HW1 Answers

Yiwen Li

2024-02-13

Problem 1

(a)

```
set.seed(123)

library(tidyverse)
library(neuralnet)
library(dplyr)
library(ggplot2)
library(caret)
library(class)
```

(b)

```
# Load the data
my_data <- read.csv('income_evaluation.csv')
```

(c)

```
# Randomly select 3000 row indices
sample_indices <- sample(nrow(my_data), 3000)
my_data <- my_data[sample_indices, ]
```

(d)

Numerical: Age, fnlwgt, education.num, capital.gain, capital.loss, hours.per.week, Categorical: Workclass, education, marital.status, occupation, relationship, race, sex, native.country, income

(e)

income; levels: $> 50K$ and $<= 50K$

(f)

```
categorical_cols <- c("workclass", "education", "marital.status", "occupation",
                      "relationship", "race", "sex", "native.country")

# Step 2: Convert categorical variables to factors
my_data <- mutate_at(my_data, categorical_cols, as.factor)

# Step 3: Convert factors to appropriate numeric representations
# Using one-hot encoding (dummy variables)
my_data <- my_data %>%
  mutate_at(categorical_cols, ~as.integer(as.factor(.)))
```

(g)

```r
# Scale the variables
# variables_to_scale <- c("age", "fnlwgt", "education.num", "capital.gain",
#                          "capital.loss", "hours.per.week")
variables_to_scale <- colnames(my_data)[1:14]
my_data[variables_to_scale] <- scale(my_data[variables_to_scale])
my_data$income <- ifelse(my_data$income == " >50K", 1, 0)
```

(h)

```r
# Split the data into training (70%) and testing sets (30%)
data_rows <- floor(0.7 * nrow(my_data))
train_indices <- sample(c(1:nrow(my_data)), data_rows)
train_data <- my_data[train_indices,]
test_data <- my_data[-train_indices,]
```

Problem 2

(a)

```r
# Build the neural network with 1 layer and 5 neurons
model_1 <- neuralnet(
    income~.,
    data=train_data,
    hidden=c(5),
    linear.output=FALSE,
    learningrate = 0.0001
)
```

(b)

```r
nn.results <- compute(model_1, test_data)
results <- data.frame(actual = test_data$income, prediction = nn.results$net.result)
results$prediction <- ifelse(results$prediction > 0.5, 1, 0)
```

(c)

```r
# Check if the predicted classes match the actual classes
check_1 <- results$prediction == results$actual

# Calculate accuracy as a percentage
accuracy_1 <- (sum(check_1) / nrow(test_data)) * 100
print(paste("Accuracy:", accuracy_1, "%"))
```

```
## [1] "Accuracy: 81.4444444444444 %"
```

```r
# Confusion Matrix
confusionMatrix <- confusionMatrix(as.factor(results$prediction), as.factor(results$actual))
print(confusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 637 111
##          1  56  96
##
##                Accuracy : 0.8144
##                  95% CI : (0.7875, 0.8393)
##     No Information Rate : 0.77
##     P-Value [Acc > NIR] : 0.000691
##
##                   Kappa : 0.4223
##
##  Mcnemar's Test P-Value : 2.933e-05
##
##             Sensitivity : 0.9192
##             Specificity : 0.4638
##          Pos Pred Value : 0.8516
##          Neg Pred Value : 0.6316
##              Prevalence : 0.7700
##          Detection Rate : 0.7078
##    Detection Prevalence : 0.8311
##       Balanced Accuracy : 0.6915
##
##        'Positive' Class : 0
##
```

Problem 3

(a)

```
knn_model <- knn(train = train_data, test = test_data, cl = train_data$income, k = 10)
```

(b)

```
misClassError <- mean(knn_model != test_data$income)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.915555555555556"
```

```
# results_knn <- data.frame(actual = test_data$income, prediction = knn_model)
# check_2 <- results_knn$prediction == results_knn$actual
# accuracy_2 <- (sum(check_2) / nrow(test_data)) * 100
# print(paste("Accuracy:", accuracy_2, "%"))
```

```
confusionMatrix <- confusionMatrix(as.factor(knn_model), as.factor(test_data$income))
print(confusionMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 676  59
##          1  17 148
##
##                Accuracy : 0.9156
##                  95% CI : (0.8954, 0.9329)
##     No Information Rate : 0.77
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7433
##
##  Mcnemar's Test P-Value : 2.563e-06
```

```
##
##             Sensitivity : 0.9755
##             Specificity : 0.7150
##          Pos Pred Value : 0.9197
##          Neg Pred Value : 0.8970
##              Prevalence : 0.7700
##          Detection Rate : 0.7511
##    Detection Prevalence : 0.8167
##       Balanced Accuracy : 0.8452
##
##        'Positive' Class : 0
##
```

(c) Based on the performance metrics, the KNN model is preferable for this classification task. KNN outperforms the Neural Network model with an accuracy of 91.56% compared to 81.44%. This suggests that, overall, KNN makes more correct predictions for both positive and negative classes. KNN also has higher sensitivity, with 97.55% compared to Neural Network's 91.92%. In terms of specificity, KNN significantly outperformed the Neural Network, scoring 71.50% against 46.38%.

Problem 4

(a)

```r
km_model <- kmeans(my_data, centers = 2 , nstart = 30)
```

(b)

```r
my_data$cluster <- km_model$cluster
my_data$income_over_50K <- my_data$income == 1
proportions <- my_data %>%
  group_by(cluster) %>%
  summarise(ProportionOver50K = mean(income_over_50K))
print(proportions)
```

```
## # A tibble: 2 x 2
##   cluster ProportionOver50K
##     <int>             <dbl>
## 1       1            0.0738
## 2       2            0.342
```

From the outputs, cluster 1 has a proportion of approximately 7.38% of individuals earning over $50K, while cluster 2 has a significantly higher proportion, approximately 34.22%, of individuals earning over $50K. The substantial difference in proportions between the two clusters indicates that the k-means clustering has effectively grouped individuals into two segments based on similarities in their data that correlate with income levels. Cluster 2 likely represents a group with higher overall income levels or features strongly associated with higher income.

(c)

Both Neural Networks (NNs) and K-Nearest Neighbors (KNN) are directly applicable to our classification task of predicting income levels because they are supervised learning algorithms that learn from labeled data. NNs offer the ability to capture deep, complex patterns and are scalable with large datasets, while KNN provides a simpler, intuitive approach that can be very effective with a well-chosen distance metric and k value. K-means clustering, while not a classification algorithm, can still be valuable in providing insights into the structure of the dataset. In terms of our goal of correctly classifying income level, KNN might be the preferred model to apply because of its outstanding performance on our data.