DSE6211 Preliminary Results Appendix A

Ben Kelley

2024-06-10

```r
## building deep neural network model based on labs with information from
## project dataset

library(readr)
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
```

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.3.3
```

```r
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 4.3.3
```

```r
library(tensorflow)
```

```
## Warning: package 'tensorflow' was built under R version 4.3.3
```

```r
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.3.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.3.3
```

```r
use_virtualenv("my_tf_workspace", required = TRUE)

data <- read_csv("~/Data Science Masters Program/DSE6211/project_data.csv")
```

```
## Rows: 36238 Columns: 17
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): Booking_ID, type_of_meal_plan, room_type_reserved, market_segment...
## dbl  (11): no_of_adults, no_of_children, no_of_weekend_nights, no_of_week_ni...
## date  (1): arrival_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# adding a column to the data set which I will assign a 1 or 0 depending on
# customer status
# data[ , 'cancel'] <- NA
# I decided to simply replace the original column entries with 1 and 0 instead
# of creating a new column - I may change this later.

# using if else statement to assign a 1 or 0 to the 'booking_status' column
# denoting 1 for a cancelled booking and 0 for fulfilled booking.
data$booking_status <- ifelse(data$booking_status %in%
                                c('canceled'), 1, 0)

training_ind <- createDataPartition(data$booking_status,
                                    p = 0.75,
                                    list = FALSE,
                                    times = 1)
training_set <- data[training_ind, ]
test_set <- data[-training_ind, ]
unique(training_set$type_of_meal_plan)
```

```
## [1] "meal_plan_1"  "not_selected" "meal_plan_2"  "meal_plan_3"
```

```r
unique(training_set$room_type_reserved)
```

```
## [1] "room_type1" "room_type4" "room_type2" "room_type6" "room_type5"
## [6] "room_type7" "room_type3"
```

```r
unique(training_set$arrival_date) ## for now I am going to leave the arrival date out,
```

```
##   [1] "2017-10-02" "2018-02-28" "2018-04-11" "2018-09-13" "2017-10-15"
##   [6] "2018-12-26" "2018-07-06" "2018-10-18" "2018-09-11" "2018-04-30"
##  [11] "2018-11-26" "2017-10-20" "2018-06-15" "2017-10-05" "2017-08-10"
##  [16] "2017-10-30" "2018-11-25" "2018-03-20" "2018-10-13" "2018-05-22"
##  [21] "2018-04-28" "2018-05-19" "2017-11-06" "2017-09-17" "2017-09-19"
##  [26] "2018-12-07" "2018-10-07" "2018-04-27" "2017-10-17" "2018-11-19"
##  [31] "2018-07-30" "2018-11-01" "2018-06-21" "2018-04-10" "2018-06-27"
##  [36] "2017-11-18" "2017-12-29" "2018-04-06" "2018-12-29" "2018-05-30"
##  [41] "2018-04-22" "2018-07-28" "2018-04-20" "2018-09-02" "2018-05-03"
##  [46] "2018-06-24" "2017-09-10" "2018-12-18" "2018-10-05" "2018-06-03"
##  [51] "2018-08-25" "2018-11-14" "2017-10-25" "2018-03-09" "2018-12-31"
##  [56] "2018-06-28" "2018-03-29" "2018-04-01" "2018-12-25" "2018-03-14"
```

