Week5_SimonsenHomework

Steven Simonsen

2024-09-25

```r
getwd()
```

```
## [1] "C:/Users/steve/OneDrive/Documents/School/DSE6211/Week5"
```

```r
setwd("C:\\Users\\steve\\OneDrive\\Documents\\School\\DSE6211\\Week5")

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(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
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
library(MESS)

data <- read.csv("lab_5_data.csv")
```

```r
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, ]

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)

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")


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))
```

```r
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)))


use_virtualenv("my_tf_workspace")

set.seed(42)
model <- keras_model_sequential() %>%
  layer_dense(units = 50, activation = "relu") %>%
  layer_dense(units = 25, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

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

set.seed(42)
history <- fit(model, training_features, training_labels,
               epochs = 40, batch_size = 512, validation_split = 0.33)
```

```
## Epoch 1/40
## 9/9 - 1s - 80ms/step - accuracy: 0.5711 - loss: 0.7033 - val_accuracy: 0.4889 - val_loss: 0.7057
## Epoch 2/40
## 9/9 - 0s - 5ms/step - accuracy: 0.6689 - loss: 0.6290 - val_accuracy: 0.5320 - val_loss: 0.6872
## Epoch 3/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7045 - loss: 0.5941 - val_accuracy: 0.5626 - val_loss: 0.6839
## Epoch 4/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7289 - loss: 0.5694 - val_accuracy: 0.5695 - val_loss: 0.6871
## Epoch 5/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7415 - loss: 0.5502 - val_accuracy: 0.5732 - val_loss: 0.6893
## Epoch 6/40
## 9/9 - 0s - 6ms/step - accuracy: 0.7483 - loss: 0.5346 - val_accuracy: 0.5788 - val_loss: 0.6873
## Epoch 7/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7609 - loss: 0.5209 - val_accuracy: 0.5890 - val_loss: 0.6865
## Epoch 8/40
## 9/9 - 0s - 8ms/step - accuracy: 0.7680 - loss: 0.5087 - val_accuracy: 0.5867 - val_loss: 0.7001
## Epoch 9/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7687 - loss: 0.4996 - val_accuracy: 0.6001 - val_loss: 0.6928
## Epoch 10/40
```

```
## 9/9 - 0s - 5ms/step - accuracy: 0.7757 - loss: 0.4897 - val_accuracy: 0.6089 - val_loss: 0.6891
## Epoch 11/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7824 - loss: 0.4823 - val_accuracy: 0.6149 - val_loss: 0.6882
## Epoch 12/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7849 - loss: 0.4759 - val_accuracy: 0.6168 - val_loss: 0.6928
## Epoch 13/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7872 - loss: 0.4706 - val_accuracy: 0.6117 - val_loss: 0.7120
## Epoch 14/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7885 - loss: 0.4657 - val_accuracy: 0.6126 - val_loss: 0.7022
## Epoch 15/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7906 - loss: 0.4625 - val_accuracy: 0.6158 - val_loss: 0.7051
## Epoch 