Rivera\_Gabi\_Assignment6.1

Gabi Rivera

library(tidyverse)  
library(gt)  
library(caret)  
library(e1071)  
library(AppliedPredictiveModeling)  
library(pROC)  
library(e1071)

## Assignment 6

### Question 1

The table below lists a dataset that was used to create a nearest neighbor model that predicts whether observations belong to class A or B.

# Data 6.1.a  
data61a <- tibble::tibble(  
 X1 = 1:6,  
 X2 = 2:7,  
 X3 = seq(30, 80, by = 10),)   
class61a <- c("A", "B", "A", "B", "B", "A")  
gt(data61a)

| X1 | X2 | X3 |
| --- | --- | --- |
| 1 | 2 | 30 |
| 2 | 3 | 40 |
| 3 | 4 | 50 |
| 4 | 5 | 60 |
| 5 | 6 | 70 |
| 6 | 7 | 80 |

Assuming that the model uses Euclidean distance to find the nearest neighbor, what prediction will the model return for each of the following query instances.

# data 6.1.b  
data61b <- tibble::tribble(  
 ~X1, ~X2, ~X3,  
 9, 4, 15,  
 7, 8, 45,  
 8, 7, 75)  
gt(data61b)

| X1 | X2 | X3 |
| --- | --- | --- |
| 9 | 4 | 15 |
| 7 | 8 | 45 |
| 8 | 7 | 75 |

a. Perform the analysis WITHOUT doing any preprocessing of the data.

ctrl <- trainControl(method = "cv",  
 number = 5,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE,  
 savePredictions = TRUE)  
  
# Create KNN model without preprocessing  
set.seed(123)  
knnFit\_wo <- train(x = data61a,  
 y = class61a,  
 method = "knn",  
 metric = "ROC",  
 tuneLength = 10,  
 trControl = ctrl)  
knnFit\_wo

k-Nearest Neighbors   
  
6 samples  
3 predictors  
2 classes: 'A', 'B'   
  
No pre-processing  
Resampling: Cross-Validated (5 fold)   
Summary of sample sizes: 4, 5, 5, 5, 5   
Resampling results across tuning parameters:  
  
 k ROC Sens Spec   
 5 0.5 0.3333333 0.0000000  
 7 0.5 0.0000000 0.0000000  
 9 0.5 0.3333333 0.0000000  
 11 0.5 0.3333333 0.0000000  
 13 0.5 0.3333333 0.0000000  
 15 0.5 0.0000000 0.0000000  
 17 0.5 0.3333333 0.3333333  
 19 0.5 0.3333333 0.3333333  
 21 0.5 0.0000000 0.3333333  
 23 0.5 0.3333333 0.3333333  
  
ROC was used to select the optimal model using the largest value.  
The final value used for the model was k = 23.

# Predict class using data61b  
knnFit\_wo\_predictions <- predict(knnFit\_wo, newdata = data61b, type = "raw")  
knnFit\_wo\_predictions <- data.frame(class = knnFit\_wo\_predictions)  
knnFit\_wo\_predictions <- cbind(data61b, knnFit\_wo\_predictions)   
knnFit\_wo\_predictions |> gt() |>   
 tab\_header(title = "KNN model test result without preprocessing")

Table 1: KNN model test result without preprocessing

| X1 | X2 | X3 | class |
| --- | --- | --- | --- |
| 9 | 4 | 15 | A |
| 7 | 8 | 45 | B |
| 8 | 7 | 75 | B |

b. Perform the analysis WITH doing any preprocessing of the data.

# No missing data  
# All predictors are highly correlated to the outcome  
correlation\_matrix <- cor(data61a)  
# Center and scale  
data61a\_scaled <- scale(data61a)  
  
# Create KNN model with preprocessing  
set.seed(123)  
knnFit\_w <- train(x = data61a\_scaled,  
 y = class61a,  
 method = "knn",  
 tuneLength = 10,  
 metric = "ROC",  
 trControl = ctrl)  
knnFit\_w

k-Nearest Neighbors   
  
6 samples  
3 predictors  
2 classes: 'A', 'B'   
  
No pre-processing  
Resampling: Cross-Validated (5 fold)   
Summary of sample sizes: 4, 5, 5, 5, 5   
Resampling results across tuning parameters:  
  
 k ROC Sens Spec   
 5 0.5 0.3333333 0.0000000  
 7 0.5 0.0000000 0.0000000  
 9 0.5 0.3333333 0.0000000  
 11 0.5 0.3333333 0.0000000  
 13 0.5 0.3333333 0.0000000  
 15 0.5 0.0000000 0.0000000  
 17 0.5 0.3333333 0.3333333  
 19 0.5 0.3333333 0.3333333  
 21 0.5 0.0000000 0.3333333  
 23 0.5 0.3333333 0.3333333  
  
ROC was used to select the optimal model using the largest value.  
The final value used for the model was k = 23.