```
##  [61] "2018-03-04" "2018-10-17" "2018-12-10" "2018-08-09" "2018-03-18"
##  [66] "2018-04-09" "2018-08-30" "2018-08-04" "2018-04-13" "2018-12-22"
##  [71] "2018-10-23" "2018-05-13" "2018-12-13" "2017-10-31" "2017-10-08"
##  [76] "2017-09-21" "2018-06-30" "2017-09-05" "2018-09-30" "2018-06-16"
##  [81] "2018-06-26" "2018-08-13" "2018-11-12" "2017-11-14" "2018-09-01"
##  [86] "2018-09-16" "2018-03-10" "2018-05-05" "2018-04-24" "2018-03-30"
##  [91] "2018-11-03" "2018-10-14" "2018-06-05" "2018-10-20" "2018-12-28"
##  [96] "2017-10-19" "2017-10-09" "2018-04-15" "2018-05-09" "2018-11-18"
## [101] "2018-03-28" "2018-06-01" "2018-05-04" "2018-04-08" "2018-06-08"
## [106] "2018-04-07" "2018-05-02" "2018-04-04" "2017-07-17" "2018-05-24"
## [111] "2018-02-17" "2018-04-14" "2018-04-17" "2018-08-01" "2018-07-12"
## [116] "2018-11-16" "2017-09-13" "2018-12-16" "2018-11-05" "2018-03-22"
## [121] "2018-08-17" "2018-05-20" "2017-10-24" "2018-07-21" "2018-03-31"
## [126] "2018-10-11" "2018-04-02" "2018-08-22" "2018-08-08" "2018-08-19"
## [131] "2018-11-06" "2018-10-10" "2018-10-06" "2018-08-03" "2017-09-11"
## [136] "2018-09-26" "2018-08-24" "2018-08-21" "2018-12-03" "2018-02-27"
## [141] "2017-09-22" "2018-12-05" "2018-09-05" "2018-05-27" "2018-02-05"
## [146] "2017-10-18" "2017-08-14" "2017-09-03" "2018-09-14" "2018-02-08"
## [151] "2018-05-31" "2018-08-15" "2017-11-09" "2018-07-26" "2018-09-07"
## [156] "2017-11-15" "2017-10-16" "2018-07-23" "2018-06-11" "2018-01-16"
## [161] "2018-10-15" "2017-08-12" "2017-08-23" "2018-08-28" "2018-03-25"
## [166] "2018-12-09" "2018-06-12" "2018-03-05" "2017-08-18" "2018-04-21"
## [171] "2018-06-18" "2018-08-10" "2018-10-28" "2018-03-17" "2018-05-01"
## [176] "2018-04-03" "2017-09-04" "2017-12-27" "2018-01-02" "2018-05-16"
## [181] "2018-03-23" "2017-09-30" "2018-07-25" "2017-12-26" "2018-07-09"
## [186] "2018-06-06" "2018-08-18" "2018-02-22" "2018-06-17" "2018-07-08"
## [191] "2017-10-06" "2017-08-11" "2017-07-25" "2017-12-15" "2017-10-23"
## [196] "2017-10-10" "2018-10-02" "2018-12-21" "2018-11-10" "2018-02-24"
## [201] "2018-07-01" "2018-07-05" "2018-07-11" "2018-10-22" "2018-06-13"
## [206] "2018-09-22" "2018-07-10" "2017-09-25" "2018-10-04" "2018-10-27"
## [211] "2018-03-03" "2018-12-02" "2018-09-18" "2018-06-19" "2018-05-14"
## [216] "2017-09-15" "2018-02-19" "2018-09-28" "2018-08-12" "2018-09-09"
## [221] "2018-01-29" "2018-08-06" "2018-01-20" "2017-12-07" "2018-04-05"
## [226] "2018-07-02" "2018-09-27" "2018-03-24" "2018-09-03" "2017-10-21"
## [231] "2017-09-08" "2018-02-16" "2018-09-06" "2018-10-24" "2018-05-11"
## [236] "2017-10-13" "2018-10-16" "2018-10-31" "2018-06-23" "2018-12-08"
## [241] "2017-08-08" "2017-12-09" "2017-10-22" "2018-03-08" "2018-02-11"
## [246] "2018-11-04" "2018-11-23" "2018-06-07" "2018-10-21" "2018-10-30"
## [251] "2017-12-24" "2018-06-14" "2018-06-10" "2017-12-17" "2017-10-07"
## [256] "2018-04-23" "2018-11-02" "2018-03-19" "2018-02-03" "2018-03-16"
## [261] "2018-10-19" "2017-12-30" "2017-07-01" "2017-09-06" "2018-12-30"
## [266] "2018-05-12" "2018-01-06" "2018-03-13" "2018-06-29" "2018-02-26"
## [271] "2017-10-01" "2018-07-27" "2017-09-07" "2018-05-29" "2017-08-25"
## [276] "2017-10-14" "2017-09-09" "2018-10-01" "2017-09-18" "2018-02-06"
## [281] "2018-01-08" "2018-03-07" "2018-09-20" "2018-01-28" "2018-02-25"
## [286] "2018-11-17" "2018-12-14" "2018-07-13" "2017-08-09" "2018-02-14"
## [291] "2017-10-04" "2018-02-13" "2018-04-16" "2018-08-14" "2017-12-01"
## [296] "2018-09-21" "2018-05-06" "2018-06-02" "2018-04-12" "2018-02-15"
## [301] "2018-12-27" "2018-05-25" "2018-04-29" "2018-10-03" "2018-06-04"
## [306] "2017-12-06" "2018-07-19" "2018-07-14" "2017-12-11" "2017-07-11"
## [311] "2018-01-04" "2018-09-12" "2017-08-20" "2017-08-19" "2017-10-28"
## [316] "2018-10-12" "2018-05-08" "2018-09-15" "2018-02-12" "2018-07-18"
## [321] "2017-09-16" "2017-09-24" "2018-01-15" "2017-09-14" "2018-09-08"
## [326] "2018-05-10" "2017-08-27" "2018-01-14" "2018-09-23" "2018-08-20"
```

