CIS 635 Data Mining

Project Answer Sheet

Write or paste in the answers for the following questions from the project

Part 1

1. missing values – list the record id and your choice to handle the missing value(s):

1158 only bpSys was missing, so I calculated the mean

1525 only temp was missing, so mean is calculated and replaced with missing value

1702 6 columns(headA, bodyA, cough, runny, nausea, diarrhea) are missing. The entire row was discarded.

3378 4 columns (temp, bySys, vo2, throat). The row was removed

4059 Throat is missing. Mean was calculated

4210 6 columns(headA, bodyA, cough, runny, nausea, diarrhea) are missing. Row is removed.

5260 vo2 is missing. Row is removed.

2. noise – list the record id and what you did to handle the noise:

1856 temp was in negative. Replaced with mean value

2078 vo2 was negative value. Replaced with mean value.

2524 bpSys was negative value. Replaced with mean value

3. outliers – list the counts of outliers for each attribute:

temp 4

bpSys 21

vo2 2

throat 22

4. discretizing throat – list the bin ranges (low and high values for each bin) below:

bin 1 80 101

bin 2 102 104

bin 3 105 120

Part 2: clustering results (expand table if you used more than 5 rows)

1. centroids:

temp	bpSys	vo2	throat	headA	bodyA
99.87975	125.09	39.30843	100.09	5.015326	3.965517
97.98151	118.9983	40.55511	103.0013	2.986097	3.984773
97.99393	130.3298	30.92817	103.0668	3.009369	3.984385
100.12231	113.7364	39.56783	101.9186	5.036822	4.052326
99.82258	120.9092	40.61355	108.013	4.930586	4.013015

2. description of clusters:

cluster 1 Patients in this cluster has high systolic blood pressure and high headA values.

cluster 2 Patients in this cluster has low temperature

cluster 3 Patients in this cluster has high blood pressure but lower temperature.

cluster 4 Patients has the highest body temperatures with high headA and bodyA values.

cluster 5 Patients has the largest value for throat culture with headA and bodyA value also significantly higher.

3. optimum number of clusters 4

R code for finding optimum k

```
set.seed(123)
library(purrr)

# function to calculate within sum of squares
calculate_wss <- function(k) {
  kmeans(normalized_df, k, nstart = 10 )$tot.withinss
}
# clusters from 1 to 15
k <- 1:15
# apply calculate_wss function for k cluster and get vector
wss_values <- map_dbl(k, calculate_wss)

plot(k, wss_values,
       type="b", pch = 15, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within sum of squares")
```

Part 3: classification results

1. results of tests (precision, recall and accuracy)

	precision	recall	accuracy
decision tree	80.719	87.465	81.651
naive Bayes	79.123	85.515	79.663
KNN	79.691	86.350	80.428
SVM	78.260	85.236	78.899
ANN	82.432	84.958	81.804

2. analysis (which algorithm is best; defend your choice)

In the initial analysis of the result of the test, I found that the ANN classification model has the highest precision and accuracy. Due to this, ANN seems to be the ideal model of choice for performing the predictions. However, I identified numerous issues with ANN. For instance, ANN is very time-consuming for both training and predictions. The RAM and CPU consumption spiked significantly, making it harder to design the systems using ANN. ANN can also be harder to understand and prone to overfitting.

Furthermore, I calculated the accuracy for each classification model using 300 loops with different seed values to collect the accuracy performance and have a consistent test. In the next step, I calculated the interquartile range for each model from the accuracy value. I found the result to be 0 for all the classification models except for KNN, which had a value of 0.4. The following classification model yielded promising results: the decision tree with a precision of 80.71, 87.566 recall, and 81.651 accuracy value. I found decision trees are very fast to compute and easy to interpret the results. It also takes a deterministic approach, and we can easily understand what the model is doing. The decision trees generate good results with categorical data such as yes and no for risk and numerical data such as throat and blood pressure. In conclusion, I select the decision tree classification model for performing the periodic analysis of employee medical data.