# Predict class using data61b  
knnFit\_w\_predictions <- predict(knnFit\_w, newdata = data61b, type = "raw")  
knnFit\_w\_predictions <- data.frame(class = knnFit\_w\_predictions)  
knnFit\_w\_predictions <- cbind(data61b, knnFit\_w\_predictions)   
knnFit\_w\_predictions |> gt() |>   
 tab\_header(title = "KNN model test result with preprocessing")

Table 2: KNN model test result with preprocessing

| X1 | X2 | X3 | class |
| --- | --- | --- | --- |
| 9 | 4 | 15 | A |
| 7 | 8 | 45 | B |
| 8 | 7 | 75 | B |

c. Do you observe any differences between the predictions with and without preprocessing?

*No, the results are exactly the same using the small sample size.*

### Question 2

Email spam filtering models often use a bag-of-words representation for emails. In a bag-of-words representation, the descriptive features that describe a document (in our case, an email) each represent how many times a particular word occurs in the document. One descriptive feature is included for each word in a predefined dictionary. The dictionary is typically defined as the complete set of words that occur in the training dataset. The table below lists the bag-of-words representation for the following five emails and a target feature, SPAM, whether they are spam emails or genuine emails:

*“money, money, money”*

*“free money for free gambling fun”*

*“gambling for fun”*

*“machine learning for fun, fun, fun”*

*“free machine learning”*

Bag\_of\_words <- tibble::tibble(  
 Money = c(3,1,0,0,0),  
 Free = c(0,2,0,0,1),  
 For = c(0,1,1,1,0),  
 Gambling = c(0,1,1,0,0),  
 Fun = c(0,1,1,3,0),  
 Machine = c(0,0,0,1,1),  
 Learning = c(0,0,0,1,1))  
   
Spam <- c("True", "True", "True", "False", "False")  
gt(Bag\_of\_words)

| Money | Free | For | Gambling | Fun | Machine | Learning |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 3 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |

a. What target level would a nearest neighbor model using Euclidean distance return for the following email: “machine learning for free”?

# Return for "machine learning for free" email  
target\_level <- c(Money = 0, Free = 1, For = 1, Gambling = 0, Fun = 0,   
 Machine = 1, Learning = 1)  
  
bow\_list <- split(Bag\_of\_words, seq(nrow(Bag\_of\_words)))  
bow\_list <- lapply(bow\_list, function(row) as.numeric(unlist(row)))  
  
# Calcualte KNN using Euclidean distance  
euclid\_dist <- function(x, y) sqrt(sum((x - y)^2))  
distances <- sapply(bow\_list, function(x) euclid\_dist(x, target\_level))  
  
min\_index <- which.min(distances)  
closest\_spam <- Spam[min\_index]  
cat("Spam prediction for machine learning for free = ", closest\_spam)

**Spam prediction for machine learning for free is False**

# Table Euclidean distance   
distances <- data.frame(Euclidean\_distance = round(distances,4))  
distances |> gt()

| Euclidean\_distance |
| --- |
| 3.6056 |
| 2.4495 |
| 2.2361 |
| 3.1623 |
| 1.0000 |

b. What target level would a k-NN model with k = 3 and using Euclidean distance return for the same query?

# KNN model with K=3 using Euclidean distance  
k <- 3  
nearest\_indices <- order(distances)[1:k]  
nearest\_labels <- Spam[nearest\_indices]  
  
# Count occurrences of "True" (spam) and "False" (not spam)  
count\_true <- sum(nearest\_labels == "True")  
count\_false <- sum(nearest\_labels == "False")  
  
# Determine the majority label  
if (count\_true > count\_false) {target\_label <- "True"} else {  
 target\_label <- "False"}  
  
cat("Spam prediction at KNN K=3 is", target\_label)

**Spam prediction at KNN K=3 is True**

c. What target level would a *k*-NN model with *k* = 3 and using Manhattan distance return for the same query?

