

Final Exam Solutions

Statistical Methods, Lab Exam(통계적방법론, 실기시험)

2025 2nd semester

| Year(학년) | 4 | 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 *travel.csv* data, converting all the string variables to factors, and import *Fantasy.csv* data without converting the string variables to factors.

Problem 1 [45 points]

We aim to predict the binary variable **TravelInsurance** in *travel.csv* using the remaining variables with a random forest model.

- (1) Import *travel.csv* dataset, converting all the string variables to factors. Use the **str** function to examine the structure of the *travel* data. How many observations, how many variables, and how many factor variables does the dataset contain?

<R-code>

```
str(travel)
```

<Output>

```
'data.frame':    1490 obs. of  9 variables:
 $ Age           : int  26 30 32 26 34 34 29 28 33 29 ...
 $ Employment.Type : Factor w/ 2 levels "Government Sector",...: 2 2 1 2 2 2 2 1 2 ...
 $ GraduateOrNot  : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 2 2 2 2 ...
 $ AnnualIncome   : num  1400000 1450000 900000 1400000 1400000 1500000 1100000 1100000
 700000 1200000 ...
 $ FamilyMembers  : int  5 5 4 7 3 5 8 6 6 3 ...
 $ ChronicDiseases : int  0 0 0 0 1 1 0 1 1 1 ...
 $ FrequentFlyer  : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 1 ...
 $ EverTravelledAbroad: Factor w/ 2 levels "No","Yes": 2 2 1 2 2 2 1 1 1 2 ...
 $ TravelInsurance : Factor w/ 2 levels "No","Yes": 2 2 1 2 2 2 1 1 2 2 ...
```

The number of observations: 1490

The number of variables: 9

The number of factor variables: 5

- (2) Randomly select **1000** observations from the *travel* data and save them as the training data(**travel.train**). Save the remaining observations as the test data(**travel.test**).

<R-code>

```
n = nrow(travel)
Index = sample(1:n, 1000)
travel.train = travel[Index,]
travel.test = travel[-Index,]
```

- (3) Using the training data, fit a **random forest** model with **TravellInsurance** as the response variable and all other variables as explanatory variables. (Set the number of variables randomly selected at each split to **3**.) Using the test data, obtain the predicted response variable. Then, create a **confusion matrix** and compute the **accuracy** and **sensitivity**.

```
<R-code>
library(randomForest)
model2 = randomForest(TravellInsurance ~ ., data = travel.train,
                      mtry = 3)
pred = predict(model2, travel.test, type = "class")
true = travel.test$TravellInsurance
table(pred, true)
mean(pred == true)
sum(true == "Yes" & pred == "Yes") / sum(true == "Yes")
```

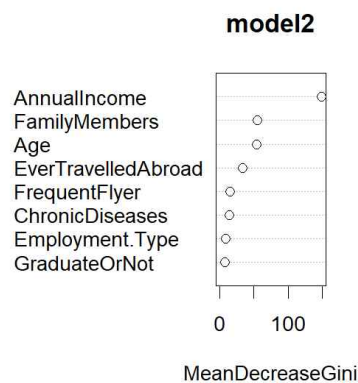
```
<Output>
      true
pred  No Yes
No   296  74
Yes   18 102
[1] 0.8122449
[1] 0.5795455
```

Accuracy: 0.8122449
 Sensitivity: 0.5795455

- (4) For the random forest model, visualize the **variable importance**. Which variable is the most important for predicting **TravellInsurance**?

```
<R-code>
varImpPlot(model2)
```

```
<Output (plot)>
```



The most important variable: AnnualIncome

Problem 2 [40 points]

We try to predict the binary variable `score` in `Fantasy.csv` using fantasy movie review with a Neural Network model. Import `Fantasy.csv` data without converting the string variables to factors.

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

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

review <- Fantasy
colnames(review) <- c("score", "comment")
review$score <- as.integer(review$score)
review <- na.omit(review)
review$score <- ifelse(review$score <= 8, 0L, 1L)

# -----
# Stratified train/test split
# -----
idx0 <- which(review$score == 0L)
idx1 <- which(review$score == 1L)

test0 <- sample(idx0, size = max(1, round(0.20 * length(idx0))))
test1 <- sample(idx1, size = max(1, round(0.20 * length(idx1))))

test_idx <- c(test0, test1)
train_idx <- setdiff(seq_len(nrow(review)), test_idx)

train_set <- review[train_idx, ]
test_set <- review[test_idx, ]

x_train <- train_set$comment
y_train <- train_set$score
x_test <- test_set$comment
y_test <- test_set$score

# -----
# Text vectorization (CHAR)
# -----
max_tokens <- 30000L # chars/character ngrams space
maxlen <- 250L # usually larger helps for reviews

vec <- layer_text_vectorization(
  max_tokens = max_tokens,
  output_mode = "int",
  output_sequence_length = maxlen,
  standardize = NULL,
  split = "character" # key change
)
adapt(vec, x_train)

