Statistical Methods Quiz #3 (40 minutes)

'25. 12. 1.(Monday) Cadet number(교번): Name: Score:

<Instructions>

1. It is open-book, closed-web test.
2. You are not allowed to discuss on the problems during the exam.
3. Import Movie.csv data first.
4. Run the provided code block (visible below) to prepare data and train a baseline model then answer the questions. Do not change the given R code.

<Baseline Code> - Run This First (Do not change the R code)

```
library(keras3); set.seed(12)

review = Movie
colnames(review) <- c("score", "comment")
review$score   <- as.integer(review$score)
review <- na.omit(review) # Remove NAs
review$score = ifelse(review$score <= 8, 0, 1) # Recode to 0-1 data

# ----------------------------
# Train / valid / test split
# ----------------------------
n    <- nrow(review) %/% 10
idx <- sample.int(n); review <- review[idx, ]

n_test  <- max(1, round(0.20 * n))
test_set  <- review[seq_len(n_test), ]; train_set <- review[-seq_len(n_test), ]

x_train <- train_set$comment; x_test <- test_set$comment

y_train <- train_set$score; y_test  <- test_set$score

# ------------------------------------------------
# Text vectorization (minimal preprocessing)
# ------------------------------------------------
# - Keeps text as-is (no lowercasing/stripping) because it's Korean.
max_words <- 20000L
maxlen    <- 50L  # you can raise to reduce truncation

vec <- layer_text_vectorization(
  max_tokens = max_words,
  output_mode = "int",
  output_sequence_length = maxlen,
  standardize = NULL # keep Korean punctuation/spacing as-is
)

# Build the vocabulary from training text only
adapt(vec, train_set$comment)

# Save the whole vocabulary (not necessary)
vocab <- keras3::get_vocabulary(vec); head(vocab, 30)

# ------------------------------------------------
# Model: TextVectorization -> Embedding -> BiLSTM
# ------------------------------------------------
units <- 64L

model <- keras3::keras_model_sequential(
  layers = list(
    vec,
    keras3::layer_embedding(input_dim = max_words, output_dim = units, mask_zero = TRUE),
    keras3::layer_lstm(units = units),
    keras3::layer_dense(units = 1, activation = "sigmoid")
  )
)

model |> compile(
  optimizer = "adam",
  loss = "binary_crossentropy",
  metrics = "accuracy"
)

# ------------------------------------------------
# Train
# ------------------------------------------------
history <- model |>
  fit(
    x = x_train, y = y_train,
    epochs = 5,
    batch_size = 64,
    validation_split = 0.2,
    verbose = 2
  )
```

1. Print a readable model summary. How many parameters are there in total?

<R-code>

```
summary(model)
```

<Output>

```
Model: "sequential_4"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ text_vectorization_4 (TextVectorization) │ (None, 50)                  │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ embedding_4 (Embedding)              │ (None, 50, 64)              │       1,280,0… │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ lstm_4 (LSTM)                        │ (None, 64)                  │          33,024 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 1)                   │              65 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 3,939,269 (15.03 MB)
 Trainable params: 1,313,089 (5.01 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,626,180 (10.02 MB)
```

The number of total parameters: 3,939,269

2. Print the accuracy using test dataset.

<R-code>

```
model |> evaluate(x_test, y_test)
```

<Output>

```
$accuracy
[1] 0.7297297

$loss
[1] 0.6561563
```

Accuracy: 0.7297297

3. Compute predicted probabilities, convert to class labels with threshold 0.6, and report the confusion matrix.

<R-code>

```
pred_prob <- model |> predict(x_test)
pred_lab <- as.integer(pred_prob > 0.6)
print(table(Predicted = pred_lab, Actual = y_test))
```

<Output>

```
         Actual
Predicted  0  1
        0 10 27
```

4. Show 4 most confidently correct predictions using the predicted probabilities. For each, print true label, predicted probability, and the raw text.

<R-code>

```
eg_idx <- tail(order(pred_prob), 4)
for (i in eg_idx) {
  cat("₩n--- Example ----------------------------₩n")
  cat("True:", y_test[i], "₩n")
  p <- as.numeric(model |> predict(x_test[i], verbose = 0))
  cat(sprintf("Pred prob (positive=1): %.3f₩n", p))
  cat("Text:", x_test[i], "₩n")
}
```

<Output>

```
--- Example ----------------------------
True: 1
Pred prob (positive=1): 0.552
Text:  시리즈 마지막 완전 아쉽다~~ 이제 무슨 낙으로 사나~이제 볼 수 없다니 슬플 뿐

--- Example ----------------------------
True: 1
Pred prob (positive=1): 0.553
Text:  스토리가 스릴있어서 정말 집중하고 보고 왔어요 헝거게임 더 파이널 재밌었습니다

--- Example ----------------------------
True: 1
Pred prob (positive=1): 0.553
Text:  다른 누군가는 결말이 허무하다 뒤로 갈수록 지루하다 하지만 저는 시즌4까지의 감정선 이야기의 과정 영화전체의 그영화만이 가진 이야기 사정 분위기 여운 모든게 깊이 남았어요 허무한결말이아닌 이 영화만의 결말이라고 생각합니다. 감사합니다.

--- Example ----------------------------
True: 1
Pred prob (positive=1): 0.555
Text:  나는 왕십리에서 4Dx로 봤다. 헝거게임 팬으로서 꼭 봤으면 끝나고 화장실 들어가는데 무슨 얘긴지 모르겠다고 난 안 봤어 이러길래 화가났다. 첫편부터 봐야한다. 볼 거면 이번 편만 보지 말고 총4회까지 꼭 보길 바란다. 쓰릴있고 감동있는 영화!
```