MATH 3190 Homework 8

Supervised and Unsupervised Machine Learning

Due 4/26/2022

This homework will focus on unsupervised clustering and dimension reduction, support vector machines, random forests, and neural networks. Please complete your work in Rmarkdown (.html output) and post your work to your GitHub page.

The TBnanostring.rds dataset contains gene expression measurements in the blood for 107 TB-related genes for 179 patients with either active tuberculosis infection (TB) or latent TB infection (LTBI). Load these data into R ({TBnanostring <- readRDS(`TBnanostring.rds')}). The TB status is found in the first column of the data frame, followed by the genes in the subsequent columns, and the rows represent each individual patient. Complete the following analyses:

(20 points) Apply a UMAP clustering of the dataset, and plot the result using {ggplot}. Color the points based on TB status. How does UMAP preform in clustering these samples?

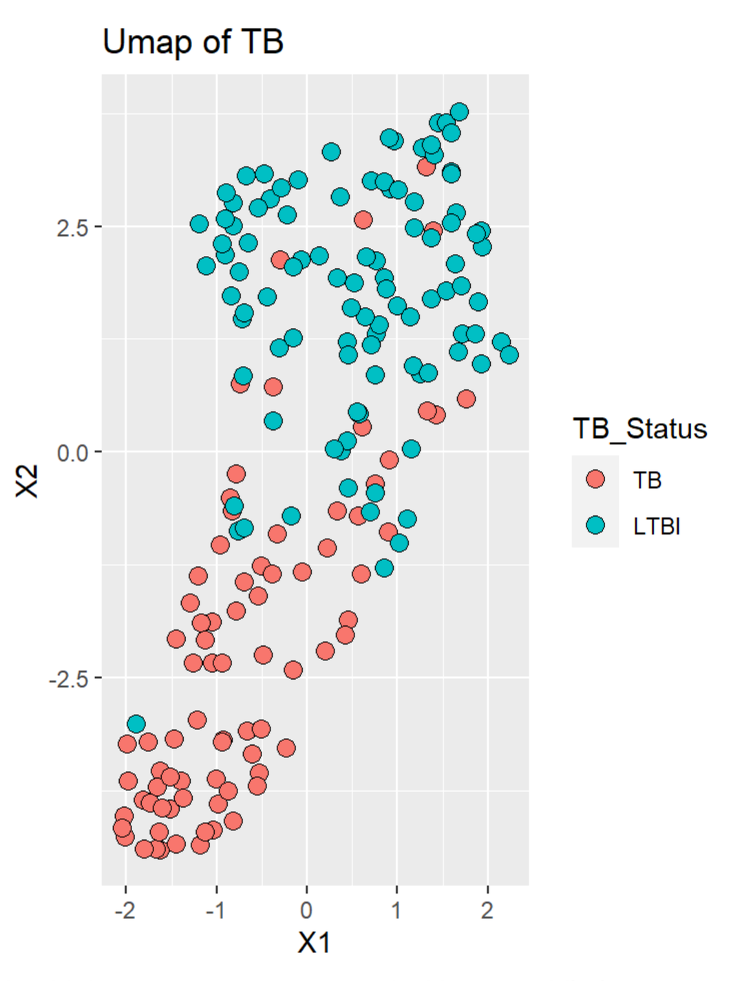
```
setwd("C:\\Users\\Owner\\Documents\\Math_3190")
TB <- readRDS("TBnanostring.rds")
library(umap)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
umap_TB <- umap(TB[,2:ncol(TB)])
data.frame(umap_TB$layout, TB_Status = TB$TB_Status) %>%
      ggplot(aes(X1,X2, fill = TB_Status))+
      geom_point(cex=3, pch=21) +
      coord_fixed(ratio = 1)+
      labs(title = "Umap of TB")
```



UMAP does pretty well at clusering

here based on X2. X2 being higher seems to correspond with LTBI while lower X2 corresponds with TBI

(20 points) Use hierarchical clustering for classification and exploration of this dataset. Do the following:

Make a dendrogram of the hierarchical clustering result. Color the sample names based on TB status (see example code embedded in this RMarkdown)

```
hc <- hclust(dist(TB))
```

```
## Warning in dist(TB): NAs introduced by coercion
```

```
## make a dendrogram object, color the labels, make them smaller using the dendextend package
library(dendextend)
```

