Multivariate statistics: Assignment 2

Team 27: **Raïsa Carmen** *s0204278* **Marco Chi Chung Fong** *r0865521*
Wenting Jiang *r0824739* **Chin Wei Ma** *r0877202*

All R scripts and the data can be found on this GitHub repository.

1 Task 1

The data consists of 18 variables of which the categorical variable *Private* denotes whether the school is private (*Private* == *Yes*) or not (*Private* == *No*). There are 777 schools in the data; 565 (72.72%) private schools and 212 (27.28%) non-private schools.

A detailed outline of all functions that are used for this task can be found in the appendix and on this GitHub repository. Before we start to build models to distinguish private and non-private schools, Wilk's lambda test is performed where the null hypothesis states that the centroid (mean) in both groups is the same. Both in the log-transformed and the original (centered or standardized) data, this null hypothesis is rejected. This means that the centroids of private and non-private schools differ significantly.

The test of Box to test for equality of within-group covariance matrices shows that H_0 of equal covariance matrices across groups is not supported by the data. However, we still report on the results from LDA (which assumes equal covariances) since the alternative, QDA, does not always perform better because the lower bias of the QDA classifier may not outweigh its higher model complexity.

Table 1 shows all results for the different performance measures (hit rate, sensitivity and specificity) and the different datasets. Overall, sensitivity (proportion of positive observations that are being correctly classified) tends to be higher than specificity (proportion of negative observations that are being correctly classified) meaning it's generally easier to detect private schools than non-private schools.

As expected, it does not matter whether the data is standardized or centered for the QDA and LDA models and log-transforming skewed variables yields a higher hit rate in LDA and QDA. The hit rate is higher in LDA than in QDA which suggests that the true decision boundary is likely close to linear. The extra complexity of QDA is not worthwhile.

It is curious that KNN works so badly on the centered data with log-transformed skewed variables. The highest overall hit rate, for all datasets, is achieved in the random forest models. Bagging achieves approximately the same performance as LDA although it sacrifices specificity for higher sensitivity.

```
College_logtranfskewed <- College %>%
  mutate(Apps = log(Apps),
         Accept = log(Accept),
         Enroll = log(Enroll),
         Top10perc = log(Top10perc),
         F.Undergrad = log(F.Undergrad),
         P.Undergrad = log(P.Undergrad),
         Books = log(Books),
         Personal = log(Personal),
         Expend = log(Expend))
College_centered <- College %>%
  dplyr::select(-Private) %>%
  scale(center = TRUE, scale = FALSE) %>%
  as.data.frame() %>%
  mutate(Private = College$Private)
College_logtranfskewed_centered <- College_logtranfskewed %>%
```

```
  dplyr::select(-Private) %>%
  scale(center = TRUE, scale = FALSE) %>%
  as.data.frame() %>%
  mutate(Private = College$Private)
College_std <- College %>%
  dplyr::select(-Private) %>%
  scale(center = TRUE, scale = TRUE) %>%
  as.data.frame() %>%
  mutate(Private = College$Private)
College_logtranfskewed_std <- College_logtranfskewed %>%
  dplyr::select(-Private) %>%
  scale(center = TRUE, scale = TRUE) %>%
  as.data.frame() %>%
  mutate(Private = College$Private)
testlist <- list(center = College_centered,
                 center_logtransfskewed = College_logtranfskewed_centered,
                 std = College_std,
                 std_logtransfskewed = College_logtranfskewed_std
                 )
#test the difference between centroids
wilks_results <- testlist %>% purrr::map(function(x) Wilks_manova(x, "Private"))
#In each dataset, H0:mu_{yes} = mu_{no} is rejected -->
# centroids of Private and non-private schools differ significantly
#test the difference between centroids
boxm_results <- testlist %>% purrr::map(function(x) boxM_test(x, "Private"))
#LDA
lda_results <- map2(testlist, names(testlist),
                    function(x,y){lda_analysis(x,"Private", y)}) %>%
  do.call("rbind", .)
#QDA
qda_results <- map2(testlist, names(testlist),
                    function(x,y){qda_analysis(x,"Private", y)}) %>%
  do.call("rbind", .)
#KNN
knn_output <- map2(testlist, names(testlist),
                    function(x,y){knn_analysis(x,"Private", y, 1:100)})
knn_results <- lapply(knn_output, function(element){
  element[[2]] # The first element contains graphs
})
knn_results <- knn_results %>%
  do.call("rbind", .)
#bagging
bagging_output <- map2(testlist, names(testlist),
                    function(x,y){bagging_analysis(x, "Private", y,
                                                   ntree = 2000)})
bagging_results <- lapply(bagging_output, function(element){
  element[[2]] # The first element contains graphs
})
bagging_results <- bagging_results %>%
  do.call("rbind", .)
```

Table 1: A performance comparison of the different methods

Method	Hit rate				Sensitivity				Specificity			
	Centered	Centered Logtransf	Standardized	Standardized Logtransf	Centered	Centered Logtransf	Standardized	Standardized Logtransf	Centered	Centered Logtransf	Standardized	Standardized Logtransf
LDA	0.941	0.942	0.941	0.942	0.952	0.949	0.952	0.949	0.910	0.925	0.910	0.925
QDA	0.909	0.929	0.909	0.929	0.961	0.940	0.961	0.940	0.769	0.901	0.769	0.901
KNN	0.937	0.852	0.932	0.943	0.968	0.901	0.972	0.975	0.854	0.722	0.825	0.858
bagging	0.941	0.942	0.941	0.940	0.966	0.970	0.968	0.970	0.873	0.868	0.868	0.858
randomForest	0.947	0.946	0.950	0.949	0.973	0.973	0.973	0.975	0.877	0.873	0.887	0.877

```
#random forests
rf_output <- map2(testlist, names(testlist),
                  function(x,y){RF_analysis(x, "Private", y,
                                            ntree = 2000, mtry = "optimize")})
rf_results <- lapply(rf_output, function(element){
  element[[2]] # The first element contains graphs
})
rf_results <- rf_results %>%
  do.call("rbind", .)
```

For the K-nearest neighbour (KNN) approach, the function knn_cv from the *class* package is used to execute KNN with LOOCV and test values of K between 1 and 100. The model with the highest hit rate is chosen as the final model in Table 1. If there are multiple models with the same hit rate, the smallest K is chosen, Figure 1 shows the results in each of the datasets. The chosen K is 6 for the centered data, 64 for the centered data where the skewed variables are log-transformed, 7 for the standardized data, 11 for the standardized data where the skewed variables are log-transformed.