# Calculate using Manhattan distance   
manhattan\_dist <- function(x, y) sum(abs(x - y))  
distances\_m <- sapply(bow\_list, function(x) manhattan\_dist(x, target\_level))  
  
# KNN model with K=3  
nearest\_indices\_m <- order(distances\_m)[1:k]  
nearest\_labels\_m <- Spam[nearest\_indices\_m]  
  
# Count occurrences of "True" (spam) and "False" (not spam)  
count\_true\_m <- sum(nearest\_labels\_m == "True")  
count\_false\_m <- sum(nearest\_labels\_m == "False")  
  
# Determine the majority label  
if (count\_true\_m > count\_false\_m) {target\_label\_m <- "True"} else {  
 target\_label\_m <- "False"}  
  
cat("Spam prediction at KNN K= 3 using Manhattan distribution is",   
 target\_label\_m)

**Spam prediction at KNN K= 3 using Manhattan distribution is False**

d. There are a lot of zero entries in the spam bag-of-words dataset. This is indicative of sparse data and is typical for text analytics. Cosine similarity is often a good choice when dealing with sparse non-binary data. What target level would a 3-NN model using cosine similarity return for the query?

# Normalize the list  
bow\_list\_cos <- lapply(split(as.matrix(Bag\_of\_words), seq(nrow(Bag\_of\_words))),   
 function(row) row / sqrt(sum(row^2)))  
target\_point\_cos <- target\_level / sqrt(sum(target\_level^2))  
  
# Calculate cosine similarity  
cosine\_similarity <- function(x, y) sum(x \* y)  
similarities <- sapply(bow\_list\_cos,   
 function(x) cosine\_similarity(x, target\_point\_cos))  
  
nearest\_indices\_cos <- order(similarities, decreasing = TRUE)[1:k]  
nearest\_labels\_cos <- Spam[nearest\_indices\_cos]  
  
# Count occurrences of "True" (spam) and "False" (not spam)  
count\_true\_cos <- sum(nearest\_labels\_cos == "True")  
count\_false\_cos <- sum(nearest\_labels\_cos == "False")  
  
# Determine the majority label  
if (count\_true\_cos > count\_false\_cos) {target\_label\_cos <- "True"} else {  
 target\_label\_cos <- "False"}  
  
cat("Spam prediction at KNN K= 3 using cosine similarity is",   
 target\_label\_cos)

**Spam prediction at KNN K= 3 using cosine similarity is False**

### Question 3

*Predictive data analytics models are often used as tools for process quality control and fault detection. The task in this question is to create a naive Bayes model to monitor a waste water treatment plant.1 The table below lists a dataset containing details of activities at a waste water treatment plant for 14 days. Each day is described in terms of six descriptive features that are generated from different sensors at the plant. SS-IN measures the solids coming into the plant per day; SED-IN measures the sediment coming into the plant per day; COND-IN measures the electrical conductivity of the water coming into the plant.2 The features SS-OUT, SED-OUT, and CONDOUT are the corresponding measurements for the water flowing out of the plant. The target feature, STATUS, reports the current situation at the plant: ok, everything is working correctly; settler, there is a problem with the plant settler equipment; or solids, there is a problem with the amount of solids going through the plant.*

waste\_water <- tibble::tibble(  
 Ss\_In = c(168,156,176,256,230,116,242,242,174,1004,1228,964,2008),  
 Sed\_In = c(3,3,3.5,3,5,3,7,4.5,2.5,35,46,17,32),  
 Cond\_In = c(1814,1358,2200,2070,1410,1238,1315,1183,1110,1218,1889,2120,1257),  
 Ss\_Out = c(15,14,16,27,131,104,104,78,73,81,82.4,20,13),  
 Sed\_Out = c(.001,.01,.005,.2,3.5,.06,.01,.02,1.5,1172,1932,1030,1038),  
 Cond\_Out = c(1879,1425,2140,2700,1575,1221,1434,1374,1256,33.3,43.1,1966,1289),  
 Status = c("ok", "ok", "ok", "ok", "settler", "settler", "settler", "settler", "settler",  
 "solid", "solid", "solid", "solid"))  
gt(waste\_water)