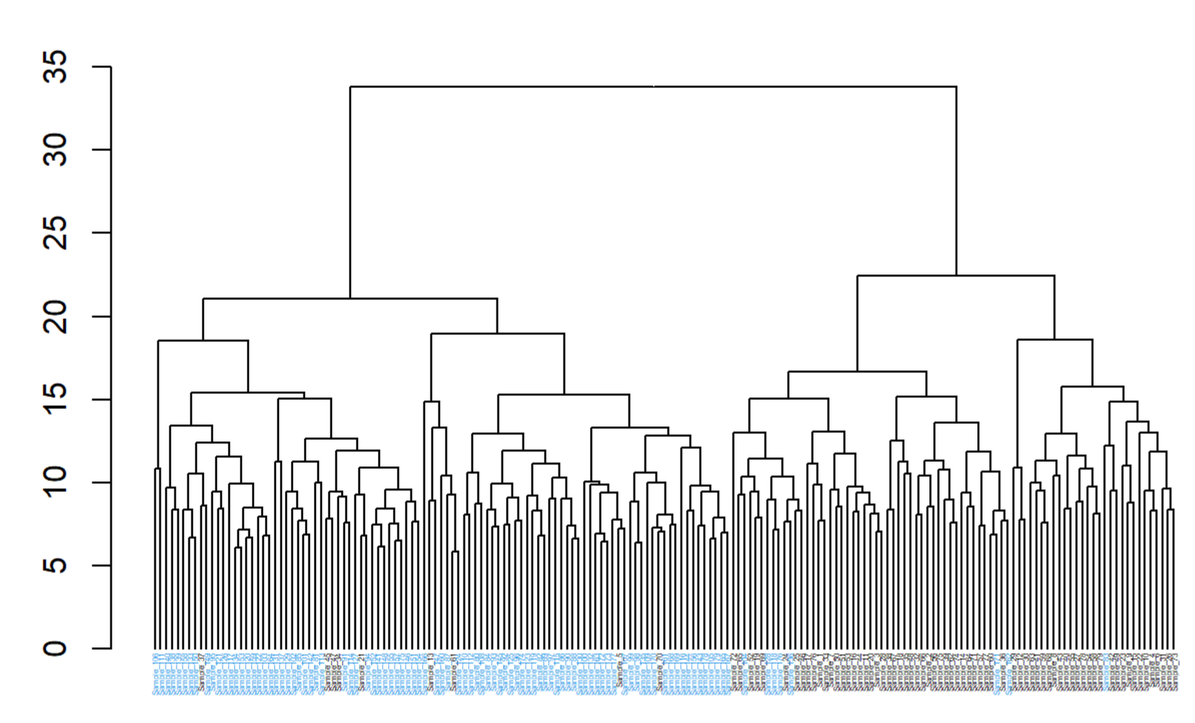
```
##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
dend <- as.dendrogram(hc)
dend <- set(dend, "labels_cex", .25)
temp_TB <- TB[rownames(TB)[order.dendrogram(dend)],1]
labels_colors(dend) <- ifelse(temp_TB == "TB",1,4)

## plot the dendrogram
plot(dend)
```



Cut the tree into 2 clusters. What are the proportions of TB/LTBI samples in each cluster?

```
groups <- cutree(dend, k = 2)

names((groups)[groups==1])
```

```
##  [1] "Sample_1"   "Sample_2"   "Sample_3"   "Sample_4"   "Sample_6"
##  [6] "Sample_7"   "Sample_8"   "Sample_9"   "Sample_10"  "Sample_11"
## [11] "Sample_12"  "Sample_14"  "Sample_15"  "Sample_16"  "Sample_17"
## [16] "Sample_18"  "Sample_19"  "Sample_20"  "Sample_22"  "Sample_23"
## [21] "Sample_24"  "Sample_25"  "Sample_26"  "Sample_27"  "Sample_28"
## [26] "Sample_29"  "Sample_30"  "Sample_31"  "Sample_32"  "Sample_33"
## [31] "Sample_35"  "Sample_36"  "Sample_38"  "Sample_39"  "Sample_40"
## [36] "Sample_41"  "Sample_42"  "Sample_43"  "Sample_44"  "Sample_46"
## [41] "Sample_48"  "Sample_49"  "Sample_50"  "Sample_51"  "Sample_52"
## [46] "Sample_53"  "Sample_54"  "Sample_55"  "Sample_56"  "Sample_57"
## [51] "Sample_58"  "Sample_59"  "Sample_60"  "Sample_62"  "Sample_63"
## [56] "Sample_64"  "Sample_65"  "Sample_66"  "Sample_67"  "Sample_68"
## [61] "Sample_69"  "Sample_71"  "Sample_72"  "Sample_73"  "Sample_74"
## [66] "Sample_75"  "Sample_76"  "Sample_77"  "Sample_78"  "Sample_79"
## [71] "Sample_83"  "Sample_104" "Sample_118" "Sample_128" "Sample_136"
## [76] "Sample_162" "Sample_171" "Sample_176"
```

```
group1 <- TB[row.names(TB) %in% names((groups)[groups==1]),]

group2 <- TB[row.names(TB) %in% names((groups)[groups==2]),]

sum(group1$TB_Status == 'TB')/nrow(group1)
```