For the bagging and Random forest (RF) approach, the function randomForest from the *randomForest* package is used. 2000 bootstrapped data and trees are drawn in all models. Figure 2 and 3 show the importance of different variables in the final models. *F.Undergrad* and *outstate* are the most important one in all bagging and RF models. The difference in MeanDecreaseGini (the mean decrease in the Gini index) between the most important variables (*F.Undergrad* and *Outstate*) and the other variables is much larger in the bagging model than in the RF models. *Enroll*, which is not at all important in the bagging models, is consistently the third most important variable in the RF models.

For the RF results, we try different values for the mtry parameter which is the number of variables randomly sampled as candidates at each split. The model with the highest hit rate is chosen as the final model in Table 1. If there are multiple models with the same hit rate, the smallest mtry is chosen, Figure 4 shows the results in each of the datasets. The chosen value for mtry is 8 for the centered data (Figure 4A), 9 for the centered data where the skewed variables are log-transformed (Figure 4B), 12 for the standardized data (Figure 4C), 9 for the standardized data where the skewed variables are log-transformed (Figure 4D).

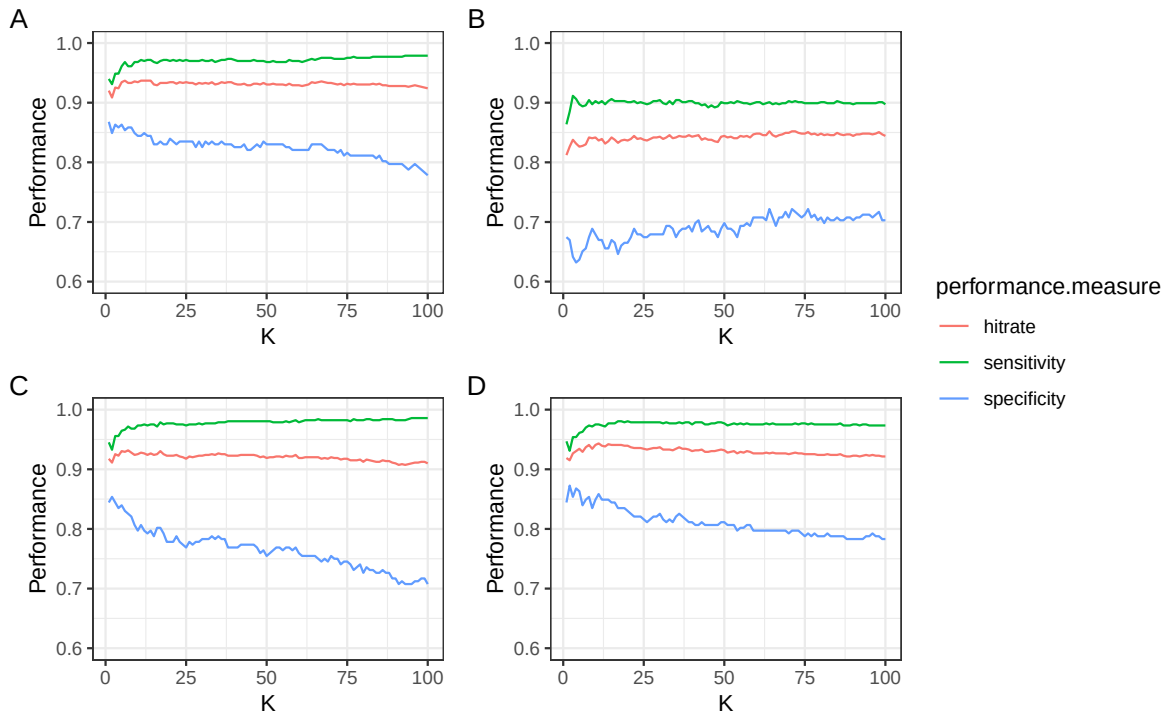


Figure 1: Evolution of hit rate, sensitivity and specificity for different K in the KNN approach. A shows the centered data, B the centered data with log-transformed skewed variables, C shows the standardized data, and D the standardized data with log-transformed skewed variables.

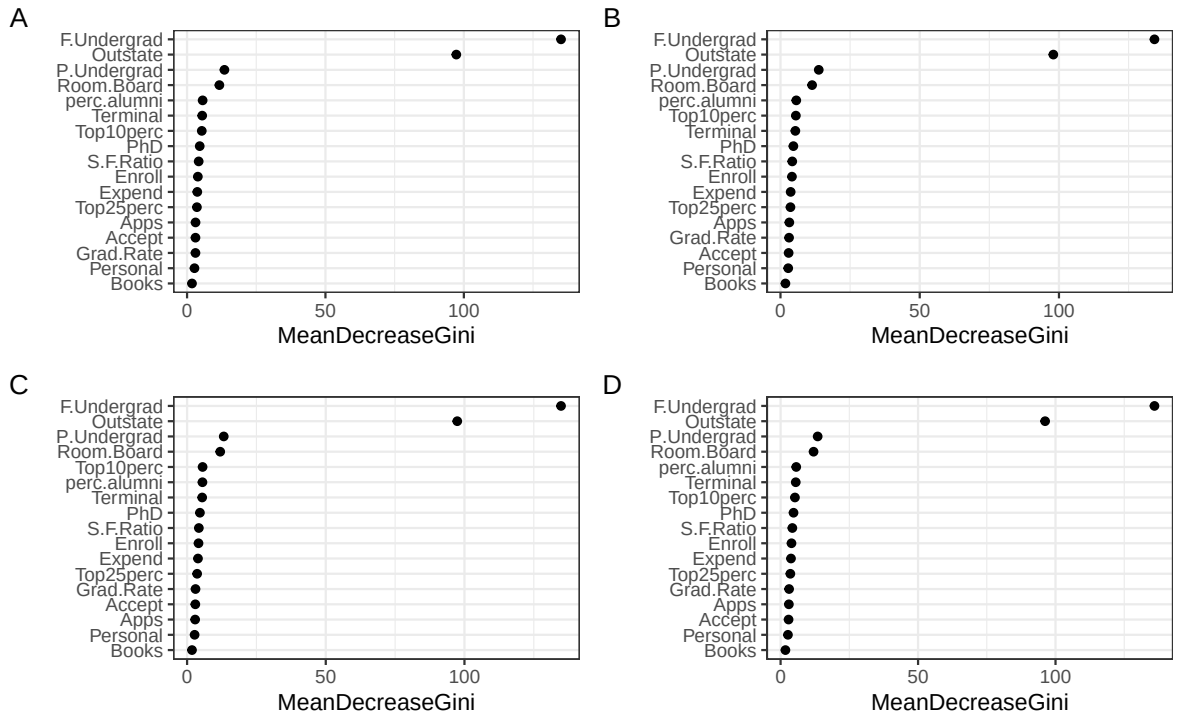


Figure 2: Variance importance plots for the bagging models. A shows the centered data, B the centered data with log-transformed skewed variables, C shows the standardized data, and D the standardized data with log-transformed skewed variables.