16/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7904 - loss: 0.4578 - val_accuracy: 0.6191 - val_loss: 0.7107
## Epoch 17/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7915 - loss: 0.4546 - val_accuracy: 0.6274 - val_loss: 0.7077
## Epoch 18/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7924 - loss: 0.4534 - val_accuracy: 0.6196 - val_loss: 0.7196
## Epoch 19/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7931 - loss: 0.4500 - val_accuracy: 0.6191 - val_loss: 0.7239
## Epoch 20/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7936 - loss: 0.4480 - val_accuracy: 0.6126 - val_loss: 0.7324
## Epoch 21/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7938 - loss: 0.4455 - val_accuracy: 0.6311 - val_loss: 0.7212
## Epoch 22/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7942 - loss: 0.4445 - val_accuracy: 0.6168 - val_loss: 0.7330
## Epoch 23/40
## 9/9 - 0s - 4ms/step - accuracy: 0.7956 - loss: 0.4418 - val_accuracy: 0.6186 - val_loss: 0.7365
## Epoch 24/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7979 - loss: 0.4395 - val_accuracy: 0.6172 - val_loss: 0.7438
## Epoch 25/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7949 - loss: 0.4380 - val_accuracy: 0.6186 - val_loss: 0.7475
## Epoch 26/40
## 9/9 - 0s - 5ms/step - accuracy: 0.7961 - loss: 0.4367 - val_accuracy: 0.6200 - val_loss: 0.7499
## Epoch 27/40
## 9/9 - 0s - 14ms/step - accuracy: 0.7981 - loss: 0.4343 - val_accuracy: 0.6260 - val_loss: 0.7458
## Epoch 28/40
## 9/9 - 0s - 7ms/step - accuracy: 0.8006 - loss: 0.4328 - val_accuracy: 0.6311 - val_loss: 0.7483
## Epoch 29/40
## 9/9 - 0s - 8ms/step - accuracy: 0.8022 - loss: 0.4315 - val_accuracy: 0.6307 - val_loss: 0.7515
## Epoch 30/40
## 9/9 - 0s - 6ms/step - accuracy: 0.8013 - loss: 0.4308 - val_accuracy: 0.6293 - val_loss: 0.7569
## Epoch 31/40
## 9/9 - 0s - 6ms/step - accuracy: 0.8029 - loss: 0.4279 - val_accuracy: 0.6284 - val_loss: 0.7621
## Epoch 32/40
## 9/9 - 0s - 6ms/step - accuracy: 0.8034 - loss: 0.4274 - val_accuracy: 0.6288 - val_loss: 0.7661
## Epoch 33/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8061 - loss: 0.4252 - val_accuracy: 0.6237 - val_loss: 0.7748
## Epoch 34/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8025 - loss: 0.4237 - val_accuracy: 0.6247 - val_loss: 0.7774
## Epoch 35/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8070 - loss: 0.4232 - val_accuracy: 0.6307 - val_loss: 0.7755
## Epoch 36/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8047 - loss: 0.4216 - val_accuracy: 0.6288 - val_loss: 0.7800
## Epoch 37/40
```

```
## 9/9 - 0s - 5ms/step - accuracy: 0.8059 - loss: 0.4201 - val_accuracy: 0.6279 - val_loss: 0.7835
## Epoch 38/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8084 - loss: 0.4192 - val_accuracy: 0.6293 - val_loss: 0.7841
## Epoch 39/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8091 - loss: 0.4173 - val_accuracy: 0.6316 - val_loss: 0.7872
## Epoch 40/40
## 9/9 - 0s - 5ms/step - accuracy: 0.8100 - loss: 0.4180 - val_accuracy: 0.6293 - val_loss: 0.7931
```