```
## [331] "2017-08-05" "2018-05-21" "2018-09-10" "2017-08-30" "2018-03-02"
## [336] "2018-06-22" "2017-12-04" "2017-08-03" "2018-11-07" "2018-07-03"
## [341] "2018-12-06" "2018-02-01" "2018-08-02" "2018-12-12" "2018-11-11"
## [346] "2017-09-01" "2017-12-10" "2018-01-27" "2018-12-23" "2018-06-20"
## [351] "2018-07-16" "2017-07-05" "2018-08-16" "2018-10-26" "2018-05-07"
## [356] "2018-01-25" "2018-07-15" "2017-11-20" "2018-01-24" "2018-03-27"
## [361] "2017-08-17" "2017-08-21" "2017-07-06" "2018-11-08" "2018-07-22"
## [366] "2017-11-11" "2018-12-24" "2018-05-18" "2018-08-27" "2018-10-09"
## [371] "2018-10-29" "2018-03-06" "2018-09-17" "2018-09-29" "2018-01-19"
## [376] "2018-11-15" "2018-11-09" "2018-05-17" "2018-08-29" "2018-08-31"
## [381] "2018-10-08" "2018-03-26" "2018-01-30" "2017-11-08" "2018-04-18"
## [386] "2018-03-01" "2017-12-31" "2018-07-29" "2017-11-28" "2018-12-04"
## [391] "2018-04-19" "2017-08-28" "2018-09-25" "2017-12-18" "2018-03-21"
## [396] "2018-10-25" "2017-09-12" "2018-11-13" "2018-09-04" "2018-05-26"
## [401] "2017-07-07" "2018-01-07" "2018-03-11" "2017-09-20" "2017-08-22"
## [406] "2017-09-28" "2017-08-29" "2018-08-26" "2018-03-12" "2018-02-09"
## [411] "2017-11-03" "2017-12-22" "2018-11-20" "2018-07-04" "2017-09-29"
## [416] "2018-11-22" "2018-08-07" "2017-10-29" "2018-12-20" "2017-08-01"
## [421] "2017-12-05" "2017-11-01" "2017-12-19" "2017-12-20" "2018-05-23"
## [426] "2017-11-04" "2017-07-23" "2017-11-13" "2018-07-20" "2018-07-31"
## [431] "2018-07-07" "2018-12-11" "2018-05-15" "2018-12-17" "2017-11-02"
## [436] "2018-01-31" "2018-02-23" "2018-12-01" "2017-11-10" "2018-01-10"
## [441] "2018-01-03" "2017-10-12" "2018-07-17" "2017-08-24" "2018-02-10"
## [446] "2018-11-24" "2018-05-28" "2018-03-15" "2018-09-24" "2018-06-09"
## [451] "2018-07-24" "2018-01-22" "2018-09-19" "2017-11-27" "2018-04-25"
## [456] "2018-01-05" "2018-01-13" "2017-11-05" "2018-11-28" "2018-01-26"
## [461] "2017-11-25" "2017-08-06" "2017-12-16" "2018-01-21" "2018-02-20"
## [466] "2018-11-21" "2018-12-19" "2017-07-18" "2018-02-21" "2018-12-15"
## [471] "2017-11-12" "2018-08-05" "2018-08-23" "2017-07-16" "2018-11-30"
## [476] "2017-09-02" "2018-02-04" "2017-08-04" "2017-10-11" "2018-08-11"
## [481] "2018-02-07" "2017-08-07" "2018-01-18" "2017-11-07" "2018-01-12"
## [486] "2017-11-19" "2018-11-27" "2017-07-29" "2017-08-31" "2017-08-15"
## [491] "2017-07-13" "2017-07-15" "2017-11-26" "2017-08-26" "2017-09-23"
## [496] "2017-10-26" "2018-11-29" "2017-07-09" "2018-02-18" "2017-10-03"
## [501] "2017-07-27" "2017-11-21" "2017-11-24" "2018-04-26" "2017-10-27"
## [506] "2017-12-03" "2017-08-13" "2018-01-23" "2018-01-01" "2017-12-25"
## [511] "2017-11-30" "2017-12-02" "2017-12-28" "2017-12-23" "2017-12-12"
## [516] "2017-09-27" "2017-11-22" "2018-06-25" "2017-07-31" "2018-01-17"
## [521] "2017-11-16" "2018-01-09" "2017-11-17" "2017-07-10" "2017-12-14"
## [526] "2017-09-26" "2017-11-23" "2017-07-02" "2018-01-11" "2017-08-16"
## [531] "2017-07-30" "2017-12-21" "2018-02-02" "2017-07-20" "2017-07-04"
## [536] "2017-07-22" "2017-07-28" "2017-12-13" "2017-07-08" "2017-07-24"
## [541] "2017-11-29" "2017-07-03" "2017-12-08" "2017-07-14" "2017-07-19"
## [546] "2017-08-02" "2017-07-12" "2017-07-26"
```

```r
# I may sort this by month instead of exact date for simplicity.
unique(training_set$market_segment_type)
```

```
## [1] "offline"       "online"        "corporate"     "aviation"
## [5] "complementary"
```

```r
top_20_dates <- training_set %>%
  group_by(arrival_date) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
```

```r
  select(arrival_date) %>%
  top_n(20)
```

```
## Selecting by arrival_date
```

```r
training_set$arrival_date <- ifelse(training_set$arrival_date %in% top_20_dates$arrival_date,
                                    training_set$arrival_date,
                                    "other")

training_set$type_of_meal_plan <- factor(training_set$type_of_meal_plan)
training_set$room_type_reserved <- factor(training_set$room_type_reserved)
training_set$arrival_date <- factor(training_set$arrival_date)
training_set$market_segment_type <- factor(training_set$market_segment_type)

class(training_set$type_of_meal_plan)
```

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

```r
class(training_set$room_type_reserved)
```

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

```r
class(training_set$arrival_date)
```

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

```r
class(training_set$market_segment_type)
```

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

```r
levels(training_set$type_of_meal_plan)
```

```
## [1] "meal_plan_1"  "meal_plan_2"  "meal_plan_3"  "not_selected"
```

```r
levels(training_set$room_type_reserved)
```

```
## [1] "room_type1" "room_type2" "room_type3" "room_type4" "room_type5"
## [6] "room_type6" "room_type7"
```