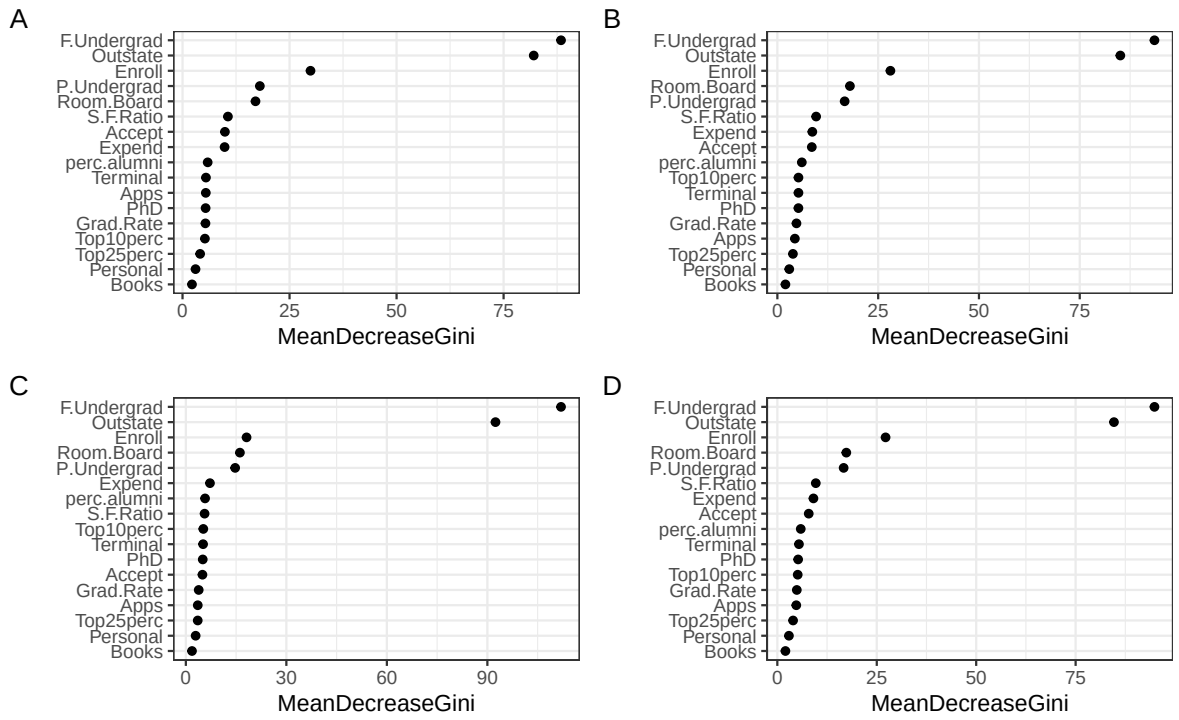


Figure 3: Variance importance plots for the random forest models. A shows the centered data, B the centered data with log-transformed skewed variables, C shows the standardized data, and D the standardized data with log-transformed skewed variables.

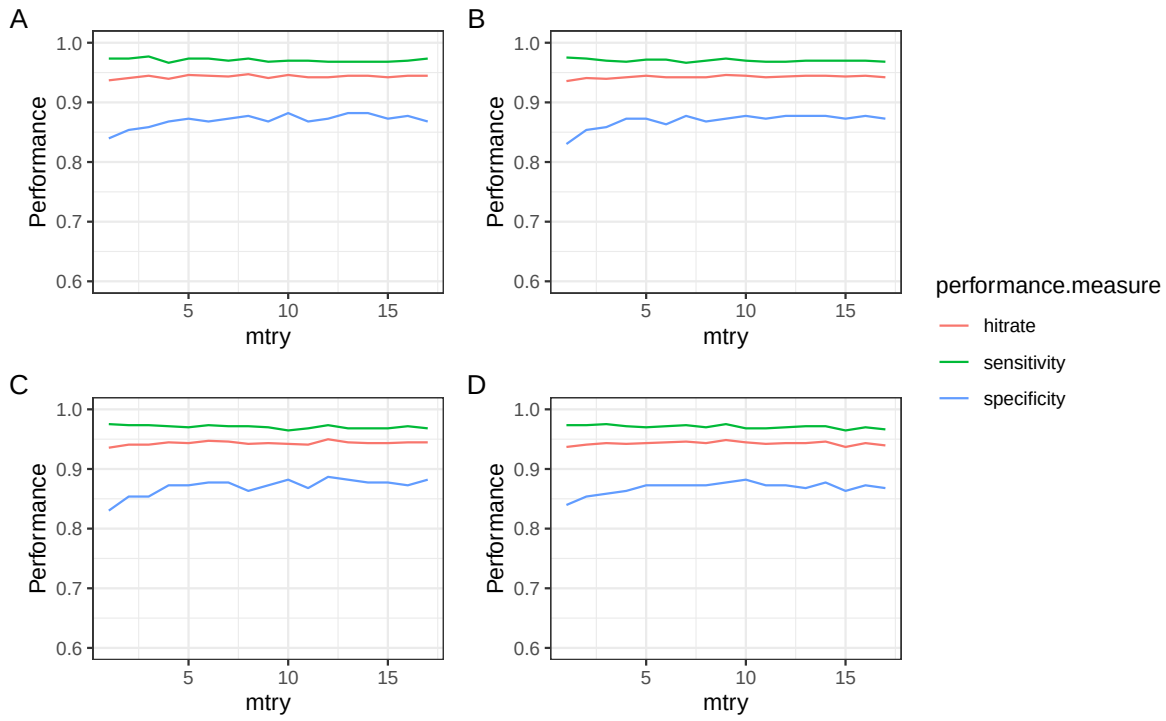


Figure 4: Optimization of the mtry parameter for the random forest models. A shows the centered data, B the centered data with log-transformed skewed variables, C shows the standardized data, and D the standardized data with log-transformed skewed variables.

