Université Paris Cité

Chemoinformatic project
M1- ISDD

Student name: Van Thinh TO
Student ID: 22518830

1 Preparation of data and computation of molecule similarity

1.1 Analysis of the data set

```
# load the necessary package
library(ChemmineR)
```

```
# load data set
training_data <- read.csv("data/training_data.csv", sep=",")
dim(training_data)
```

```
## [1] 224945      5
```

```
colnames(training_data)
```

```
## [1] "SID"       "partition" "activity"  "potency"   "smiles"
```

```
table(training_data$activity)
```

```
##
##   Active Inactive
##     1453   223492
```

How many molecules are included in the data set? How many are Active and how many are Inactive? There are 224945 molecules in the data set, which have 1453 active and 223492 inactive data points.

What information is provided in each column? What is the data type of each variable?

There are five columns in the data set: * “SID”: a unique identifier (integer) * “partition”: Indicates whether the molecule belongs to the Training or Test set (character) * “activity”: Biological class: Active or Inactive (character) * “potency”: Numerical activity value (e.g., –log(IC50) or pIC50) (numeric) * “smiles”: Smiles string for each molecule (character)

```
str(training_data)
```

```
## 'data.frame':    224945 obs. of  5 variables:
##  $ SID      : int  405266972 405266969 405258654 405266973 405266971 439657910 440228774 381749740 4
##  $ partition: chr  "Training" "Training" "Training" "Training" ...
##  $ activity : chr  "Active" "Active" "Active" "Active" ...
##  $ potency  : num  9.3 9.3 9.3 9.3 9.22 ...
##  $ smiles   : chr  "C1=CC(=C(C=C1N2C=C(N=N2)CN3C=CN=C3N)C(F)F)F" "C1=CC(=C(C=C1N2C=C(N=N2)CN3C=CN=C3
```

Check whether the data set contains any missing values (NA). If this is the case, remove all molecules with missing entries. How many molecules are removed?

```
# get nan data points
# which(is.na(training_data), arr.ind = TRUE)
sum(is.na(training_data))
```

```
## [1] 223157
```

```
# get nan values for each column
sapply(training_data, function(x) sum(is.na(x)))
```

```
##       SID partition  activity   potency    smiles
##         0         0         0    223157         0
```

```
# drop rows with nan value
training_data <- na.omit(training_data)
dim(training_data)
```

```
## [1] 1788    5
```

1.2 Exploring the Chemical Space and Computing Molecular Similarity

```
# create data set for the sdf conversion
herg <- training_data[c("smiles", "SID")]
write.table(herg, "data/herg.smi", sep = "\t", quote = FALSE, col.names = FALSE,
↪  row.names = FALSE)
```

```
# read sdf file after creation
sdf_df <- read.SDFset("data/herg.sdf")
```