```r
levels(training_set$arrival_date)
```

```
##  [1] "17877" "17878" "17879" "17880" "17881" "17882" "17883" "17884" "17885"
## [10] "17886" "17887" "17888" "17889" "17890" "17891" "17892" "17893" "17894"
## [19] "17895" "17896" "other"
```

```r
levels(training_set$market_segment_type)
```

```
## [1] "aviation"      "complementary" "corporate"     "offline"
## [5] "online"
```

```r
onehot_encoder <- dummyVars(~ type_of_meal_plan + room_type_reserved +
                            arrival_date + market_segment_type,
                            training_set[, c("type_of_meal_plan",
                                             "room_type_reserved",
                                             "arrival_date",
                                             "market_segment_type")],
                            levelsOnly = TRUE,
                            fullRank = TRUE)

onehot_enc_training <- predict(onehot_encoder,
```

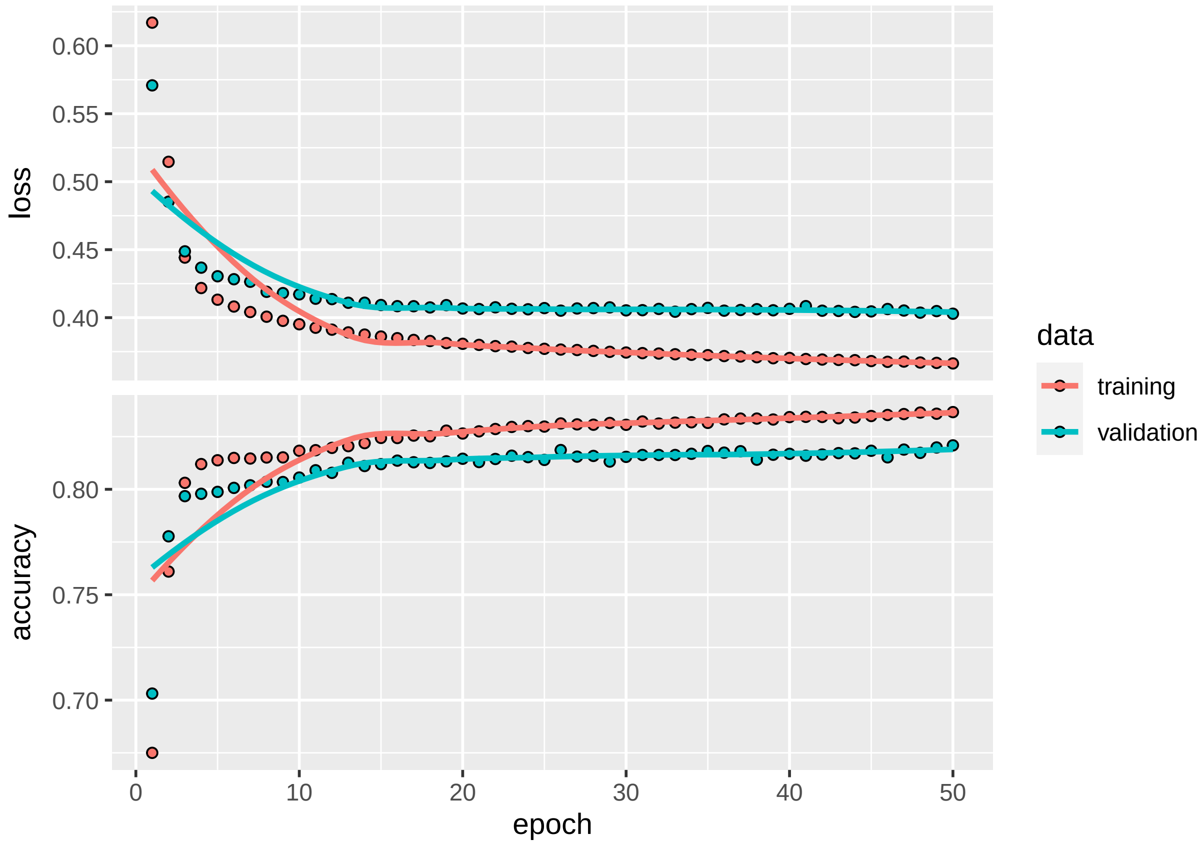
```r
                                training_set[, c("type_of_meal_plan",
                                                 "room_type_reserved",
                                                 "arrival_date",
                                                 "market_segment_type")])

training_set <- cbind(training_set, onehot_enc_training)

test_set$arrival_date <- ifelse(test_set$arrival_date %in%
                              top_20_dates$arrival_date,
                              test_set$arrival_date,
                              "other")

test_set$type_of_meal_plan <- factor(test_set$type_of_meal_plan)
test_set$room_type_reserved <- factor(test_set$room_type_reserved)
test_set$arrival_date <- factor(test_set$arrival_date)
test_set$market_segment_type <- factor(test_set$market_segment_type)

onehot_enc_test <- predict(onehot_encoder, test_set[, c("type_of_meal_plan",
                                                         "room_type_reserved",
                                                         "arrival_date",
                                                         "market_segment_type")])
test_set <- cbind(test_set, onehot_enc_test)