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

```
## 69/69 - 0s - 1ms/step
```

```
test_set$p_prob <- predictions[, 1]

head(predictions, 10)
```

```
##            [,1]
##  [1,] 0.9412008
##  [2,] 0.8120941
##  [3,] 0.8959284
##  [4,] 0.7213657
##  [5,] 0.3067715
##  [6,] 0.6732711
##  [7,] 0.7964265
##  [8,] 0.6596538
##  [9,] 0.8249925
## [10,] 0.8376285
```

```
over_threshold <- test_set[test_set$p_prob >= 0.5, ]

fpr <- sum(over_threshold$lodgepole_pine==0)/sum(test_set$lodgepole_pine==0)
fpr
```

```
## [1] 0.3315168
```

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

```
## [1] 0.8181818
```

```
over_threshold2 <- test_set[test_set$p_prob >= 0.75, ]
fpr2 <- sum(over_threshold2$lodgepole_pine==0)/sum(test_set$lodgepole_pine==0)
fpr2
```

```
## [1] 0.147139
```

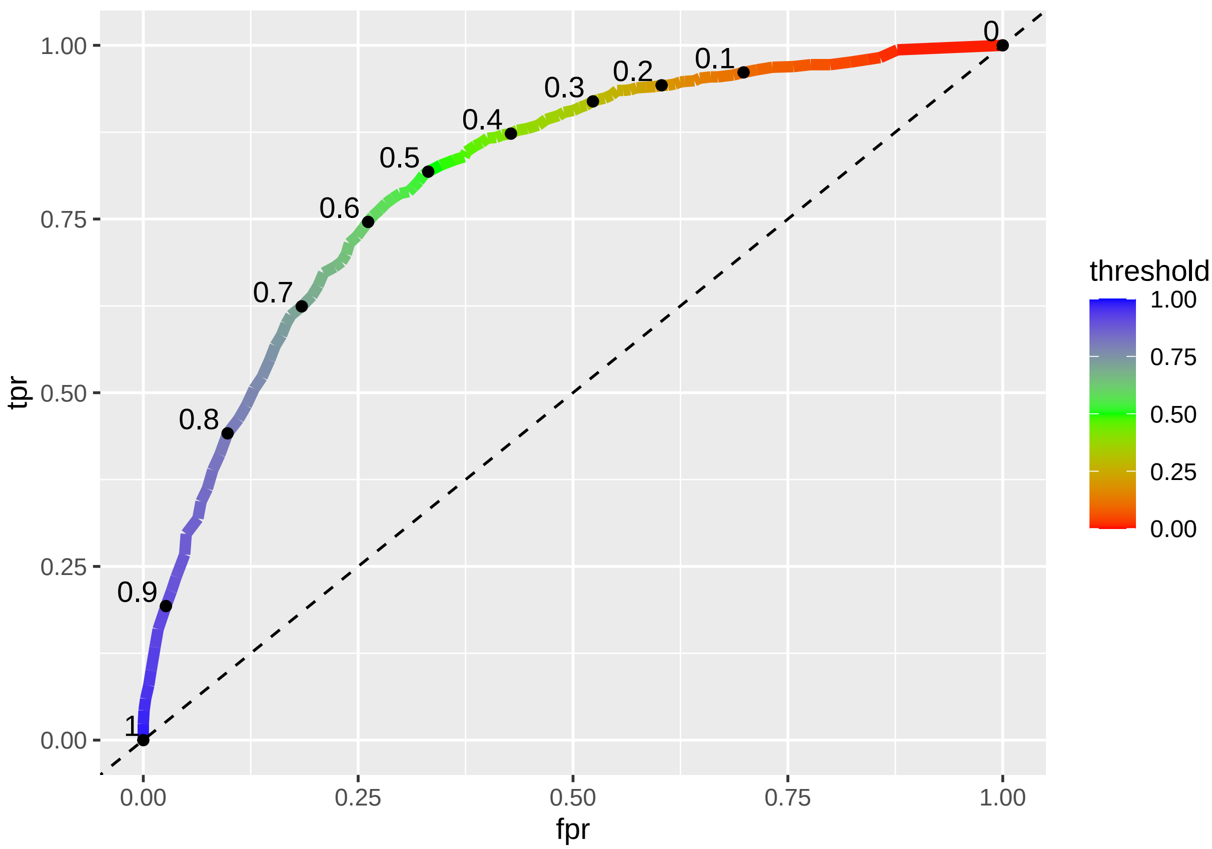
```
tpr2 <- sum(over_threshold2$lodgepole_pine==1)/sum(test_set$lodgepole_pine==1)
tpr2
```

```
## [1] 0.5473098
```

```
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$lodgepole_pine==0)/sum(test_set$lodgepole_pine==0)
  roc_data[roc_data$threshold==i, "fpr"] <- fpr
  tpr <- sum(over_threshold$lodgepole_pine==1)/sum(test_set$lodgepole_pine==1)
  roc_data[roc_data$threshold==i, "tpr"] <- tpr
}
```

```r
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.
```