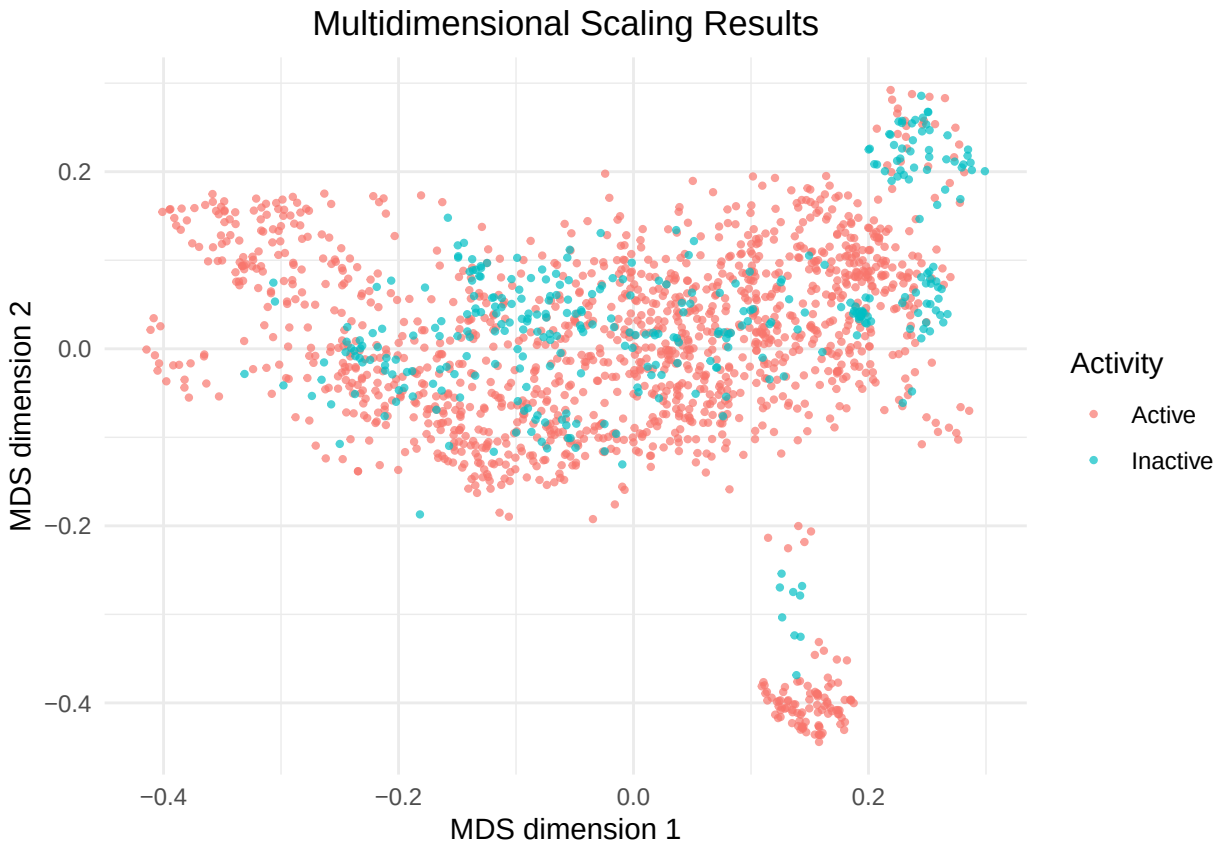
```
# compute the AP fingerprints
ap <- sdf2ap(sdf_df)
```

```
# compute the similarity matrix
# simi_matrix = matrix(0, nrow=length(ap),ncol=length(ap))

# for (i in 1:length(ap)){
#   for (j in i:length(ap)){
#     simi_matrix[i, j] <- cmp.similarity(ap[i], ap[j])
#     simi_matrix[j, i] <- cmp.similarity(ap[i], ap[j])
#   }
# }

# save similarity matrix for the later usage
# save(simi_matrix, file="data/simi_matrix.RData")
```

```r
# load the similarity matrix
load("data/simi_matrix.RData")
```

```r
# compute the dissimilarity matrix
dissimi_matrix <- 1-simi_matrix

# perform MDs
mds <- cmdscale(dissimi_matrix)
```

```r
# Visualization MDs
# load ggplot
library(ggplot2)

# create data frame for visualization
df_mds <- data.frame(
  x = mds[,1],
  y = mds[,2],
  activity = training_data$activity
)

# plot
ggplot(df_mds, aes(x = x, y = y, color = activity)) +
  # scatter plot
  geom_point(size = 0.8, alpha = 0.7) +
  labs(
    x = "MDS dimension 1",
    y = "MDS dimension 2",
    title = "Multidimensional Scaling Results",
    color = "Activity"
  ) +
  # set theme
  theme_minimal() +
  # align the title
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



We can observe that the molecule are widely dispersed. However, there is a huge overlap between active and inactive data points.