test_set[,-c(1, 6, 8, 10, 11, 17)] <- scale(test_set[,-c(1, 6, 8, 10, 11, 17)],
                               center = apply(training_set[,-c(1, 6, 8, 10, 11, 17)], 2, mean),
                               scale = apply(training_set[,-c(1, 6, 8, 10, 11, 17)], 2, sd))
training_set[,-c(1, 6, 8, 10, 11, 17)] <- scale(training_set[,-c(1, 6, 8, 10, 11, 17)])

training_features <- array(data = unlist(training_set[,-c(1, 6, 8, 10, 11, 17)]),
                           dim = c(nrow(training_set), 44))
training_labels <- array(data = unlist(training_set[, 17]),
                         dim = c(nrow(training_set)))
test_features <- array(data = unlist(test_set[,-c(1, 6, 8, 10, 11, 17)]),
                       dim = c(nrow(test_set), 44))
test_labels <- array(data = unlist(test_set[, 17]),
                     dim = c(nrow(test_set)))


model <- keras_model_sequential(list(
  layer_dense(units = 10, activation = "relu"),
  layer_dense(units = 10, activation = "relu"),
  layer_dense(units = 1, activation = "sigmoid")
))

compile(model,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")
history <- fit(model, training_features, training_labels,
               epochs = 50, batch_size = 128, validation_split = 0.33)
```

```
## Epoch 1/50
## 143/143 - 1s - loss: 0.6170 - accuracy: 0.6750 - val_loss: 0.5709 - val_accuracy: 0.7031 - 673ms/epo
## Epoch 2/50
```

```
## 143/143 - 0s - loss: 0.5146 - accuracy: 0.7610 - val_loss: 0.4853 - val_accuracy: 0.7777 - 182ms/epoc
## Epoch 3/50
## 143/143 - 0s - loss: 0.4442 - accuracy: 0.8031 - val_loss: 0.4487 - val_accuracy: 0.7968 - 167ms/epoc
## Epoch 4/50
## 143/143 - 0s - loss: 0.4218 - accuracy: 0.8120 - val_loss: 0.4368 - val_accuracy: 0.7979 - 168ms/epoc
## Epoch 5/50
## 143/143 - 0s - loss: 0.4132 - accuracy: 0.8138 - val_loss: 0.4304 - val_accuracy: 0.7988 - 190ms/epoc
## Epoch 6/50
## 143/143 - 0s - loss: 0.4082 - accuracy: 0.8149 - val_loss: 0.4283 - val_accuracy: 0.8007 - 167ms/epoc
## Epoch 7/50
## 143/143 - 0s - loss: 0.4041 - accuracy: 0.8146 - val_loss: 0.4264 - val_accuracy: 0.8019 - 168ms/epoc
## Epoch 8/50
## 143/143 - 0s - loss: 0.4007 - accuracy: 0.8151 - val_loss: 0.4190 - val_accuracy: 0.8036 - 164ms/epoc
## Epoch 9/50
## 143/143 - 0s - loss: 0.3976 - accuracy: 0.8151 - val_loss: 0.4180 - val_accuracy: 0.8035 - 167ms/epoc
## Epoch 10/50
## 143/143 - 0s - loss: 0.3951 - accuracy: 0.8183 - val_loss: 0.4171 - val_accuracy: 0.8056 - 163ms/epoc
## Epoch 11/50
## 143/143 - 0s - loss: 0.3926 - accuracy: 0.8186 - val_loss: 0.4139 - val_accuracy: 0.8090 - 167ms/epoc
## Epoch 12/50
## 143/143 - 0s - loss: 0.3911 - accuracy: 0.8196 - val_loss: 0.4136 - val_accuracy: 0.8078 - 166ms/epoc
## Epoch 13/50
## 143/143 - 0s - loss: 0.3891 - accuracy: 0.8205 - val_loss: 0.4109 - val_accuracy: 0.8126 - 182ms/epoc
## Epoch 14/50
## 143/143 - 0s - loss: 0.3875 - accuracy: 0.8219 - val_loss: 0.4110 - val_accuracy: 0.8110 - 166ms/epoc
## Epoch 15/50
## 143/143 - 0s - loss: 0.3861 - accuracy: 0.8244 - val_loss: 0.4093 - val_accuracy: 0.8119 - 166ms/epoc
## Epoch 16/50
## 143/143 - 0s - loss: 0.3849 - accuracy: 0.8243 - val_loss: 0.4084 - val_accuracy: 0.8136 - 167ms/epoc
## Epoch 17/50
## 143/143 - 0s - loss: 0.3836 - accuracy: 0.8255 - val_loss: 0.4084 - val_accuracy: 0.8128 - 166ms/epoc
## Epoch 18/50
## 143/143 - 0s - loss: 0.3828 - accuracy: 0.8251 - val_loss: 0.4075 - val_accuracy: 0.8125 - 167ms/epoc
## Epoch 19/50
## 143/143 - 0s - loss: 0.3813 - accuracy: 0.8278 - val_loss: 0.4092 - val_accuracy: 0.8133 - 168ms/epoc
## Epoch 20/50
## 143/143 - 0s - loss: 0.3807 - accuracy: 0.8265 - val_loss: 0.4067 - val_accuracy: 0.8145 - 167ms/epoc
## Epoch 21/50
## 143/143 - 0s - loss: 0.3799 - accuracy: 0.8275 - val_loss: 0.4063 - val_accuracy: 0.8129 - 182ms/epoc
## Epoch 22/50
## 143/143 - 0s - loss: 0.3790 - accuracy: 0.8286 - val_loss: 0.4076 - val_accuracy: 0.8144 - 167ms/epoc
## Epoch 23/50
## 143/143 - 0s - loss: 0.3787 - accuracy: 0.8296 - val_loss: 0.4065 - val_accuracy: 0.8159 - 167ms/epoc
## Epoch 24/50
## 143/143 - 0s - loss: 0.3777 - accuracy: 0.8300 - val_loss: 0.4062 - val_accuracy: 0.8153 - 168ms/epoc
## Epoch 25/50
## 143/143 - 0s - loss: 0.3771 - accuracy: 0.8298 - val_loss: 0.4070 - val_accuracy: 0.8139 - 165ms/epoc
## Epoch 26/50
## 143/143 - 0s - loss: 0.3765 - accuracy: 0.8312 - val_loss: 0.4051 - val_accuracy: 0.8186 - 166ms/epoc
## Epoch 27/50
## 143/143 - 0s - loss: 0.3762 - accuracy: 0.8308 - val_loss: 0.4068 - val_accuracy: 0.8155 - 168ms/epoc
## Epoch 28/50
## 143/143 - 0s - loss: 0.3755 - accuracy: 0.8306 - val_loss: 0.4070 - val_accuracy: 0.8158 - 168ms/epoc
## Epoch 29/50
```