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

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

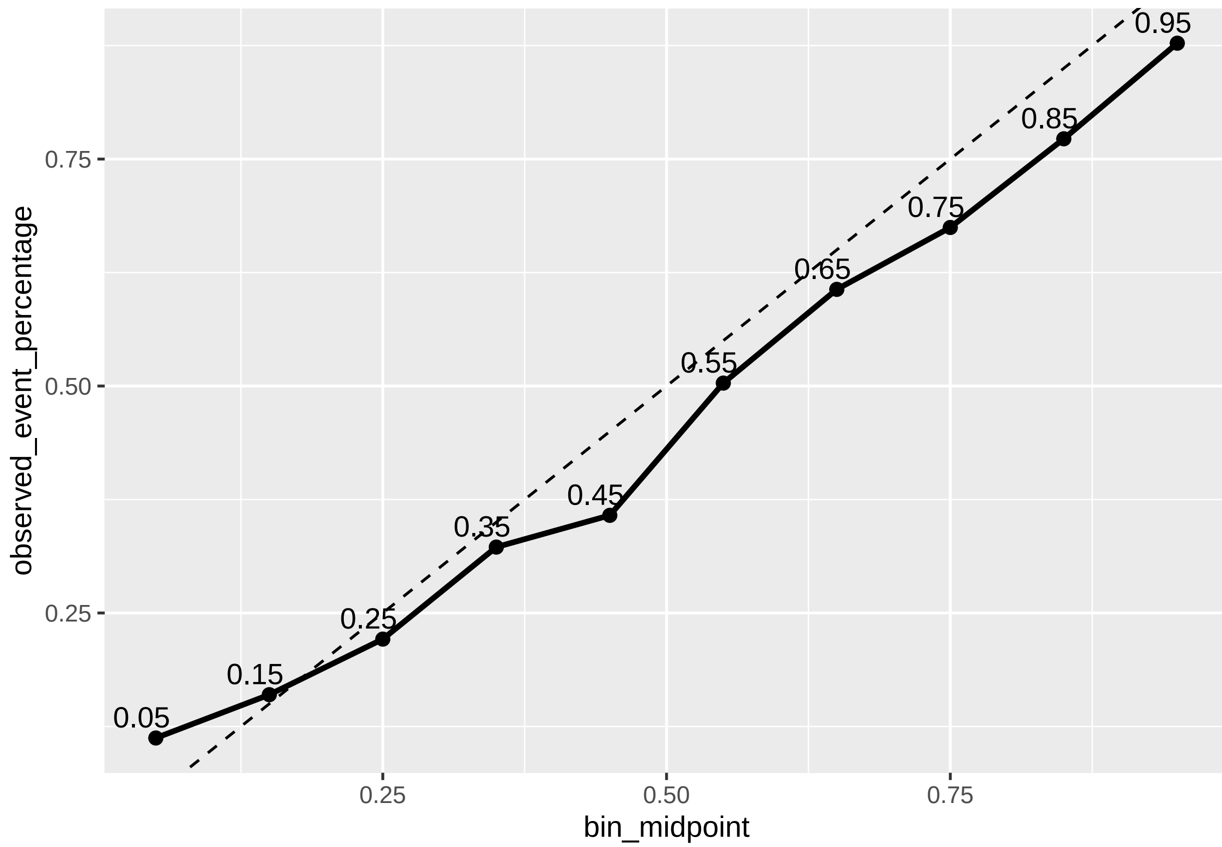
```r
auc
```

```
## [1] 0.8099783
```

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

```
## [1] 0.6746575
```

```r
set.seed(42)
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$lodgepole_pine==1, ])/nrow(in_interval)
  calibration_data[calibration_data$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)
```



#Exercises 1) In the ROC curve above, what is the TPR and FPR associated with the threshold value of 0.3?

```r
over_threshold3 <- test_set[test_set$p_prob >= 0.3, ]
fpr3 <- sum(over_threshold3$lodgepole_pine==0)/sum(test_set$lodgepole_pine==0)
fpr3
```

```
## [1] 0.5231608
```

```r
tpr3 <- sum(over_threshold3$lodgepole_pine==1)/sum(test_set$lodgepole_pine==1)
tpr3
```