```
table(training_data$activity)
```

```
##
##   Active Inactive
##     1453      335
```

2 Balancing the dataset and preparing training and test sets

2.1 Cluster-based undersampling method

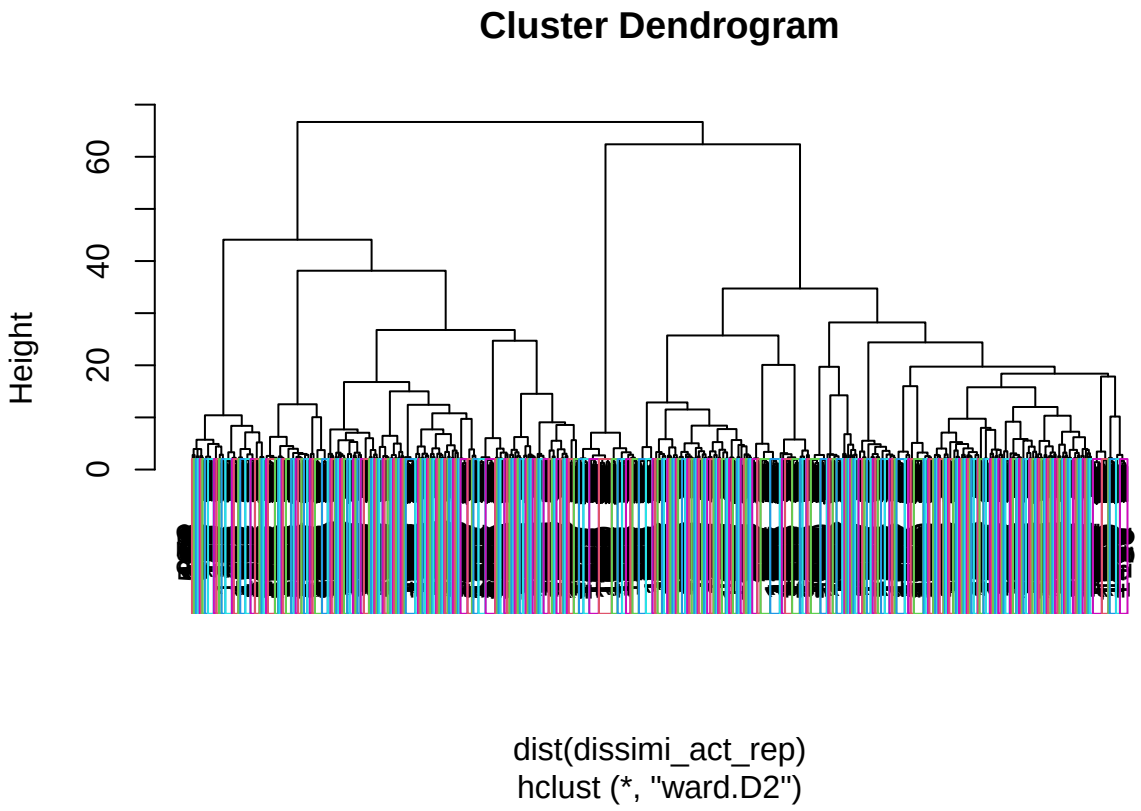
```
# Extract active molecules
active_idx <- which(training_data$activity == "Active")
```

```
# Calculate the dissimilarity matrix for actives only
dissimi_act_rep <- 1 - simi_matrix[active_idx, active_idx]
```

```
# Perform hierarchical clustering
hc <- hclust(dist(dissimi_act_rep), method = "ward.D2")
```

```r
# Cut the dendrogram into 350 clusters
active_groups <- cutree(hc, k=350)
```

```r
# plot the clusters
plot(hc)
rect.hclust(hc , k = 350, border = 2:6)
```

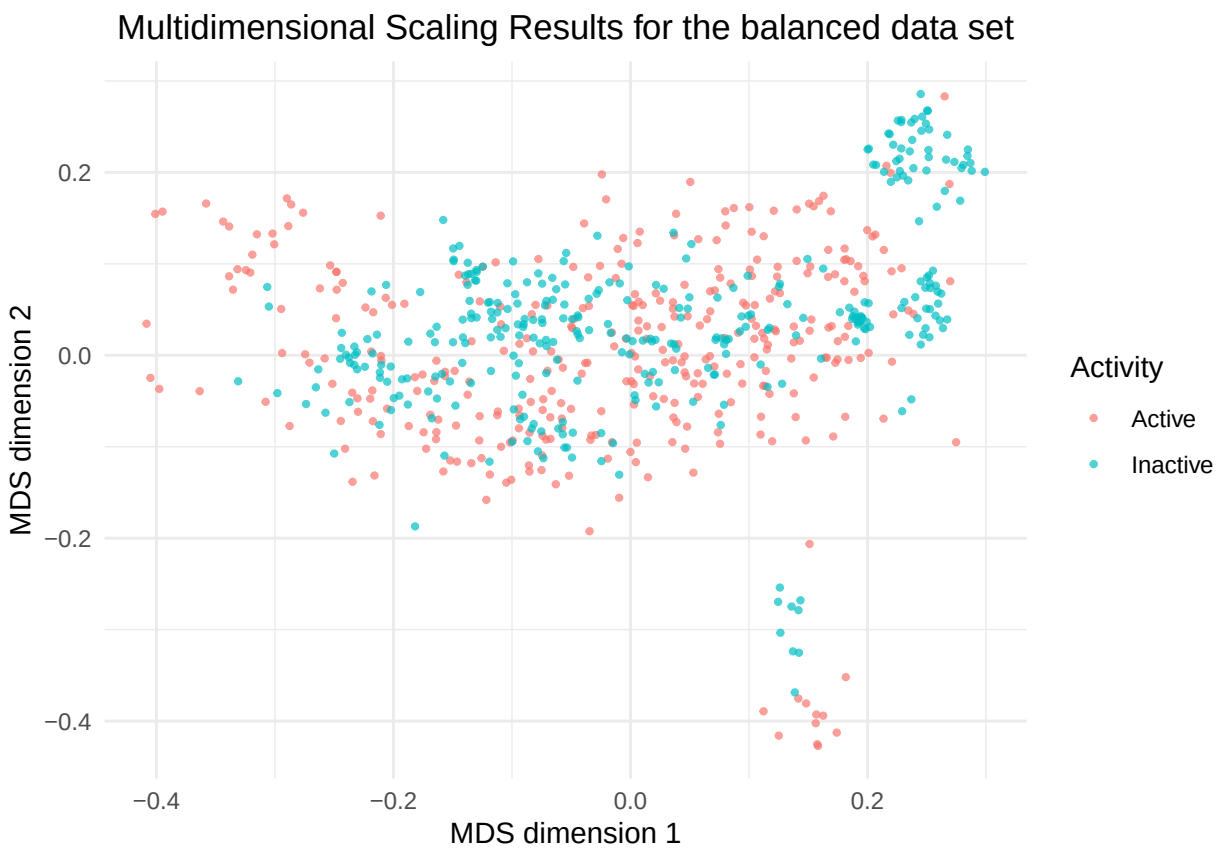