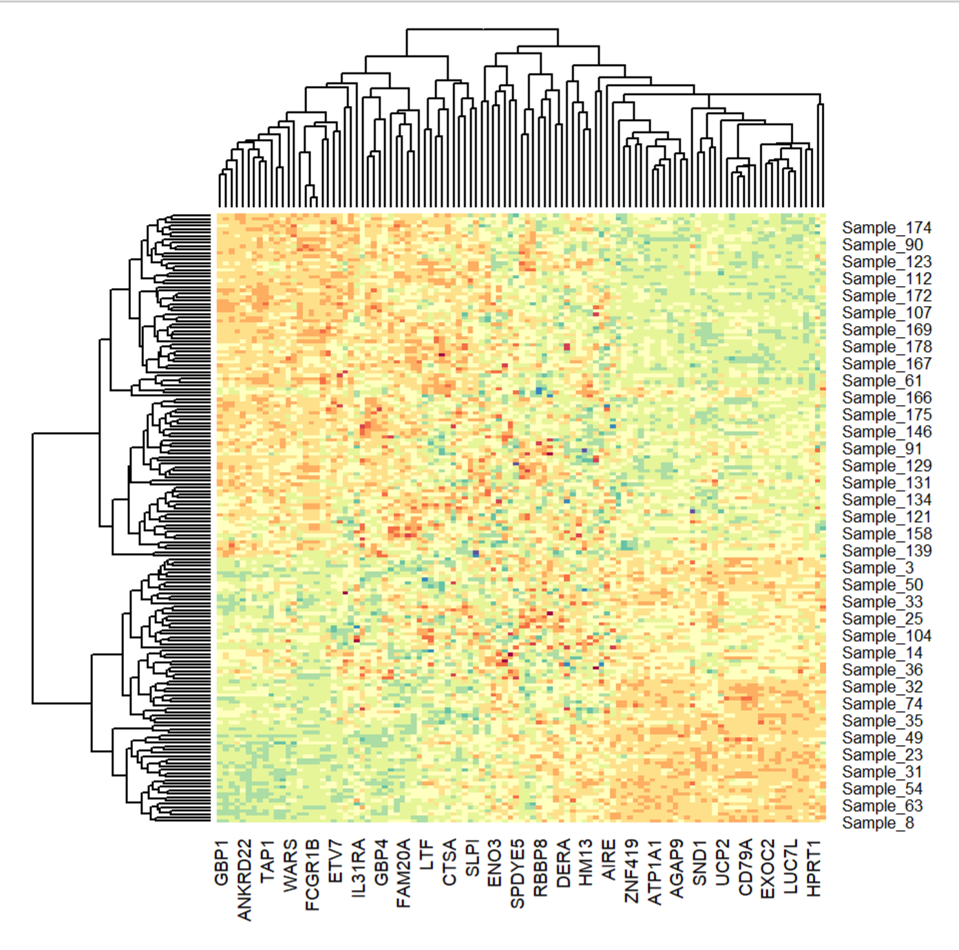
```
## [1] 0.8974359
```

```
sum(group2$TB_Status == 'TB')/nrow(group2)
```

```
## [1] 0.08910891
```

The first group has about 89.7% with TB, while the second group has about 8.9% with TB. Make a clustered heatmap of these data (see the lecture slides).

```
heatmap(as.matrix(TB[,-1]), col = RColorBrewer::brewer.pal(11, "Spectral"))
```



(40 points) Split the dataset into “training” and “testing” sets using a 70/30 partition, using {set.seed(0)} and the {createDataPartition} function from the {caret} package (code for this is `hiding' in this .Rmd file!). Apply the following machine learning methods to make a predictive biomarker to distinguish between the TB and control samples. Use the {caret} package and cross validation to find the “finalModel” parameters to for each method.

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

Support Vector Machine (try linear, radial, and polynomial kernels)

```
### Linear SVM
svm_Linear <- train(TB_Status ~., data = training, method = "svmLinear",
                 tuneLength = 10)

test_pred_lin <- predict(svm_Linear, newdata = testing)

confusionMatrix(test_pred_lin, testing$TB_Status)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8113208
```

```
### Radial SVM
svm_Radial <- train(TB_Status ~., data = training, method = "svmRadial",
                 tuneLength = 10)

test_pred_rad <- predict(svm_Radial, newdata = testing)

confusionMatrix(test_pred_rad, testing$TB_Status)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8679245
```

```
### Ploynomial SVM
svm_Poly <- train(TB_Status ~., data = training, method = "svmPoly",
                 tuneLength = 10)

test_pred_poly <- predict(svm_Poly, newdata = testing)

confusionMatrix(test_pred_poly, testing$TB_Status)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8490566
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1
```