2 Task 2

```
load(find_root_file("data/fashion.Rdata",
     criterion = has_file("MultivariateStatistics_assignment.Rproj")))
#centered variables
zshopping <- scale(train.data, center = TRUE, scale = FALSE)
##### HDclassif ######
set.seed(20)
#common dimension hddc() models "AkjBkQkD" and "AkjBQkD" with
# com_dim=2,3,4,5, or 6 fitted on the training data.
names <- list()
ari_values <- list()
bic <- list()
for (model in c("AkjBkQkD", "AkjBQkD")) {
  for (dim in 2:6)
  {
    hddc.out <- hddc(zshopping, K = 3, model = model, com_dim = dim,
                     d_select = "BIC", init = "kmeans")
    ari <- adjustedRandIndex(hddc.out$class, train.target)
    model_name = paste(model,"_",toString(dim), sep = "")
    names <- append(names, model_name)
    ari_values <- append(ari_values, ari)
    bic <- append(bic, hddc.out$BIC)
  }
}

###### mclust model ######
set.seed(201)
#Mclust() models "VVE","VEV","EVV","VVV" fitted on the first 2,3,4,5, or 6
#unstandardized principal components extracted from the training data.
prcomp.out <- prcomp(zshopping)
comp <- zshopping %*% prcomp.out$rotation
names2 <- list()
ari_values2 <- list()
bic2 <- list()
for (model in c("VVE","VEV","EVV","VVV")) {
  for (c in 2:6)
  {
    mclust.out <- Mclust(comp[,1:c], G = 3, modelNames = model)
    ari <- adjustedRandIndex(mclust.out$class, train.target)
    model_name = paste(model, "_", toString(c), sep = "")
    names2 <- append( names2, model_name)
    ari_values2 <- append(ari_values2, ari)
    bic2 <- append(bic2,mclust.out$bic)
  }
}
###### the best model (highest ARI value) selected is AkjBQkD_4 ######
hddc.out <- hddc(zshopping, K = 3, model = "AkjBQkD", com_dim = 4, d_select = "BIC",
                 init = "kmeans")
ari <- adjustedRandIndex(hddc.out$class, train.target)
```

Table 2: A comparison of HDDC models

Method_dimensions	ARI	BIC
AkjBkQkD_2	0.820	-142484930
AkjBkQkD_3	0.853	-140494797
AkjBkQkD_4	0.878	-139060581
AkjBkQkD_5	0.869	-138065164
AkjBkQkD_6	0.873	-137012940
AkjBQkD_2	0.921	-142899972
AkjBQkD_3	0.929	-140964637
AkjBQkD_4	0.937	-139519143
AkjBQkD_5	0.914	-138558927
AkjBQkD_6	0.922	-137642837

```
tab <- table(hddc.out$class, train.target)
#switch class labels to have maximum correspondance
mapping <- mapClass(hddc.out$class, train.target)
#compute principal components
loading <- prcomp.out$rotation %*% diag(prcomp.out$sd)
Z <- predict(prcomp.out, zshopping)
```

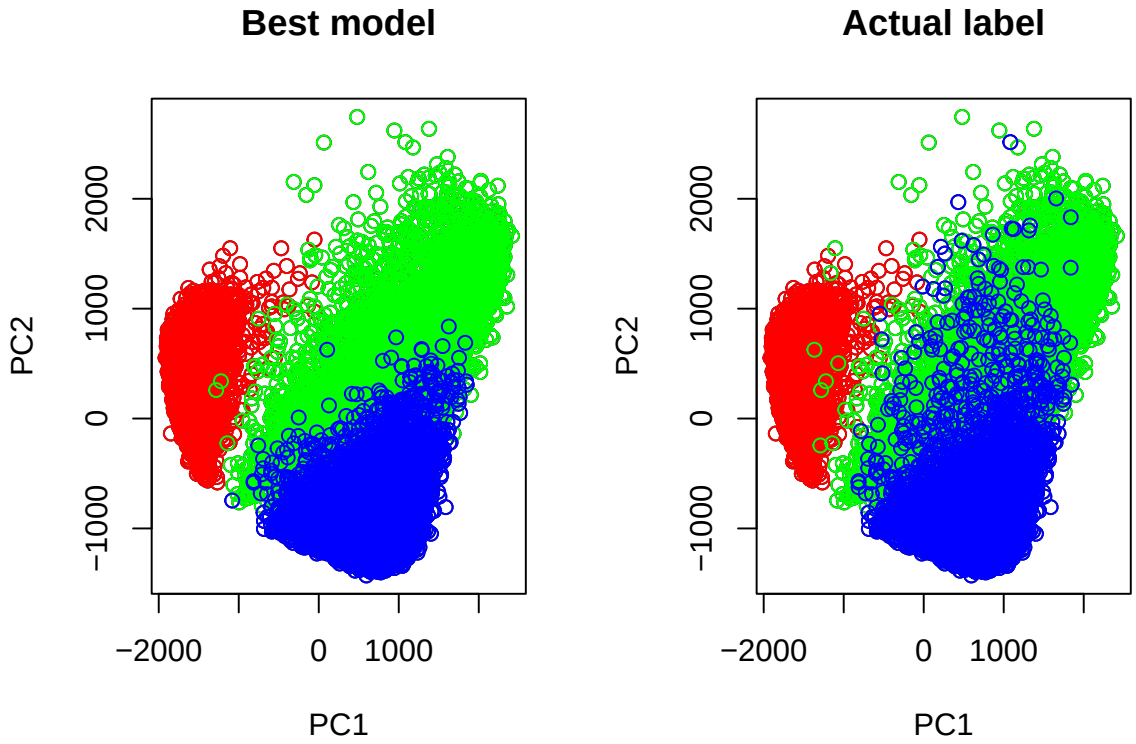


Figure 5: A comparison of the best models with the true, actual labels.

To evaluate the extent to which the model-based unsupervised clustering can recover true class labels, we use the Adjusted Rand Index (ARI) based on the true and predicted class labels. Model with a higher ARI is then considered as a better model.