```r
# Select one molecule per cluster
set.seed(20022001) # seed for reproduction
selected_active_idx <- c()
for (group in 1:350){
  # for each group, randomly choose one active molecule, then append to a vector
  selected_active_idx <- c(selected_active_idx, sample(which(active_groups==group),
↪  size=1))
}
```

```r
# create a balanced data set
# select inactive
inactive_idx <- which(training_data$activity == "Inactive")

# concate inactive and selected active
balanced_df <- training_data[c(selected_active_idx, inactive_idx),]
# rownames(balanced_df) <- NULL # reset row names
# rownames(balanced_df) <- 1:nrow(balanced_df)
dim(balanced_df)
```

```
## [1] 685   5
```

```r
# Visualization MDs
# load ggplot
library(ggplot2)

# create data frame for visualization
df_mds_selected <- data.frame(
  # select mds coordinations for the balanced data set
  x = mds[c(selected_active_idx, inactive_idx),1],
  y = mds[c(selected_active_idx, inactive_idx),2],
  activity = balanced_df$activity
)

# plot
ggplot(df_mds_selected, aes(x = x, y = y, color = activity)) +
  # scatter plot
  geom_point(size = 0.8, alpha = 0.7) +
  labs(
    x = "MDS dimension 1",
    y = "MDS dimension 2",
    title = "Multidimensional Scaling Results for the balanced data set",
    color = "Activity"
  ) +
  # set theme
  theme_minimal() +
  # align the title
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



According to the Figure above, the cluster-based undersampling method avoids redundancy, preserves chemical diversity, and produces a balanced dataset suitable for building robust predictive models.

```
# save the balanced data set
save(balanced_df, file = "data/balanced_df.RData")
```

2.2 Preparing training and test set

```
# randomly select 80% of molecules from the balanced data set
training_idx <- sample(1:nrow(balanced_df), size = 8/10 * nrow(balanced_df))

# get the training and test sets
training_df <- balanced_df[training_idx,]
test_df <- balanced_df[-training_idx,]
```

```
# check the data points for each partition
print(dim(balanced_df))
```

```
## [1] 685   5
```

```
print(dim(training_df))
```

```
## [1] 548   5
```

```r
print(dim(test_df))
```

```
## [1] 137   5
```

```r
# Create a new variable to indicate whether each molecule belongs to the training or the
↪  test set
balanced_df[-training_idx,"partition"] <- "Test"
table(balanced_df$partition)
```

```
##
##     Test Training
##      137      548
```

```r
# check the distribution of Active and Inactive molecules in the training set
table(training_df$activity)/nrow(training_df)
```

```
##
##    Active  Inactive
## 0.5182482 0.4817518
```

```r
# check the distribution of Active and Inactive molecules in the test set
table(test_df$activity)/nrow(test_df)
```

```
##
##    Active  Inactive
## 0.4817518 0.5182482
```

```r
# save the training and test sets
save(training_df, file = "data/training.RData")
save(test_df, file = "data/test.RData")
```