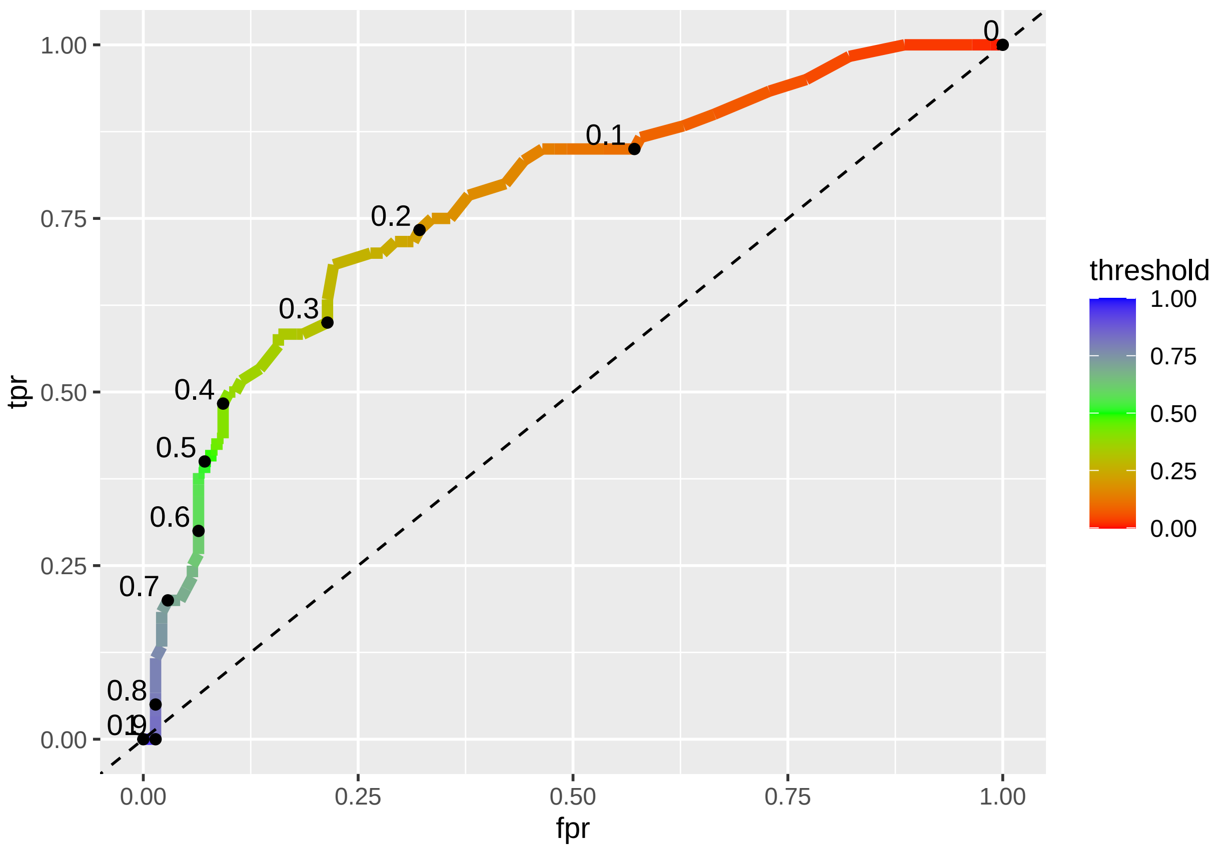
```
## [1] 0.919295
```

2) In the calibration curve above, are the predicted probabilities in the interval (0.2, 0.3) under-confidentor over-confident?