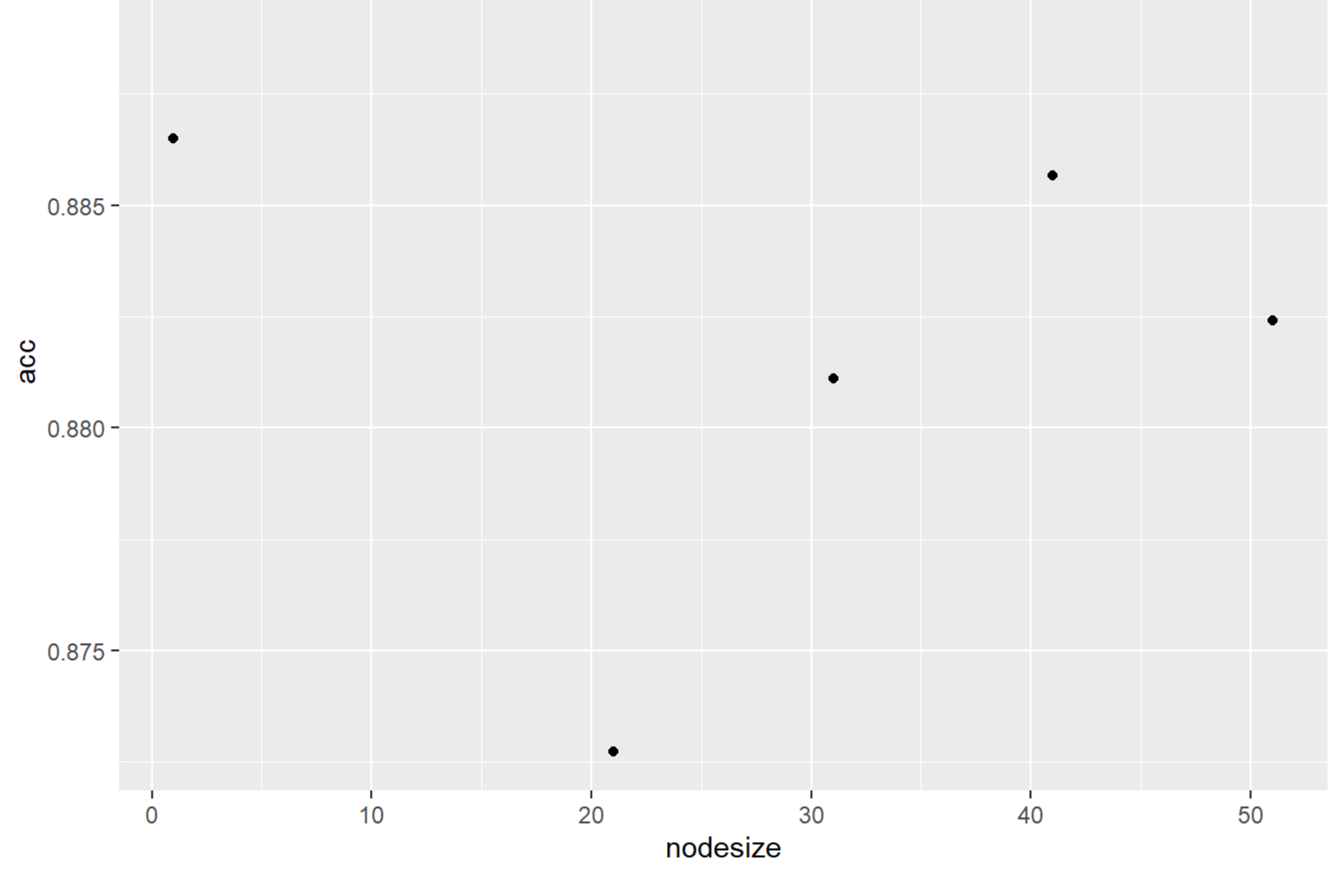
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
nodesize <- seq(1, 51, 10)
acc <- sapply(nodesize, function(ns){
      train(TB_Status ~ ., method = "rf", data = training, tuneGrid = data.frame(mtry = 2),
      nodesize = ns)$results$Accuracy
})
qplot(nodesize, acc)
```




```
train_rf_2 <- randomForest(TB_Status ~ ., data=training,
    nodesize = nodesize[which.max(acc)])
confusionMatrix(predict(train_rf_2, testing),
    testing$TB_Status)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8679245
```

Feedforward Perceptron Neural Network

```
nn_caret <- (caret::train(TB_Status~., data = training,
method = "nnet"))
```

```
ps <- predict(nn_caret, testing)
print(confusionMatrix(ps, testing$TB_Status)$overall["Accuracy"])
```

```
##  Accuracy
## 0.8490566
```

(20 points) Compare the overall accuracy of the prediction methods for each of the machine learning tools in the previous problem. Which one performs the best?

The Random forest and radial SVM with their 86.7% accuracy do the best of these methods here.The dendogram, however did Even better at classifying than the random Forest and Radial SVM.