3 Prediction of molecule class (Active / Inactive) Using a k-NN Approach based on chemical similarity

3.1 First steps with the k-NN algorithm

3.1.1 Manual computation

```r
# Extract the training-training submatrix
# get training-training indices
subtraining_idx <- strtoi(row.names(training_df))
# get submatrix
subtraining_simi_matrix <- simi_matrix[subtraining_idx, subtraining_idx]
```

```r
# Assign NA to the diagonal of this submatrix
diag(subtraining_simi_matrix) = NA
```

```r
# get the index of three molecules, which have the highest similarity compared to
↪  molecule 5
man_idx <- order(subtraining_simi_matrix[5,], decreasing = TRUE)[1:3]
```

```r
# Identify the activity classes of the 3 neighbors of molecule 5
votes <- training_df$activity[man_idx]
votes
```

```
## [1] "Active"   "Inactive" "Inactive"
```

```r
# Determine the majority class
names(which.max(table(votes)))
```

```
## [1] "Inactive"
```

3.1.2 Building a k-NN model on the training set

```r
# function for training the knn model
knn_train <- function(train, simi_matrix, k){
  pred <- c()
  for (i in 1:nrow(train)){
    # get the indices of k neighbors
    neighbor_idx <- order(simi_matrix[i,], decreasing = TRUE)[1:k]
    # majority voting
    votes <- train$activity[neighbor_idx]
    pred_label <- names(which.max(table(votes)))
    # update the label of the prediction vector
    pred <- c(pred, pred_label)
  }
  # compute the confusion matrix
  confusion_matrix <- table(True = train$activity, Predicted = pred)

  return (confusion_matrix)
}

# function for evaluating the knn model
knn_eval <- function(train, test, simi_matrix, k){
  pred <- c()
  for (i in 1:nrow(test)){
    # get the indices of k neighbors
    neighbor_idx <- order(simi_matrix[i,], decreasing = TRUE)[1:k]
    # majority voting
    votes <- train$activity[neighbor_idx]
    pred_label <- names(which.max(table(votes)))
    # update the label of the prediction vector
    pred <- c(pred, pred_label)
```

```
  }
  # compute the confusion matrix
  confusion_matrix <- table(True = test$activity, Predicted = pred)

  return (confusion_matrix)
}

# function to extract classification metrics
class_metrics <- function(confusion_matrix){
  # extract TP, FN, FP, TN values
  TP <- confusion_matrix["Active", "Active"]
  FN <- confusion_matrix["Active", "Inactive"]
  FP <- confusion_matrix["Inactive", "Active"]
  TN <- confusion_matrix["Inactive", "Inactive"]

  # calculate metrics
  accuracy  <- (TP + TN) / sum(confusion_matrix)
  recall    <- TP / (TP + FN)
  precision <- TP / (TP + FP)
  specificity <- TN / (TN + FP)
  balance_accuracy <- mean(recall, specificity)
  f1_score <- (2 * recall * precision)/(recall + precision)

  return (
    list(
      accuracy=accuracy,
      recall=recall,
      precision=precision,
      specificity=specificity,
      balance_accuracy = balance_accuracy,
      f1_score = f1_score
      )
  )
}
```

3.2 Implementing the k-NN model: selection of the optimal number of neighbors

```
# randomly select 90% of molecules from the training data set and 10% for the validation
↪  set
train_idx <- sample(1:nrow(training_df), size = 9/10 * nrow(training_df))

# get the training and test sets
train_df <- training_df[train_idx,]
valid_df <- training_df[-train_idx,]
```

```
# Extract the training-training submatrix
# get training-training indices
train_idx <- strtoi(row.names(train_df))
# get submatrix
train_simi_matrix <- simi_matrix[train_idx, train_idx]
```

```r
# Assign NA to the diagonal of this submatrix
diag(train_simi_matrix) = NA
dim(train_simi_matrix)
```

```
## [1] 493 493
```