```
## 143/143 - 0s - loss: 0.3748 - accuracy: 0.8315 - val_loss: 0.4075 - val_accuracy: 0.8132 - 181ms/epoc
## Epoch 30/50
## 143/143 - 0s - loss: 0.3743 - accuracy: 0.8306 - val_loss: 0.4053 - val_accuracy: 0.8154 - 165ms/epoc
## Epoch 31/50
## 143/143 - 0s - loss: 0.3739 - accuracy: 0.8322 - val_loss: 0.4055 - val_accuracy: 0.8163 - 166ms/epoc
## Epoch 32/50
## 143/143 - 0s - loss: 0.3736 - accuracy: 0.8312 - val_loss: 0.4064 - val_accuracy: 0.8163 - 164ms/epoc
## Epoch 33/50
## 143/143 - 0s - loss: 0.3731 - accuracy: 0.8316 - val_loss: 0.4043 - val_accuracy: 0.8163 - 164ms/epoc
## Epoch 34/50
## 143/143 - 0s - loss: 0.3727 - accuracy: 0.8318 - val_loss: 0.4063 - val_accuracy: 0.8168 - 166ms/epoc
## Epoch 35/50
## 143/143 - 0s - loss: 0.3724 - accuracy: 0.8315 - val_loss: 0.4072 - val_accuracy: 0.8182 - 165ms/epoc
## Epoch 36/50
## 143/143 - 0s - loss: 0.3717 - accuracy: 0.8332 - val_loss: 0.4051 - val_accuracy: 0.8174 - 165ms/epoc
## Epoch 37/50
## 143/143 - 0s - loss: 0.3714 - accuracy: 0.8335 - val_loss: 0.4056 - val_accuracy: 0.8181 - 182ms/epoc
## Epoch 38/50
## 143/143 - 0s - loss: 0.3709 - accuracy: 0.8335 - val_loss: 0.4062 - val_accuracy: 0.8140 - 166ms/epoc
## Epoch 39/50
## 143/143 - 0s - loss: 0.3702 - accuracy: 0.8331 - val_loss: 0.4054 - val_accuracy: 0.8164 - 165ms/epoc
## Epoch 40/50
## 143/143 - 0s - loss: 0.3703 - accuracy: 0.8343 - val_loss: 0.4065 - val_accuracy: 0.8168 - 168ms/epoc
## Epoch 41/50
## 143/143 - 0s - loss: 0.3695 - accuracy: 0.8344 - val_loss: 0.4085 - val_accuracy: 0.8159 - 165ms/epoc
## Epoch 42/50
## 143/143 - 0s - loss: 0.3692 - accuracy: 0.8343 - val_loss: 0.4051 - val_accuracy: 0.8165 - 168ms/epoc
## Epoch 43/50
## 143/143 - 0s - loss: 0.3689 - accuracy: 0.8337 - val_loss: 0.4049 - val_accuracy: 0.8172 - 166ms/epoc
## Epoch 44/50
## 143/143 - 0s - loss: 0.3687 - accuracy: 0.8340 - val_loss: 0.4042 - val_accuracy: 0.8171 - 165ms/epoc
## Epoch 45/50
## 143/143 - 0s - loss: 0.3680 - accuracy: 0.8348 - val_loss: 0.4046 - val_accuracy: 0.8183 - 182ms/epoc
## Epoch 46/50
## 143/143 - 0s - loss: 0.3675 - accuracy: 0.8352 - val_loss: 0.4063 - val_accuracy: 0.8152 - 166ms/epoc
## Epoch 47/50
## 143/143 - 0s - loss: 0.3677 - accuracy: 0.8357 - val_loss: 0.4052 - val_accuracy: 0.8188 - 166ms/epoc
## Epoch 48/50
## 143/143 - 0s - loss: 0.3670 - accuracy: 0.8364 - val_loss: 0.4037 - val_accuracy: 0.8173 - 166ms/epoc
## Epoch 49/50
## 143/143 - 0s - loss: 0.3667 - accuracy: 0.8359 - val_loss: 0.4048 - val_accuracy: 0.8198 - 166ms/epoc
## Epoch 50/50
## 143/143 - 0s - loss: 0.3663 - accuracy: 0.8366 - val_loss: 0.4029 - val_accuracy: 0.8208 - 166ms/epoc
```