Part 4: code for the weekly process

```
# Load library
library(e1071)
library(dplyr)

# Useful functions

get_normalized_data <- function(data) {
  norm_value <- (data - min(data)) / (max(data) - min(data))
  return(norm_value)
}

get_results <- function(c) {
  TP <- c[1, 1]
  FP <- c[1, 2]
  FN <- c[2, 1]
  TN <- c[2, 2]

  acc <- (TP + TN) / (TP + FP + FN + TN) * 100
  pre <- (TP) / (TP + FP) * 100
```

```
  rec <- (TP) / (TP + FN) * 100

  print(paste("Accuracy:", acc, "Precision:",  pre, "Recall:",  rec))

  return(c(acc, pre, rec))
}


################################ CLEAN AND MERGE DATA SOURCE
#######################################

load_and_merge_data <- function(file_name_a, file_name_b) {
  # load data
  a_df <- read.table(file_name_a, header = TRUE)
  b_df <- read.table(file_name_b, header = TRUE)

  # merge
  df <- merge(a_df, b_df, by = "id", all = TRUE)

  # Sort the data by id
  df <- df[order(df$id), ]

  # move atRisk column to end
  df <-  df %>% relocate(atRisk, .after = diarrhea)

  return(df)
}

train_df <- load_and_merge_data(file_name_a = "data8_A_train.txt", file_name_b =
"data8_B_train.txt")
test_df <- load_and_merge_data(file_name_a = "data8_A_test.txt", file_name_b =
"data8_B_test.txt")

cat("Dataset Summary:\n")
print(summary(train_df))

# handle missing data
handle_missing_data <- function(df) {
  missing_df = df[!complete.cases(df), ]
  print("Below dataset are missing")
  print(missing_df)

  print("Cleaning missing data with more than 2 empty columns")
  ind <- rowSums(is.na(df)) > 2
  missing_df <- df[ind,]

  print("Following rows has more thant 2 missing columns")
  print(missing_df)

  # remove
  df <- df[!ind,]

  print("Remaining missing data: ")
  print(df[!complete.cases(df),])

  print("Estimating missing data for remaining:")
  # estimate values for temp, bpSys, vo2 and throat using mean
  for (i in 2:7) {
    df[, i][is.na(df[, i])] <- mean(df[, i], na.rm = TRUE)
```

```
  }

  print("checking for missing data:")
  print(df[!complete.cases(df),])

  return(df)
}

print("Handling missing data:")
print(paste("Number of rows:",nrow(train_df)))

train_df <- handle_missing_data(train_df)

print(paste("Number of rows after cleaning:",nrow(train_df)))


handle_noise <- function(df) {
  # assign NA to values less than 0
  df[df < 0] = NA

  # Find noise
  noise_df = df[!complete.cases(df), ]
  print(paste("Rows with noise:", nrow(noise_df)))

  # only temp, bpSys, vo2 has noise in 3 rows
  # estimate values for temp, bpSys, vo2 using mean
  for (i in 2:4) {
    df[, i][is.na(df[, i])] <- mean(df[, i], na.rm = TRUE)
  }

  return(df)
}

print("Handling noise data:")
train_df <- handle_noise(train_df)

# remove id as it is not required during training
train_df <- subset(train_df, select=-id)

# calculate normalized data
print("Normalized data")
normalized_df <- as.data.frame(lapply(train_df[1:6], get_normalized_data))

################################### DECISION TREE (rpart)
#########################################
library(rpart)
library(rpart.plot)

build_decision_tree <- function(df, test_df) {
  set.seed(123)
  # combine normalized data and factor of
  dt_df <- cbind(df[1:4], as.data.frame(lapply(df[5:11], as.factor)))

  # build model
  dt_mod  <- rpart(atRisk ~ ., dt_df, method = "class")

  print(rpart.plot(dt_mod))
  print(dt_mod)
```

```
  # perform predictions
  dt_test_df <- cbind(test_df[1:5], as.data.frame(lapply(test_df[6:12], as.fac-
tor)))
  dt_pred = predict(dt_mod, dt_test_df, type = "class")
  # create confusion matrix
  dt_c_matrix <- table(dt_pred, dt_test_df[, 12])
  results <- get_results(dt_c_matrix)

  return(dt_pred)
}

print("DECISION TREE (rpart)  **********************")

dt_pred <- build_decision_tree(train_df, test_df)

######################################## NAIVE BAYES
################################################

build_naive_bayes <- function(df, test_df) {
  naive_bayes_df <- cbind(df[1:4], as.data.frame(lapply(df[5:11], as.factor)))
  naive_bayes_test_df = cbind(test_df[1:5], as.data.frame(lapply(test_df[6:12],
as.factor)))

  # build model
  naive_bayes_model = naiveBayes(atRisk ~ ., naive_bayes_df)
  print(naive_bayes_model)

  # Predictions for naive bayes
  naive_bayes_pred = predict(naive_bayes_model, naive_bayes_test_df)

  naive_bayes_c_matrix <- table(naive_bayes_pred, naive_bayes_test_df[, 12])