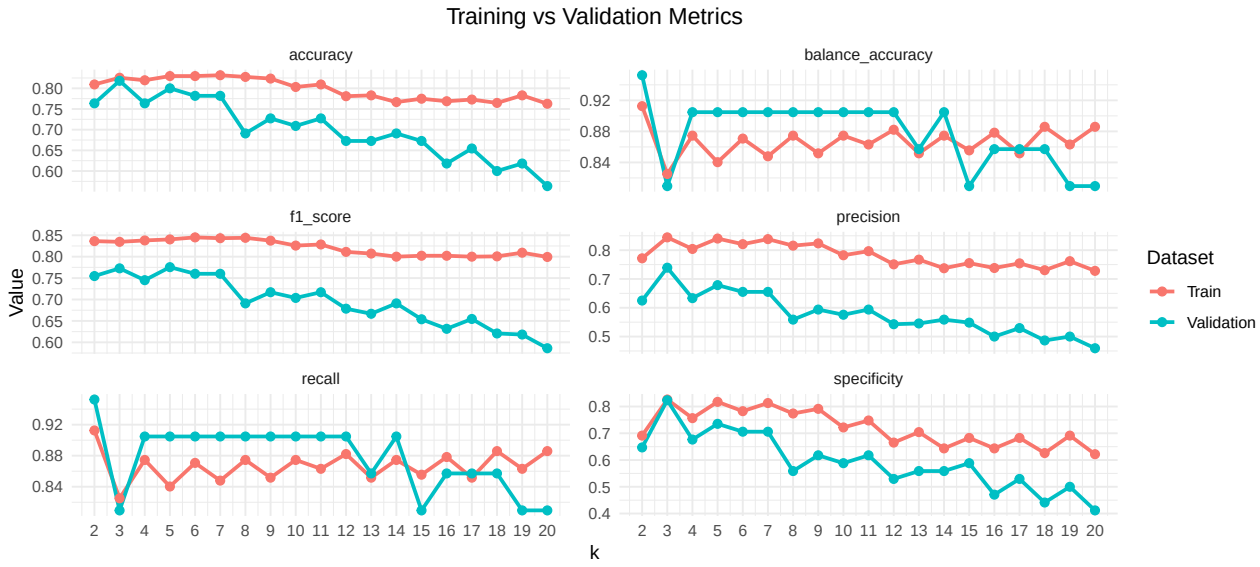
```r
# select the matrix for evaluating the knn model
valid_idx <- strtoi(row.names(valid_df))
valid_simi_matrix <- simi_matrix[valid_idx, train_idx]
dim(valid_simi_matrix)
```

```
## [1]  55 493
```