Table 3: A comparison of mclust models

Method_dimensions	ARI	BIC
VVE_2	0.819	-564812.0
VVE_3	0.886	-807892.3
VVE_4	0.884	-1048496.5
VVE_5	0.896	-1276211.0
VVE_6	0.931	-1512498.8
VEV_2	0.807	-560253.8
VEV_3	0.837	-801893.2
VEV_4	0.856	-1035633.4
VEV_5	0.903	-1265311.9
VEV_6	0.928	-1495718.6
EVV_2	0.784	-570128.0
EVV_3	0.755	-811909.7
EVV_4	0.870	-1037155.2
EVV_5	0.925	-1267018.9
EVV_6	0.880	-1503120.6
VVV_2	0.812	-560231.2
VVV_3	0.861	-799824.9
VVV_4	0.876	-1030593.7
VVV_5	0.929	-1259639.6
VVV_6	0.919	-1489252.7

When fitting the common dimension hddc models on the training data, with a 3-cluster model specified as "AkjBkQkD" or "AkjBQkD", dimensions from 2 to 6 (Table 2), we can see that the best performing model is "AkjBQkD" with 4 dimensions (ARI = 0.937). It has a BIC of -1.3951914×10^8, which is however not the lowest value among all models. The model "AkjBQkD" with 2 dimensions has a BIC of -1.4289997×10^8 and an ARI of 0.921.

In addition, when fitting the Mclust models "VVE","VEV","EVV","VVV" on the first 2 to 6 unstandardized principal components (Table 3), we can see that the best performing model is "VVE" fitted on 6 principal components (ARI = 0.937). Comparatively, the best hddc model outperforms the Mclust model in terms of ARI. Also, the BIC values of Mclust are all much higher than those of hdcc models.

When visualizing the predicted class labels for the best clustering model in the space of the first two principal components, we can see that the space is divided into 3 parts colored each in red, green and blue (Figure 5). There is little overlap among the clusters, providing a visual illustration that the clustering worked well. On the other hand, when visualizing the actual class labels in the space of the first two principal components, we see some clear overlaps between two clusters.

Table 4: Stress-1, mean-2*standard deviation for stress norm test, and mean for the permutation test.

measurement levels	stress-1	stress norm	permutation test
ratio	0.153	0.344	0.302
interval	0.127	0.275	0.295
ordinal	0.089	0.246	0.268
mspline	0.107	0.267	0.287

3 Task 3

```
#ratio
m1 <- smacofSym(delta = rectangles, ndim = 2, type = "ratio", init = "torgerson")
#interval
m2 <- smacofSym(delta = rectangles, ndim = 2, type = "interval", init = "torgerson")
#ordinal
m3 <- smacofSym(delta = rectangles, ndim = 2, type = "ordinal", init = "torgerson")
#spline
m4 <- smacofSym(delta = rectangles, ndim = 2, type = "mspline",
                spline.degree = 4, spline.intKnots = 4, init = "torgerson")
#stress-1 values
round(c(m1$stress, m2$stress, m3$stress, m4$stress), 3)
```

```
#check goodness of fit using stress-norms and permutation test
#stress norm
set.seed(1)
rstress.m1 <- randomstress(n = 16, ndim = 2, nrep = 500, type = "ratio")
rstress.m2 <- randomstress(n = 16, ndim = 2, nrep = 500, type = "interval")
rstress.m3 <- randomstress(n = 16, ndim = 2, nrep = 500, type = "ordinal")
rstress.m4 <- randomstress(n = 16, ndim = 2, nrep = 500, type = "mspline")
#distribution of stress for random data
distr_stress1 <- mean(rstress.m1) - 2 * sd(rstress.m1)
distr_stress2 <- mean(rstress.m2) - 2 * sd(rstress.m2)
distr_stress3 <- mean(rstress.m3) - 2 * sd(rstress.m3)
distr_stress4 <- mean(rstress.m4) - 2 * sd(rstress.m4)
#permutation test
set.seed(1)
perm.m1 <- permtest(m1, nrep = 500)
perm.m2 <- permtest(m2, nrep = 500)
perm.m3 <- permtest(m3, nrep = 500)
perm.m4 <- permtest(m4, nrep = 500)
#stability of solution using jackknife
jack.car <- jackmds(m3)
```

```
#compute MDS biplot: run multivariate linear regression of
#external variables on configuration
zrect_constr <- scale(rect_constr, center = TRUE)
#biplot MDS: regression of standardized external variables on configuration dimensions
bimorse <- biplotmds(m3, zrect_constr, scale = TRUE)
#coefficients
bimorse_coef <- round(coef(bimorse), 3)
```

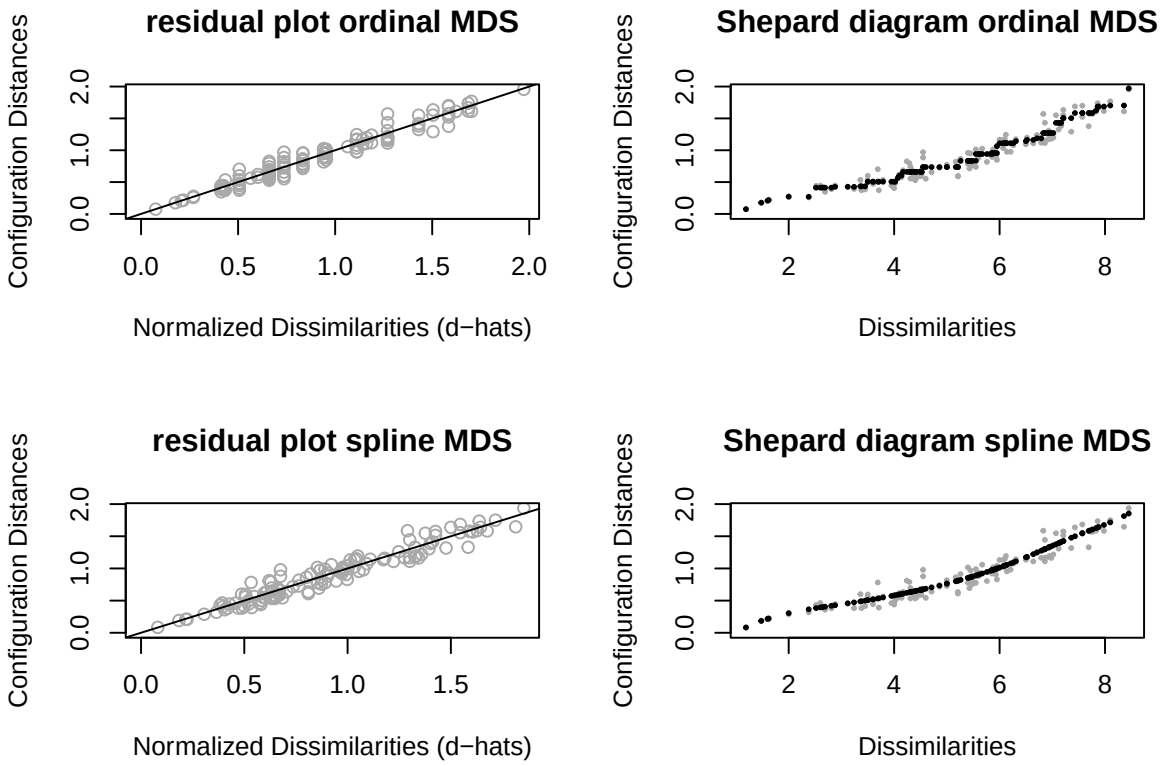


Figure 6: Residual plots and Shepard diagrams for ordinal and spline MDS.