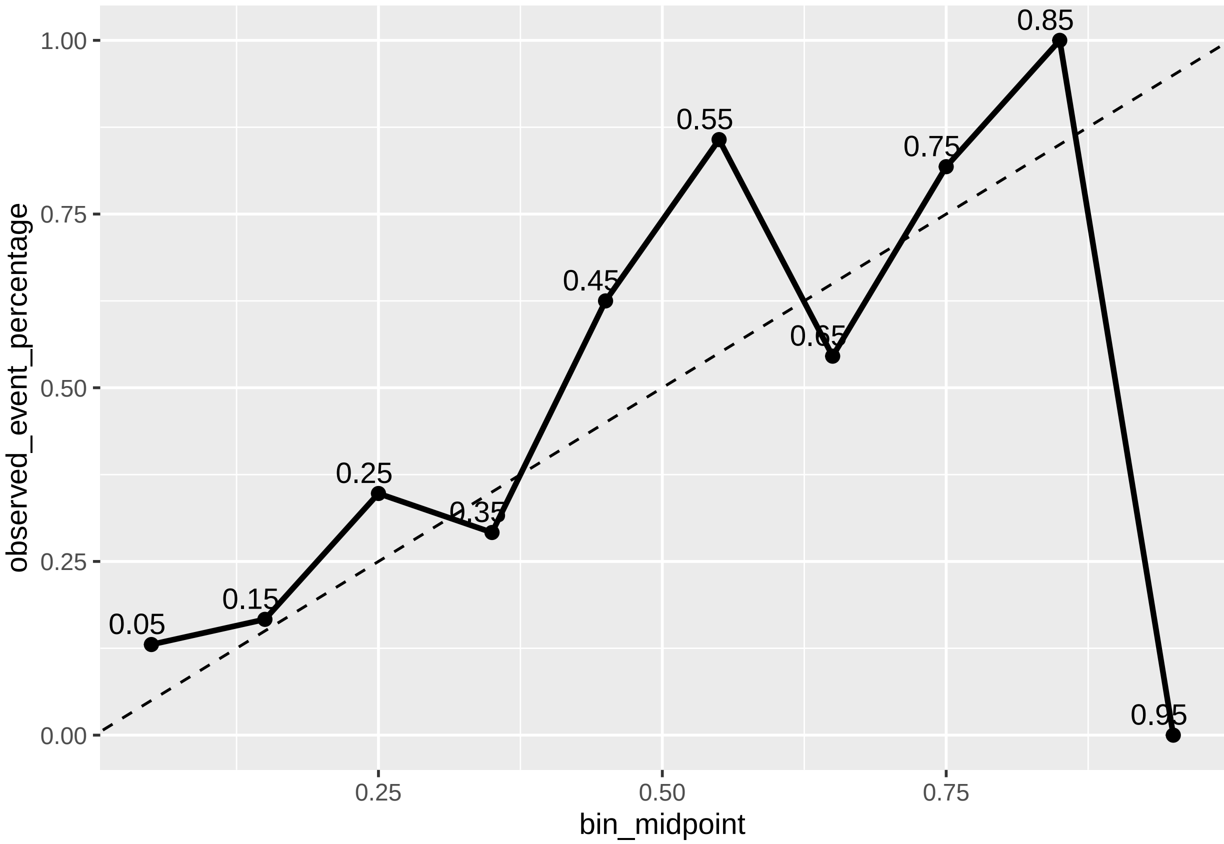
The predicted probabilities in the interval (0.2, 0.3) are over-confident since they lie below the dotted line, which represents a well calibrated classifier.

3) The 'AppliedPredictiveModeling' R package contains several datasets. One such dataset is the 'logisticCreditPredictions' dataframe, which contains the predictions and predicted probabilities for a credit dataset containing a binary target variable with the classes 'Good' and 'Bad'. The positive class is the 'Bad' class, since we are trying to identify customers with bad credit. The 'logisticCreditPredictions' dataframe has 4 columns: the columns 'Bad' and 'Good' contain the predicted probabilities of class membership, the column 'pred' contains the predicted class using the threshold 0.5, and the column 'obs' contains the actual class. Use the code below to plot an ROC curve and calibration curve for the predicted probabilities. To do this, fill in the question marks with the appropriate column names and values. Copy and paste the ROC curve and calibration curve.

```r
# Hint: the column 'pred' is not needed.
# Hint: there are a total of 8 question marks.
library(AppliedPredictiveModeling)
data("logisticCreditPredictions")
lcp <- logisticCreditPredictions # only do this to shorten the name

#### ROC curve
roc_data <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)

for (i in roc_data$threshold) {
  over_threshold <- lcp[lcp$Bad >= i, ]
  fpr <- sum(over_threshold$obs=="Good")/sum(lcp$obs=="Good")
  roc_data[roc_data$threshold==i, "fpr"] <- fpr
  tpr <- sum(over_threshold$obs=="Bad")/sum(lcp$obs=="Bad")
  roc_data[roc_data$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))
```




```
### calibration curve
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 <- lcp[lcp$Bad >= (i-0.05) & lcp$Bad <= (i+0.05), ]
  temp <- nrow(in_interval[in_interval$obs=="Bad", ])/nrow(in_interval)
  calibration_data[calibration_data$bin_midpoint==i, "observed_event_percentage"] <-
    temp
}

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)
```


1.00
0.75
0.50
0.25
0.00
observed_event_percentage
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