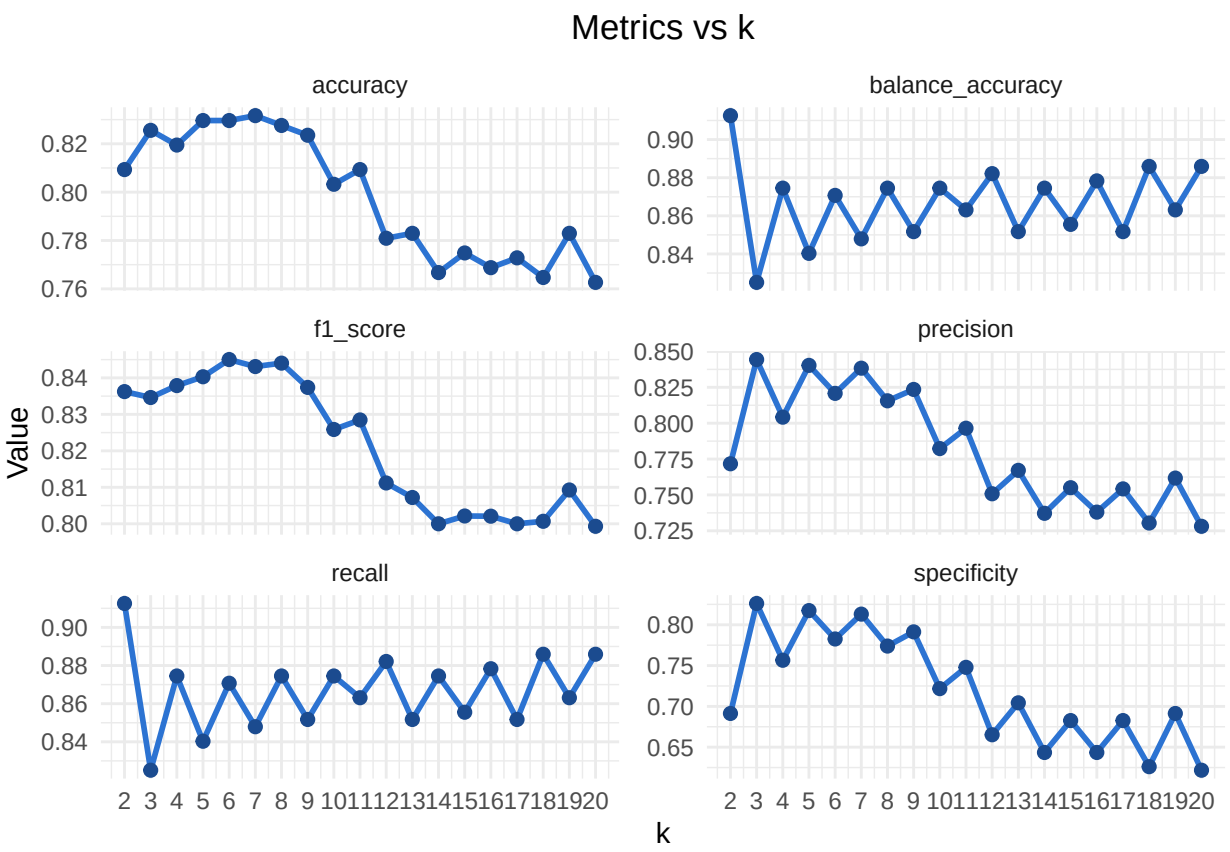
```r
df_result <- data.frame(
  "k" = integer(),
  "accuracy" = numeric(),
  "recall" = numeric(),
  "precision" = numeric(),
  "specificity" = numeric(),
  "balance_accuracy" = numeric(),
  "f1_score" = numeric()
)

for (k in 2:20){
  confusion_matrix <- knn_train(train=train_df, simi_matrix=train_simi_matrix, k=k)
  metrics = class_metrics(confusion_matrix)
  df_result <- rbind(
    df_result,
    data.frame(
         k = k,
         accuracy = metrics$accuracy,
         recall = metrics$recall,
         precision = metrics$precision,
         specificity = metrics$specificity,
         balance_accuracy = metrics$balance_accuracy,
         f1_score = metrics$f1_score
       )
    )
}

# Validate the model
df_result_valid <- data.frame(
  "k" = integer(),
  "accuracy" = numeric(),
  "recall" = numeric(),
  "precision" = numeric(),
  "specificity" = numeric(),
  "balance_accuracy" = numeric(),
  "f1_score" = numeric()
)

for (k in 2:20){
```

```
  confusion_matrix <- knn_eval(
    train=train_df,
    test = valid_df,
    simi_matrix = valid_simi_matrix,
    k=k
    )
  metrics = class_metrics(confusion_matrix)
  df_result_valid <- rbind(
    df_result_valid,
    data.frame(
         k = k,
         accuracy = metrics$accuracy,
         recall = metrics$recall,
         precision = metrics$precision,
         specificity = metrics$specificity,
         balance_accuracy = metrics$balance_accuracy,
         f1_score = metrics$f1_score
       )
    )
}
```

```
library(tidyverse)

df_train_long <- df_result |>
  pivot_longer(cols = -k, names_to = "metric", values_to = "value") |>
  mutate(partition = "Train")

df_valid_long <- df_result_valid |>
  pivot_longer(cols = -k, names_to = "metric", values_to = "value") |>
  mutate(partition = "Validation")

df_both <- bind_rows(df_train_long, df_valid_long)

ggplot(df_both, aes(k, value, color = partition)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  facet_wrap(~ metric, scales = "free_y", ncol = 2) +
  scale_x_continuous(breaks = unique(df_both$k)) +
  theme_minimal() +
  labs(
    title = "Training vs Validation Metrics",
    x = "k",
    y = "Value",
    color = "Dataset"
  ) +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
library(tidyverse)

#pivot the table
df_long <- df_result |>
  pivot_longer(cols = -k, names_to = "metric", values_to = "value")

# plot
ggplot(df_long, aes(k, value)) +
  geom_line(color = "#2C73D2", size = 1) +
  geom_point(color = "#1B4B8F", size = 2) +
  # create subfig for each metric 2 columns, 3 rows
  facet_wrap(~ metric, scales = "free_y", ncol = 2) +
  scale_x_continuous(breaks = unique(df_long$k)) +
  theme_minimal() +
  labs(
    title = "Metrics vs k",
    x = "k",
    y = "Value"
  ) +
  # align the title
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



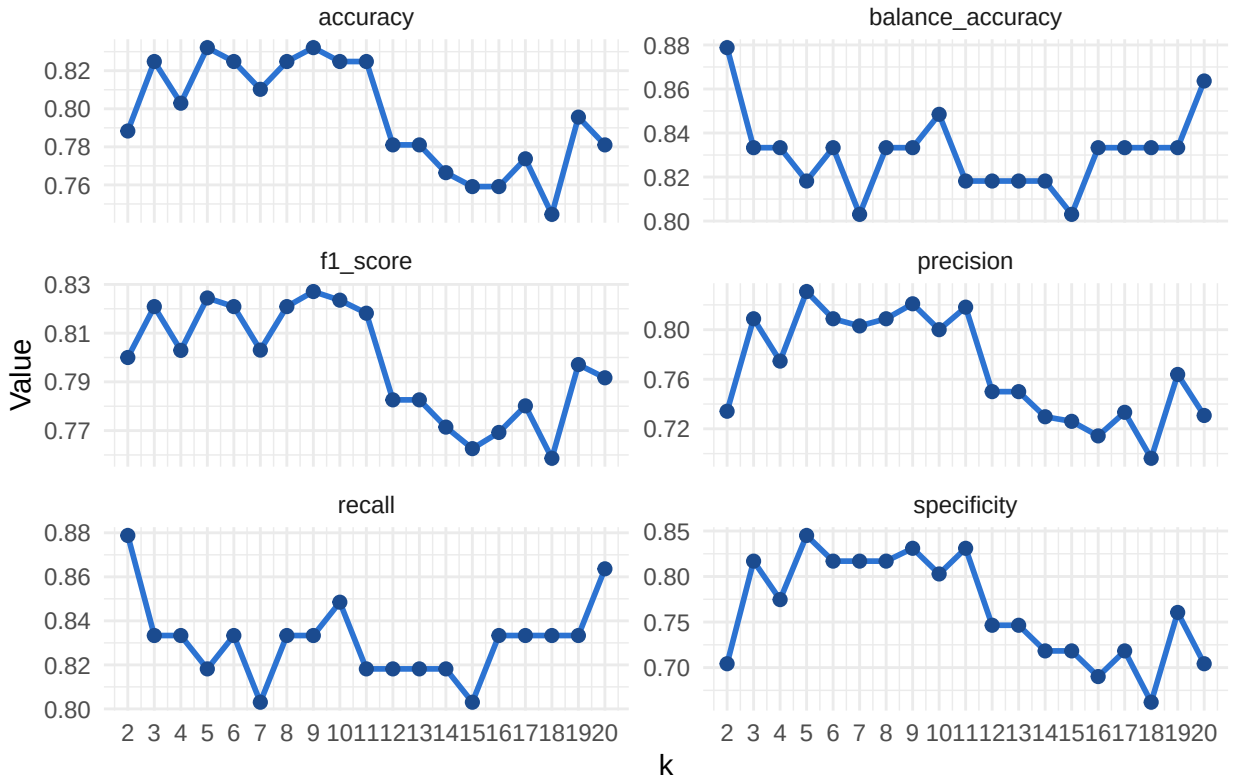
3.3 Evaluating the performance of the model on the test set