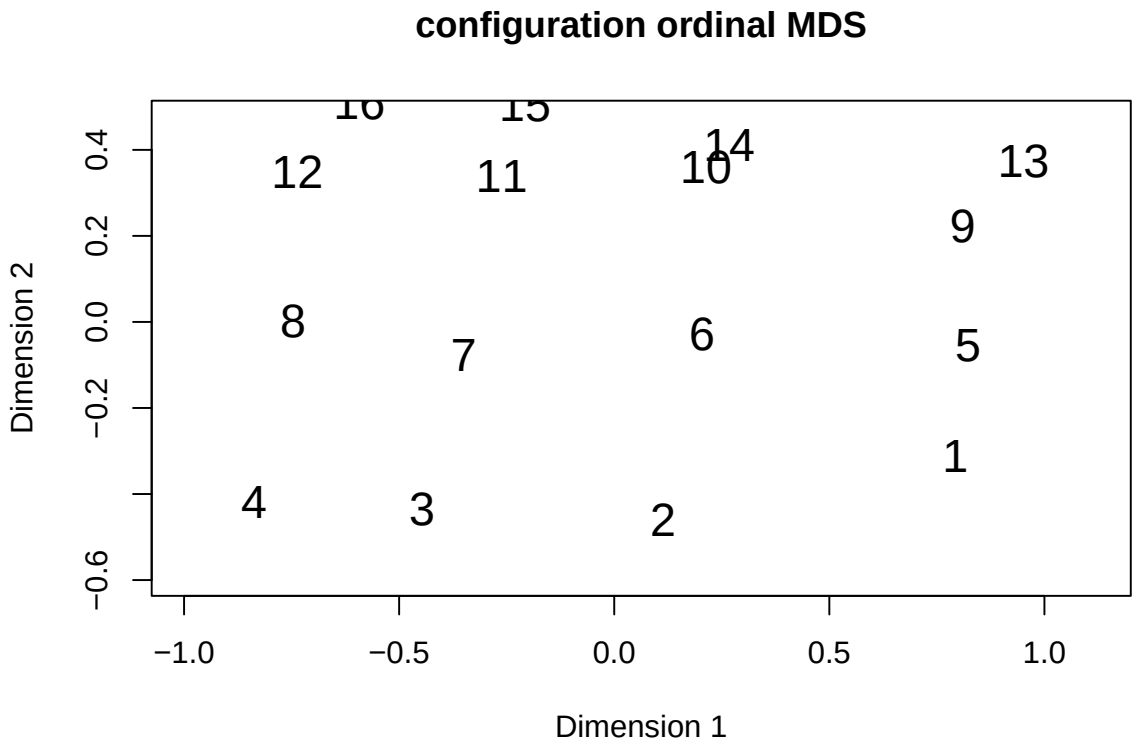


Figure 7: Configuration ordinal MDS.