  names(dimnames(naive_bayes_c_matrix)) <- c("predicted", "actual")
  get_results(naive_bayes_c_matrix)

  return(naive_bayes_pred)

}

print("NAIVE BAYES  **********************")
naive_bayes_pred <- build_naive_bayes(train_df, test_df)

######################################### KNN
################################################
set.seed(123)
library(class)

build_knn <- function(df, test_df, class) {
  knn_df = data.frame(df)
  knn_test_df <- as.data.frame(lapply(test_df[2:7], get_normalized_data))

  # run knn
  knn_pred <- knn(knn_df, knn_test_df, cl = class, k = 78, prob=TRUE)


  # create confusion matrix
  knn_c_matrix <- table(knn_pred, test_df$atRisk)

  # print the results
  knn_results <- get_results(knn_c_matrix)
```

```
  return(knn_pred)
}

print("KNN  ***********************")
knn_pred <- build_knn(normalized_df, test_df, train_df$atRisk)



########################################## SVM
#########################################################
set.seed(123)

build_svm <- function(df, test_df, at_risk) {
  svm_df <- data.frame(df)
  svm_df$atRisk <- as.factor(at_risk)
  svm_model <- svm(atRisk~., data = svm_df, kernel = "linear", cost = 10, scale =
FALSE)

  print(svm_model)

  svm_test_df <- as.data.frame(lapply(test_df[2:7], get_normalized_data))
  svm_test_df$atRisk = test_df$atRisk
  svm_pred <- predict(svm_model, svm_test_df, type="vector")

  svm_c_matrix <- table(svm_pred, svm_test_df[,7])

  svm_results <- get_results(svm_c_matrix)

  return(svm_pred)
}


print("SVM  ***********************")
svm_pred <- build_svm(normalized_df, test_df, train_df$atRisk)

########################################## ANN
#########################################################
library(neuralnet)
set.seed(123)

build_ann <- function(df, test_df, at_risk) {
  ann_df <- data.frame(df)
  ann_df$atRisk <- at_risk
  nn_model <- neuralnet(atRisk~bpSys+vo2+throat+headA+bodyA+temp, data = ann_df,
hidden = 5)
  plot(nn_model)

  # build predictions
  ann_test_df <- as.data.frame(lapply(test_df[2:7], get_normalized_data))
  ann_test_df$atRisk = (test_df$atRisk)

  ann_pred <- neuralnet::compute(nn_model, ann_test_df)

  ind = ann_pred$net.result < 0.5
  ann_pred$net.result[ind] <- 0
  ann_pred$net.result[!ind] <- 1
```

```
  ann_c_matrix <- table(as.factor(c(ann_pred$net.result)), as.fac-
tor(ann_test_df$atRisk))

  ann_results <- get_results(ann_c_matrix)

  return(as.factor(c(ann_pred$net.result)))
}

print("ANN  **********************")
ann_pred <- build_ann(normalized_df, test_df, train_df$atRisk)



# Write predictions to test file
final_preds <-
  data.frame(
    atRisk = test_df$atRisk,
    decision_tree = dt_pred,
    naive_bayes = naive_bayes_pred,
    knn = knn_pred,
    svm = svm_pred,
    ann = ann_pred
  )

write.table(final_preds, file = "final_preds.txt", sep = "\t", row.names = FALSE,
quote = FALSE)
```

Part 5: executive summary

The weekly employee medical data comes from two sources. Source A includes vital measurements of the body. The data source B includes the weekly survey results. Most of the attributes are numerical data for source A, whereas the data source B has nominal attributes. Both sources have an attribute ID which can be used to merge the two data sets. Furthermore, source A contains temp, bpSys, vo2, and throat as numeric variables and atRisk as nominal variables. The source B files include ordinal variables headA and bodyA. It also has nominal variables such as cough, runny, nausea, and diarrhea. To further understand the nature of data, I calculated the correlation between the variables using ggplot2 and cor function in the stats library. However, no correlation exists between the variables, which was quite interesting.

For prediction using the classification models, I cleaned the missing values in the dataset. I removed the rows having four or more missing values. For the missing numeric values, means were calculated. A similar approach is taken for noisy data as well. Interestingly the noise only included negative values. In the next step, the data is normalized using min-max normalization. Then models were generated for decision trees, Naive Bayes, K nearest neighbors, ANN, and KNN classification models. Then, the models were predicted, and the confusion matrix for each model was generated. Lastly, I calculated precision, recall, and accuracy of each prediction and saved it in the file.