```
# select the matrix for evaluating the knn model
subtest_idx <- strtoi(row.names(test_df))
subtest_simi_matrix <- simi_matrix[subtest_idx, subtraining_idx]
```

```
# Evaluate the model
df_result_test <- data.frame(
  "k" = integer(),
  "accuracy" = numeric(),
  "recall" = numeric(),
  "precision" = numeric(),
  "specificity" = numeric(),
  "balance_accuracy" = numeric(),
  "f1_score" = numeric()
)

for (k in 2:20){
  confusion_matrix <- knn_eval(
    train=training_df,
    test = test_df,
    simi_matrix=subtest_simi_matrix,
    k=k
    )
```

```
  metrics = class_metrics(confusion_matrix)
  df_result_test <- rbind(
    df_result_test,
    data.frame(
         k = k,
         accuracy = metrics$accuracy,
         recall = metrics$recall,
         precision = metrics$precision,
         specificity = metrics$specificity,
         balance_accuracy = metrics$balance_accuracy,
         f1_score = metrics$f1_score
       )
    )
}
```

```
library(tidyverse)

#pivot the table
df_long <- df_result_test |>
  pivot_longer(cols = -k, names_to = "metric", values_to = "value")

# plot
ggplot(df_long, aes(k, value)) +
  geom_line(color = "#2C73D2", size = 1) +
  geom_point(color = "#1B4B8F", size = 2) +
  facet_wrap(~ metric, scales = "free_y", ncol = 2) +   # 2 columns, 3 rows
  scale_x_continuous(breaks = unique(df_long$k)) +
  theme_minimal() +
  labs(
    title = "Metrics vs k",
    x = "k",
    y = "Value"
  ) +
  # align the title
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```




```
print(df_result[2,])
```

```
##   k  accuracy    recall precision specificity balance_accuracy  f1_score
## 2 3 0.8255578 0.8250951  0.844358    0.826087        0.8250951 0.8346154
```

```
print(df_result_valid[2,])
```

```
##   k  accuracy    recall precision specificity balance_accuracy  f1_score
## 2 3 0.8181818 0.8095238 0.7391304   0.8235294        0.8095238 0.7727273
```

```
print(df_result_test[2,])
```

```
##   k  accuracy    recall precision specificity balance_accuracy  f1_score
## 2 3 0.8248175 0.8333333 0.8088235   0.8169014        0.8333333 0.8208955
```