| Ss\_In | Sed\_In | Cond\_In | Ss\_Out | Sed\_Out | Cond\_Out | Status |
| --- | --- | --- | --- | --- | --- | --- |
| 168 | 3.0 | 1814 | 15.0 | 0.001 | 1879.0 | ok |
| 156 | 3.0 | 1358 | 14.0 | 0.010 | 1425.0 | ok |
| 176 | 3.5 | 2200 | 16.0 | 0.005 | 2140.0 | ok |
| 256 | 3.0 | 2070 | 27.0 | 0.200 | 2700.0 | ok |
| 230 | 5.0 | 1410 | 131.0 | 3.500 | 1575.0 | settler |
| 116 | 3.0 | 1238 | 104.0 | 0.060 | 1221.0 | settler |
| 242 | 7.0 | 1315 | 104.0 | 0.010 | 1434.0 | settler |
| 242 | 4.5 | 1183 | 78.0 | 0.020 | 1374.0 | settler |
| 174 | 2.5 | 1110 | 73.0 | 1.500 | 1256.0 | settler |
| 1004 | 35.0 | 1218 | 81.0 | 1172.000 | 33.3 | solid |
| 1228 | 46.0 | 1889 | 82.4 | 1932.000 | 43.1 | solid |
| 964 | 17.0 | 2120 | 20.0 | 1030.000 | 1966.0 | solid |
| 2008 | 32.0 | 1257 | 13.0 | 1038.000 | 1289.0 | solid |

a. Create a naive Bayes model that uses probability density functions to model the descriptive features in this dataset (assume that all the descriptive features are normally distributed). Show clearly the distribution of each of the predictors conditional on status and the prior probabilities of the status. (15 points)

# Create a naive Bayes model that uses PDF  
waste\_water\_df <- as.data.frame(waste\_water)  
ctrl1 <- trainControl(method = "cv",  
 number = 5,  
 summaryFunction = defaultSummary,  
 classProbs = TRUE,  
 savePredictions = TRUE)  
nb\_model <- train(Status ~ .,   
 data = waste\_water\_df,   
 method = "naive\_bayes",   
 trControl = ctrl1)  
  
# Retrive means and standard deviations by Status   
round(nb\_model$finalModel$table$Ss\_In,2)

**Ss\_In means and standard deviations**   
Ss\_In ok settler solid  
 mean 189.00 200.80 1301.00  
 sd 45.42 55.13 485.44

round(nb\_model$finalModel$table$Sed\_In,2)

**Sed\_In means and standard deviations**   
Sed\_In ok settler solid  
 mean 3.12 4.40 32.50  
 sd 0.25 1.78 11.96

round(nb\_model$finalModel$table$Cond\_In,2)

**Cond\_In means and standard deviations**   
Cond\_In ok settler solid  
 mean 1860.50 1251.20 1621.00  
 sd 371.40 116.24 453.04

round(nb\_model$finalModel$table$Ss\_Out,2)

**Ss\_Out means and standard deviations**   
Ss\_Out ok settler solid  
 mean 18.00 98.00 49.10  
 sd 6.06 23.38 37.76

round(nb\_model$finalModel$table$Sed\_Out,2)

**Sed\_Out means and standard deviations**   
Sed\_Out ok settler solid  
 mean 0.05 1.02 1293.00  
 sd 0.10 1.53 430.95

round(nb\_model$finalModel$table$Cond\_Out,2)

**Cond\_Out means and standard deviations**   
Cond\_Out ok settler solid  
 mean 2036.00 1372.00 832.85  
 sd 532.19 142.58 958.31

# Calculate prior probabilities  
prior\_prob <- waste\_water |>  
 count(Status) |>  
 mutate(prior\_probability = n / sum(n)) |>  
 select(Status, prior\_probability)  
prior\_prob <- as.data.frame(t(prior\_prob))  
colnames(prior\_prob) <- prior\_prob[1, ]  
prior\_prob <- prior\_prob[-1,]  
prior\_prob

**Prior probabilities**

ok settler solid  
prior\_probability 0.3077 0.3846 0.3077

b. What prediction will the naive Bayes model return for the following query? SS-IN = 222, SED-IN = 4.5, COND-IN = 1518, SS-OUT = 74, SED-OUT = 0.25, COND-OUT = 1642.

