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2)) +
  labs(title = "ROC Curve Model 5")
```




```
auc5 <- auc(x = roc_data5$fpr, y = roc_data5$tpr, type = "spline")
auc5
```

```
## [1] 0.8953037
```