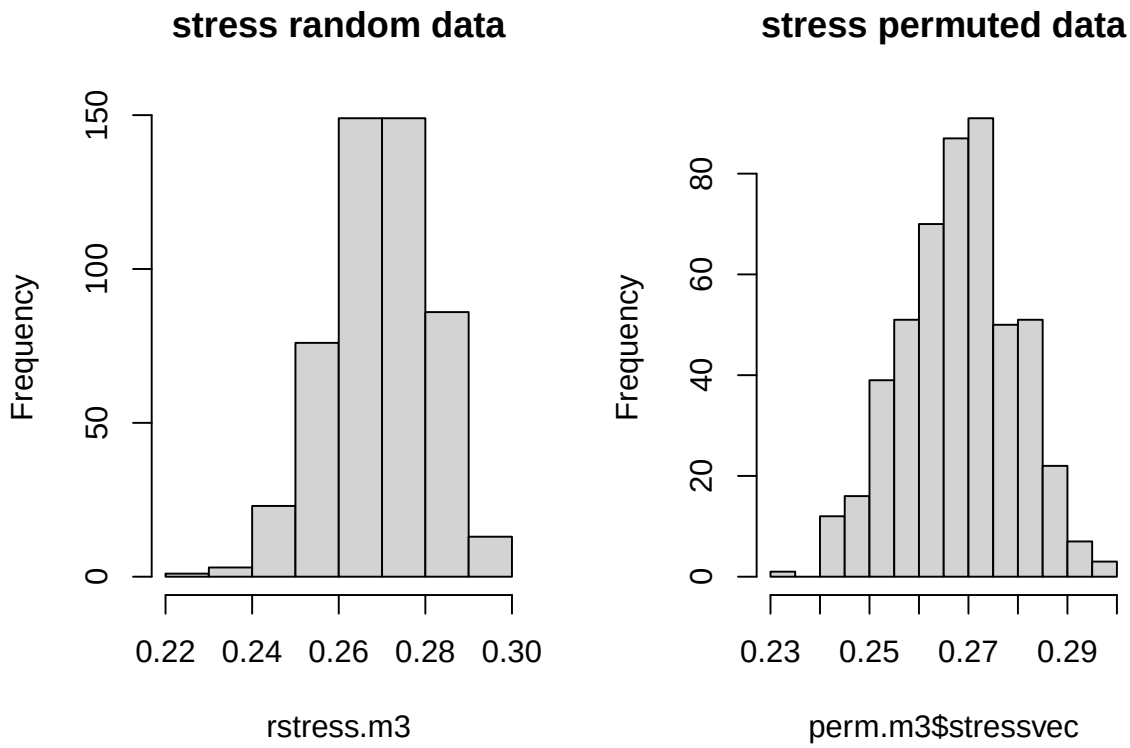


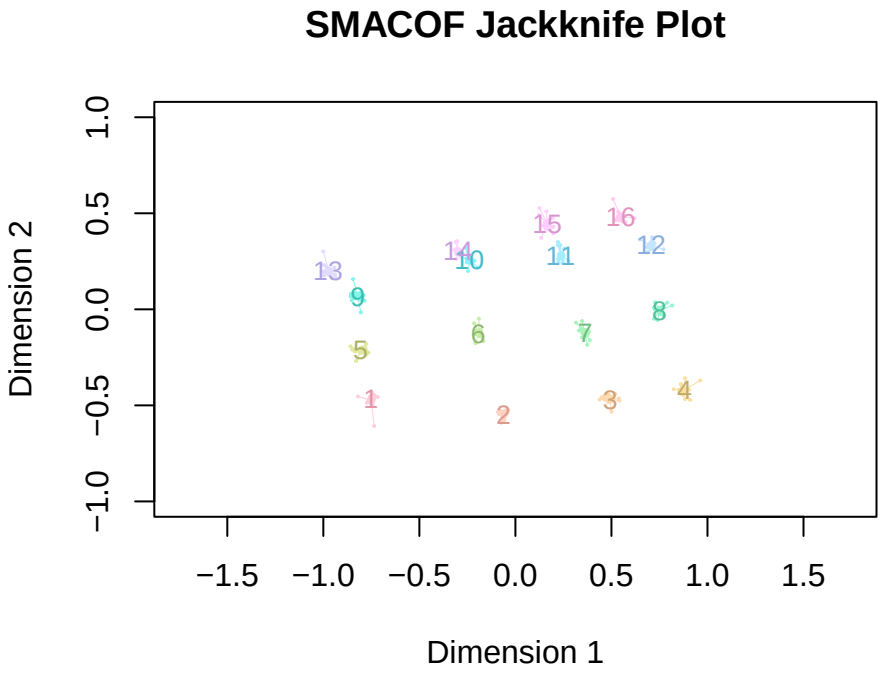
Figure 8: Distribution of stress.

```
#print R-square of regressions
bimorse_r2 <- round(bimorse$R2vec, 3)
#print correlations of dimensions and external variables
bimorse_corr <- round(cor(m3$conf, zrect_constr), 3)
```

The results show that stress-1 is lowest using ordinal MDS (0.089, Table 4) or spline MDS (0.107). A residual plot and shephard plot for these is shown in Figure 6. Since stress-1 is between 0.05 and 0.1 for ordinal MDS, it is a fair fit. The configuration can be seen in Figure 7 where we see that rectangles 10 and 14 are perceived to be very similar (as are the pairs 11 & 15 and 12 & 16). Two options to better assess goodness of fit, are stress norm and permutation test (results for all measurement levels in Table 4). For ordinal MDS, we can see that the stress-1 obtained on the real dataset (0.089) is much lower than 0.246 (the mean $-2*$ standard deviation of the stress random data). Similarly, it is much lower than the mean of the distribution from the stress permuted data (0.268, Figure 8). We can thus conclude that ordinal MDS has a satisfactory fit.

Table 5: Ordinal MDS dimensions and their link with the external variables (width and height).

	Width	Height
Coefficients		
D1	0.096	-1.623
D2	2.778	0.334
Correlations		
D1	0.117	-0.987
D2	0.970	0.056
Rsquared	0.945	0.987



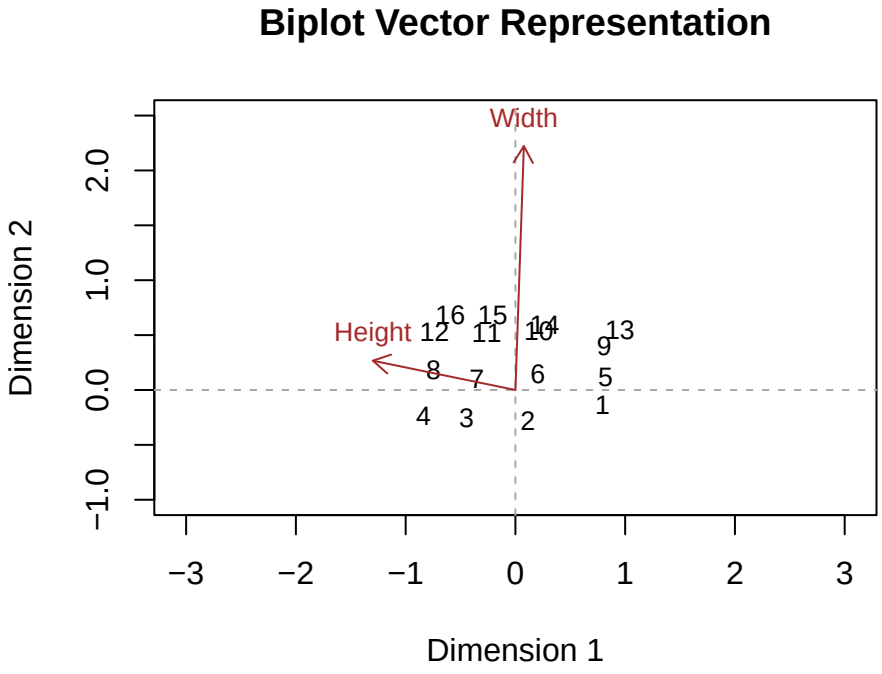


Figure 9: Jackknife plot and biplot.

From the jackknife plot for the chosen, ordinal MDS (Figure 9), we can see that the solution is quite stable, because the positions of the points do not change much. The stability measure reported by jackmds is 0.998, also confirming that the solution is very stable.

When projecting the external variables height and width in the MDS solution (Table 5), we can see from the coefficients and the plot (Figure 9) that, if D1 increases one unit, the predicted average height of rectangles will decrease 1.623 standard deviations, and if D2 increases one unit, the predicted average height will increase 0.334 standard deviations. On the other hand, if D2 increases one unit, the predicted average width of rectangles will increase 2.778 standard deviations, and if D1 increases one unit, the predicted average width will increase 0.096 standard deviations. We also see that D1 and D2 account for 98.7% of the variation in the height around its mean, and 94.5% of the variation in the width around its mean. The correlations of D1 and D2 with the external variables show that the first dimension has a strong negative correlation with the height (cor = -0.987), and that the second dimension has rather strong positive correlation with the width (0.970).