# Naive Bayes model prediction for the query below  
new\_data <- data.frame(Ss\_In = 222, Sed\_In = 4.5,  
 Cond\_In = 1518, Ss\_Out = 74,  
 Sed\_Out = 0.25, Cond\_Out = 1642)  
  
predictions <- predict(nb\_model, newdata = new\_data)  
cat("Naive Bayes model predition for the query is", predictions, "or settler.")

**Naive Bayes model predition for the query is settler.**

### Question 4

Imagine that you have been given a dataset of 1,00 documents that have been classified as being about entertainment or education. There are 700 entertainment documents in the dataset and 300 education documents in the dataset. The tables below give the number of documents from each topic that a selection of worlds occurred in.

entertainment <- tibble::tibble(fun = 415, is = 695,   
 machine = 35, christmas = 0,   
 family = 400, learning = 70)  
entertainment |> gt() |> tab\_header("Word-document counts for the entertainment dataset")

Table 3: Word-document counts for the entertainment dataset

| fun | is | machine | christmas | family | learning |
| --- | --- | --- | --- | --- | --- |
| 415 | 695 | 35 | 0 | 400 | 70 |

education <- tibble::tibble(fun = 200, is = 295,   
 machine = 120, christmas = 0,   
 family = 10, learning = 105)  
education |> gt() |> tab\_header("Word-document counts for the education dataset")

Table 4: Word-document counts for the education dataset

| fun | is | machine | christmas | family | learning |
| --- | --- | --- | --- | --- | --- |
| 200 | 295 | 120 | 0 | 10 | 105 |

1. What target level will a Naive Bayes model predict for the following query document: “machine learning is fun”?

**a.1. Conditional probabilities for Entertainment:**

P(machine|entertainment) = 35/700 = 0.050

P(learning|entertainment) = 70/700 = 0.100

P(is|entertainment) = 695/700 = 0.993

P(fun|entertainment) = 415/700 = 0.593

**a.2. Conditional probabilities for Education:**

P(machine|education) = 120/300 = 0.400

P(learning|education) = 105/300 = 0.350

P(is|education) = 295/300 = 0.983

P(fun|education) = 200/300 = 0.667

**a.3. Class priors**

P(entertainment) = 700/1000 = 0.7

P(education) = 300/1000 = 0.3

**a.4. Naïve Bayes calculation**

P(query∣entertainment)=P("machine"∣entertainment)×P("learning"∣entertainment)×P("is"∣entertainment)×P("fun"∣entertainment)×P(entertainment)

P(query∣entertainment)= 0.05×0.1×0.993×0.593×0.7 = 0.002

P(query∣education)=P("machine"∣education)×P("learning"∣education)×P("is"∣education)×P("fun"∣education)×P(education)

P(query∣education)= 0.4×0.35×0.983×0.667×0.3 = 0.03

**Machine learning is fun in this case is from the education dataset.**

1. What target level will a Naive Bayes model predict for the following query document: “christmas family fun”?

**b.1** **Conditional probabilities for Entertainment:**

P(christmas|entertainment) = 0/700 = 0

P(family|entertainment) = 400/700 = 0.571

P(fun|entertainment) = 415/700 = 0.593

**b.2** **Conditional probabilities for Education:**

P(christmas|education) = 0/300 = 0

P(family|education) = 10/300 = 0.033

P(fun|education) = 200/300 = 0.667

**b.3. Naïve Bayes calculation**

P(query∣entertainment)= 0.571×0.593×0.7 = 0.237

P(query∣education)= 0.033×0.667×0.3 = 0.007

**Christmas family fun is from the entertainment dataset.**

1. What target level will a Naive Bayes model predict for the following query document: “learning is family fun”?

**c.1. Conditional probabilities for Entertainment:**

P(learning|entertainment) = 70/700 = 0.100

P(is|entertainment) = 695/700 = 0.993

P(family|entertainment) = 400/700 = 0.571

P(fun|entertainment) = 415/700 = 0.593

**c.2. Conditional probabilities for Education:**

P(learning|education) = 105/300 = 0.350

P(is|education) = 295/300 = 0.983

P(family|education) = 10/300 = 0.033

P(fun|education) = 200/300 = 0.667

**c.3. Naïve Bayes calculation**

P(query∣entertainment)= 0.100×0.993×0.571×0.593×0.7 = 0.024

P(query∣education)= 0.350×0.983×0.033×0.667×0.3 = 0.002

**Learning is family fun is from the entertainment dataset.**