# -----
# Model: Char Embedding + CNN
# -----
embed_dim <- 128L

inputs <- layer_input(shape = 1, dtype = "string")
x <- inputs |>
  vec() |>
  layer_embedding(input_dim = max_tokens, output_dim = embed_dim, mask_zero = FALSE) |>
  layer_spatial_dropout_1d(rate = 0.2) |>
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu", padding = "same") |>
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu", padding = "same") |>
  layer_global_max_pooling_1d() |>
  layer_dropout(rate = 0.4) |>
  layer_dense(64, activation = "relu") |>
  layer_dropout(rate = 0.3) |>
  layer_dense(1, activation = "sigmoid")

model <- keras_model(inputs, x)

model |> compile(
  optimizer = optimizer_adam(learning_rate = 1e-3),
  loss = "binary_crossentropy",
  metrics = c("accuracy", metric_auc(name = "auc"))
)

# -----
# Handle imbalance (if any)
# -----
p <- mean(y_train == 1L)
w0 <- 1 / (1 - p)
w1 <- 1 / p
class_weight <- list("0" = w0, "1" = w1)

# -----
# Train with callbacks
# -----
cb <- list(
  callback_early_stopping(monитор = "val_auc", mode = "max", patience = 3, restore_best_weights = TRUE),
  callback_reduce_lr_on_plateau(monитор = "val_auc", mode = "max", factor = 0.5, patience = 2)
)

history <- model |> fit(
  x = x_train, y = y_train,
  epochs = 30,
  batch_size = 128,
  validation_split = 0.2,
  class_weight = class_weight,
  callbacks = cb,
  verbose = 2
)

model |> evaluate(x_test, y_test, verbose = 0)
```

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

<R-code>

```
summary(model)
```

<Output>

Model: "functional"

| Layer (type) | Output Shape | Param # |
|---|------------------|-----------|
| input_layer (InputLayer) | (None, 1) | 0 |
| text_vectorization (TextVectorization) | (None, 250) | 0 |
| embedding (Embedding) | (None, 250, 128) | 3,840,000 |
| spatial_dropout1d (SpatialDropout1D) | (None, 250, 128) | 0 |
| conv1d (Conv1D) | (None, 250, 128) | 82,048 |
| conv1d_1 (Conv1D) | (None, 250, 128) | 82,048 |
| global_max_pooling1d (GlobalMaxPooling1D) | (None, 128) | 0 |
| dropout (Dropout) | (None, 128) | 0 |
| dense (Dense) | (None, 64) | 8,256 |
| dropout_1 (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 1) | 65 |

Total params: 12,037,253 (45.92 MB)
Trainable params: 4,012,417 (15.31 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 8,024,836 (30.61 MB)

The number of total parameters: 12,037,253

(2) Print the accuracy using test dataset.

<R-code>

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

<Output>

```
$accuracy  
[1] 0.8040816
```

```
$auc  
[1] 0.8990967
```

```
$loss  
[1] 0.403863
```

Accuracy: 0.8040816

- (3) Compute predicted probabilities, convert to class labels with threshold 0.5, and report the confusion matrix.

```
<R-code>
pred_prob <- model |> predict(x_test)
pred_lab <- as.integer(pred_prob > 0.5)
print(table(Predicted = pred_lab, Actual = y_test))
```

```
<Output>
      Actual
Predicted 0  1
         0 336 100
         1  92 452
```

- (4) Compute predicted probabilities. Print the labels, predicted probabilities, and raw text for the four observations predicted to be the least interesting.

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

```
<Output>
--- Example -----
True: 0
Pred prob (positive=1): 0.005
Text: 솔직히 돈 아까움 --- 최악

--- Example -----
True: 0
Pred prob (positive=1): 0.009
Text: 엄청나게 실망함.... 이걸 파트2랑 가치보는게 진리인듯 영화관 비추...

--- Example -----
True: 0
Pred prob (positive=1): 0.010
Text: 시작을 안한듯한느낌...

--- Example -----
True: 0
Pred prob (positive=1): 0.011
Text: 무슨 화살로 비행기가 터지나 --- 원작소설팬 아니면 2시간 버린다고 봄. 소설자체도 애들소설이라...트와일라잇처럼
```

Problem 3 [15 points]

Choose the statistical method from the word bank that best matches each description below.

Word bank:

Linear regression, Logistic regression, K-Nearest Neighbor(KNN)

Decision tree, Random forest, Cross-validation, Generalized Additive Model

Single Layer Neural Network, Convolutional Neural Network(CNN), Recurrent Neural Network(RNN)

Principal component analysis(PCA), K-means Clustering, Hierarchical Clustering

- (1) This tree-based method improves on bagging by choosing a random subset of predictors at each split: Random forest
- (2) This neural network model uses convolution and pooling operations and is known to perform well on image data: Convolutional Neural Network(CNN)
- (3) In this neural network model, the input is a sequence; it is known to work well for sequential or time-series data: Recurrent Neural Network(RNN)
- (4) This method computes principal components to reduce dimensionality; it is related to eigenvalue decomposition: Principal component analysis(PCA)
- (5) This clustering method aims to minimize within-cluster variation using an Expectation-Maximization (EM) algorithm and may depend on the initial cluster assignments: K-means Clustering