```r
plot(history)
```




```r
predictions <- predict(model, test_features)
```

```
## 284/284 - 0s - 159ms/epoch - 561us/step
```

```r
test_set$p_prob <- predictions[, 1]
head(predictions, 10)
```

```
##             [,1]
##  [1,] 0.16449691
##  [2,] 0.98300058
##  [3,] 0.10194805
##  [4,] 0.02920475
##  [5,] 0.07867236
##  [6,] 0.17860070
##  [7,] 0.03646662
##  [8,] 0.59434116
##  [9,] 0.05330301
## [10,] 0.06974465
```

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

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

```r
over_threshold <- test_set[test_set$p_prob >= 0.5, ]
fpr <- sum(over_threshold$booking_status==0)/sum(test_set$booking_status==0)
fpr
```

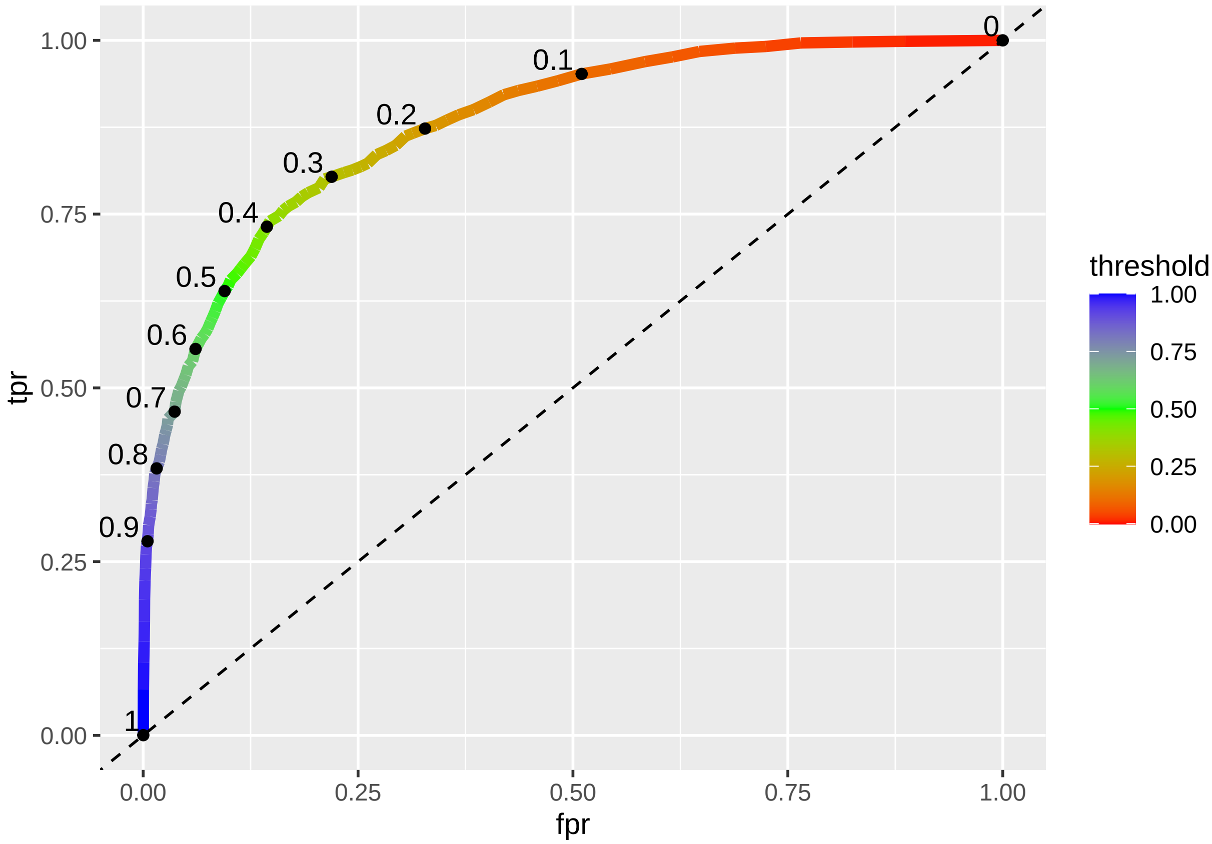
```
## [1] 0.09477124
```

```r
tpr <- sum(over_threshold$booking_status==1)/sum(test_set$booking_status==1)
tpr
```