4 Appendix

Task1: code to define the functions

```
Wilks_manova <- function(data, group = "Private"){
  lm.out <- lm(cbind(
    sapply(X = colnames(data)[colnames(data) != group],
           FUN = function(x) get(x)))~
       get((group)),
   data = data)
  s <- summary(manova(lm.out), test = c("Wilks"))
  return(s$stats[1, "Pr(>F)"])#return the pvalue
}
boxM_test <- function(data, group = "Private"){
  out <- boxM(data[,colnames(data) != group], data[,group])
  return(out$p.value)#return the pvalue
}
lda_analysis <- function(data, group = "Private", datatype){
  K <- length(unique(levels(data[,group])))
  positivelevel <- levels(data[,group])[K]# by default, it is assumed the last level is the positive
  negativelevel <-  levels(data[,group])[-K]
  lda.out <- lda(
    formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                     collapse = "+"))),
    data = data,
    prior = rep(1/K, K),
    CV = TRUE)
  tab <- table(data[,group], lda.out$class)
  out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "LDA"
  )
  return(out)
```

```
}
qda_analysis <- function(data, group = "Private", datatype){
  K <- length(unique(levels(data[,group])))
  positivelevel <- levels(data[,group])[K]# by default, it is assumed the last level is the positive
  negativelevel <-  levels(data[,group])[-K]
  qda.out <- qda(
    formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                     collapse = "+"))),
    data = data,
    prior = rep(1/K, K),
    CV = TRUE)
  tab <- table(data[,group], qda.out$class)
  out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "QDA"
  )
  return(out)
}
knn_analysis <-  function(data, group = "Private", datatype, testk){
  K <- length(unique(levels(data[,group])))
  positivelevel <- levels(data[,group])[K]# by default, it is assumed the last level is the positive
  negativelevel <-  levels(data[,group])[-K]
  output <- data.frame(hitrate = NA,
                    sensitivity = NA,
                    specificity = NA,
                    data = datatype,
                    method = "KNN",
                    K = testk)
  for (x in testk) {
    knn.out <- knn.cv(train = data[,colnames(data) != group],
                      cl = data[, group],
                      k = x,
                      prob = TRUE
                      )
    tab <- table(data[,group], unname(knn.out))
    out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "KNN",
                    K = x
    )
  output[x, ] <- out
  }
```

```
  graph <- output %>%
    pivot_longer(cols = c("hitrate","sensitivity", "specificity"),
                 names_to = "performance.measure", values_to = "value") %>%
    ggplot() +
    geom_line(aes(color = `performance.measure`, x = K, y = value)) +
    ylab("Performance") + theme_bw() + ylim(0.6, 1)
  selected <- which(output$hitrate == max(output$hitrate))[1]
  return(list(graph,output[selected,]))
}
bagging_analysis <- function(data, group = "Private", datatype, ntree){
  K <- length(unique(levels(data[,group])))
  positivelevel <- levels(data[,group])[K]# by default, it is assumed the last level is the positive
  negativelevel <-  levels(data[,group])[-K]
  nvar <- ncol(data)-1
  bagging.out <- randomForest(
    formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                     collapse = "+"))),
    data = data,
    mtry = nvar,
    ntree = ntree,
    importance = TRUE)
  tab <- table(data[,group], bagging.out$predicted) #the predicted values of the input data based on
  out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "bagging"
    )
  graph <- bagging.out$importance %>%
    as.data.frame %>%
    dplyr::select(MeanDecreaseGini) %>%
    mutate(Var = rownames( bagging.out$importance )) %>%
    ggplot() + geom_point(aes(x=MeanDecreaseGini, y=Var)) +
  scale_y_discrete(limits = names(sort(bagging.out$importance[,4]))) +
    theme_bw() + ylab("")
  return(list(graph, out))
}
RF_analysis <- function(data, group = "Private", datatype, ntree, mtry){
  K <- length(unique(levels(data[,group])))
  nvar <- ncol(data)-1
  positivelevel <- levels(data[,group])[K]# by default, it is assumed the last level is the positive
  negativelevel <-  levels(data[,group])[-K]
  if (mtry != "optimize") {
    bagging.out <- randomForest(
      formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                       collapse = "+"))),
      data = data,
      mtry = mtry,
```

```
    ntree = ntree,
    importance = TRUE)
  tab <- table(data[,group], bagging.out$predicted) #the predicted values of the input data based
  out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "randomForest"
    )
  graph <- bagging.out$importance %>%
    as.data.frame %>%
    dplyr::select(MeanDecreaseGini) %>%
    mutate(Var = rownames( bagging.out$importance )) %>%
    ggplot() + geom_point(aes(x=MeanDecreaseGini, y=Var)) +
  scale_y_discrete(limits = names(sort(bagging.out$importance[,4]))) +
    theme_bw() + ylab("")
  return(list(graph, out))

}else{
  output <- data.frame(hitrate = NA,
                    sensitivity = NA,
                    specificity = NA,
                    data = datatype,
                    method = "randomForest",
                    mtry = 1:nvar)
  for (mtry in 1:nvar) {
  bagging.out <- randomForest(
    formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                     collapse = "+"))),
    data = data,
    mtry = mtry,
    ntree = ntree,
    importance = TRUE)
  tab <- table(data[,group], bagging.out$predicted) #the predicted values of the input data based
  out <- data.frame(hitrate = sum(diag(tab))/sum(tab),
                    sensitivity = tab[positivelevel, positivelevel] /
                      sum(tab[positivelevel,]),
                    specificity = tab[negativelevel, negativelevel] /
                      sum(tab[negativelevel,]),
                    data = datatype,
                    method = "randomForest",
                    mtry = mtry
    )
output[mtry, ] <- out
}
graphopt <- output %>%
  pivot_longer(cols = c("hitrate","sensitivity", "specificity"),
               names_to = "performance.measure", values_to = "value") %>%
```

```
    ggplot() +
    geom_line(aes(color = `performance.measure`, x = mtry, y = value)) +
    ylab("Performance") + theme_bw() + ylim(0.6, 1)
  selected <- which(output$hitrate == max(output$hitrate))[1]
  bagging.out <- randomForest(
      formula(paste0(group, "~", paste(colnames(data)[colnames(data) != group],
                                       collapse = "+"))),
      data = data,
      mtry = output[selected,'mtry'],
      ntree = ntree,
      importance = TRUE)
  graph <- bagging.out$importance %>%
      as.data.frame %>%
      dplyr::select(MeanDecreaseGini) %>%
      mutate(Var = rownames( bagging.out$importance )) %>%
      ggplot() + geom_point(aes(x=MeanDecreaseGini, y=Var)) +
    scale_y_discrete(limits = names(sort(bagging.out$importance[,4]))) +
      theme_bw() + ylab("")
  return(list(graph, output[selected, ], graphopt))
  }
}
```