```
## [1] 0.6393331
```

```r
### good to here so far ###

roc_data <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)
for (i in roc_data$threshold) {
  over_threshold <- test_set[test_set$p_prob >= i, ]
  fpr <- sum(over_threshold$booking_status==0)/sum(test_set$booking_status==0)
  roc_data[roc_data$threshold==i, "fpr"] <- fpr
  tpr <- sum(over_threshold$booking_status==1)/sum(test_set$booking_status==1)
  roc_data[roc_data$threshold==i, "tpr"] <- tpr
}
ggplot() +
  geom_line(data = roc_data, aes(x = fpr, y = tpr, color = threshold), size = 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))
```

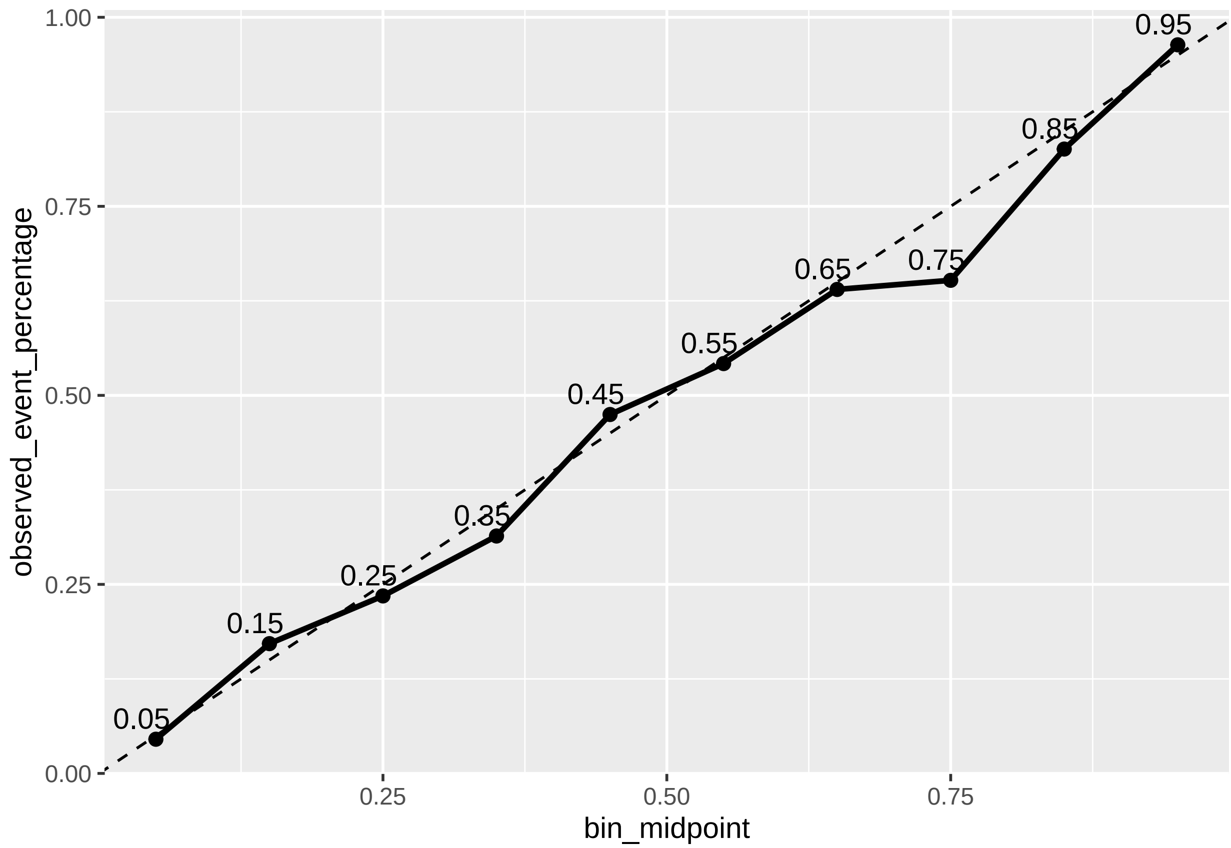
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




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

in_interval <- test_set[test_set$p_prob >= 0.7 & test_set$p_prob <= 0.8, ]
nrow(in_interval[in_interval$booking_status==1, ])/nrow(in_interval)
```

```
## [1] 0.6521739
```

```
calibration_data <- data.frame(bin_midpoint=seq(0.05,0.95,0.1),
                               observed_event_percentage=0)
for (i in seq(0.05,0.95,0.1)) {
  in_interval <- test_set[test_set$p_prob >= (i-0.05) & test_set$p_prob <= (i+0.05), ]
  oep <- nrow(in_interval[in_interval$booking_status==1, ])/nrow(in_interval)
  calibration_data[calibration_data$bin_midpoint==i, "observed_event_percentage"] <- oep
}
ggplot(data = calibration_data, aes(x = bin_midpoint, y = observed_event_percentage)) +
  geom_line(size = 1) +
  geom_abline(intercept = 0, slope = 1, lty = 2) +
  geom_point(size = 2) +
  geom_text(aes(label = bin_midpoint), hjust = 0.75, vjust = -0.5)
```


observed_event_percentage
1.00
0.75
0.50
0.25
0.00
0.05
0.15
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.25
0.50
0.75
bin_midpoint