Machine Learning Approaches to Dissect Hybrid and Vaccine-Induced Immunity

Contents

0.1 Import Required Libraries

```
# Load necessary libraries
requiredPackages <- c('mclust', 'umap', 'Rtsne', 'ggplot2', 'fpc',
                      'caret', 'dplyr', 'foreach', 'doParallel',
                      'MLmetrics', 'stringr', 'purrr')
for(p in requiredPackages){
  if(!require(p, character.only = TRUE)) install.packages(p)
  library(p, character.only = TRUE)
}
```

0.2 Part 1: Dimensionality Reduction and Clustering

```


set.seed(983)
# Load data
load("../data/data1.rda")
# Use ID as rownames
rownames(data1) <- data1$ID
data1$ID <- NULL
# Print descriptive statistics
summary(data1)
```

0.2.0.1 Load and pre-process Data

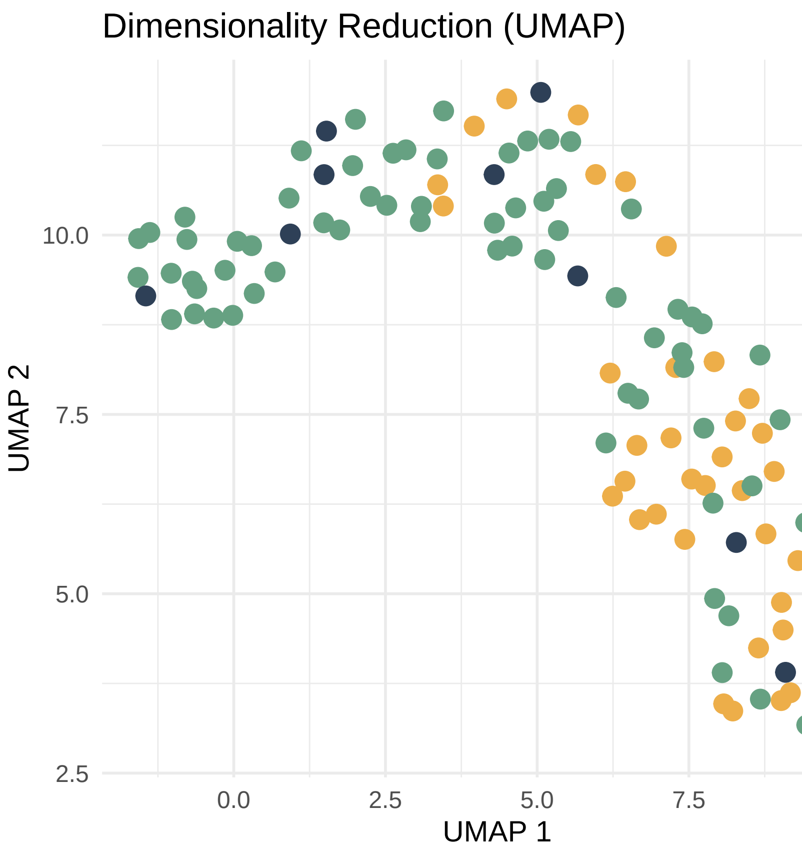
```
##  Infection_0_1pre_2post wt-spike specific IgG (ng/ml)
##  Min.   :0.00           Min.   :  226.3
##  1st Qu.:0.00           1st Qu.: 2799.5
##  Median :0.00           Median : 6846.7
##  Mean   :0.75           Mean   :10450.3
##  3rd Qu.:2.00           3rd Qu.:14957.5
##  Max.   :2.00           Max.   :49629.0
##  Delta-spike specific IgG (ng/ml) Omicron BA.1-spike specific IgG (ng/ml)
##  Min.   :  171.7                  Min.   :   85.74
##  1st Qu.: 1859.3                  1st Qu.:  942.17
##  Median : 4457.2                  Median : 2125.31
##  Mean   : 6755.1                  Mean   : 4431.61
##  3rd Qu.: 8675.3                  3rd Qu.: 4261.54
##  Max.   :41101.0                  Max.   :63891.11
##  Omicron BA.2-spike specific IgG (ng/ml) wt-RBD specific IgG (ng/ml)
##  Min.   :  268.4                         Min.   :  474.6
##  1st Qu.: 2789.6                         1st Qu.: 4585.0
##  Median : 7547.1                         Median :10314.0
##  Mean   :11831.7                         Mean   :14851.2
##  3rd Qu.:12873.4                         3rd Qu.:20192.3
##  Max.   :82626.7                         Max.   :83986.1
##  Delta-RBD specific IgG (ng/ml) Omicron BA.1-RBD specific IgG (ng/ml)
##  Min.   :  366.4                Min.   :  118.6
##  1st Qu.: 2799.5                1st Qu.: 1284.4
##  Median : 6828.9                Median : 3548.1
##  Mean   :10736.1                Mean   : 5068.2
##  3rd Qu.:14372.3                3rd Qu.: 7240.1
##  Max.   :55687.9                Max.   :32335.3
##  Omicron BA.2-RBD specific IgG (ng/ml) ACE2/wt RBD binding inhibition (%)
##  Min.   :  101.8                       Min.   :22.00
##  1st Qu.: 1279.7                       1st Qu.:96.75
##  Median : 3169.5                       Median :98.00
##  Mean   : 5183.6                       Mean   :94.28
##  3rd Qu.: 7141.6                       3rd Qu.:98.00
##  Max.   :30188.2                       Max.   :98.00
##  ACE2/Delta RBD binding inhibition (%) ACE2/BA.1 RBD binding inhibition (%)
##  Min.   :38.00                         Min.   : 0.00
##  1st Qu.:92.75                         1st Qu.: 7.00
##  Median :98.00                         Median :54.00
##  Mean   :91.00                         Mean   :48.95
##  3rd Qu.:98.00                         3rd Qu.:84.00
##  Max.   :98.00                         Max.   :98.00
##  ACE2/BA.2 RBD binding inhibition (%)
##  Min.   : 0.00
##  1st Qu.:66.00
##  Median :88.50
##  Mean   :77.34
##  3rd Qu.:96.00
##  Max.   :98.00
```

```r
str(data1)
```

```
## 'data.frame':    116 obs. of  13 variables:
##  $ Infection_0_1pre_2post                 : num  0 2 2 0 0 2 0 2 2 0 ...
##  $ wt-spike specific IgG (ng/ml)          : num  6369 24915 40220 1912 5101 ...
##  $ Delta-spike specific IgG (ng/ml)       : num  5160 18624 18178 1542 5676 ...
##  $ Omicron BA.1-spike specific IgG (ng/ml): num  2311 10836 9837 841 1505 ...
##  $ Omicron BA.2-spike specific IgG (ng/ml): num  10858 45224 37098 3796 3617 ...
##  $ wt-RBD specific IgG (ng/ml)            : num  5958 29169 23880 5529 7456 ...
##  $ Delta-RBD specific IgG (ng/ml)         : num  5299 20586 20605 5473 5676 ...
##  $ Omicron BA.1-RBD specific IgG (ng/ml)  : num  2955 9112 12893 3289 3146 ...
##  $ Omicron BA.2-RBD specific IgG (ng/ml)  : num  3021 9589 13103 2884 2902 ...
##  $ ACE2/wt RBD binding inhibition (%)     : num  98 98 98 97 98 98 98 98 98 97 ...
##  $ ACE2/Delta RBD binding inhibition (%)  : num  97 98 98 92 98 98 97 97 97 97 ...
##  $ ACE2/BA.1 RBD binding inhibition (%)   : num  0 96 94 0 9 96 97 87 95 7 ...
##  $ ACE2/BA.2 RBD binding inhibition (%)   : num  63 97 98 68 68 98 98 95 97 67 ...
```

```
# Transform the data: log2 transformation and scaling
data1[, c(2:9)] <- log2(data1[, c(2:9)] + 1)
# scale the data
data1[, c(2:13)] <- scale(data1[, c(2:13)], center = T, scale = T)
```

```
set.seed(1778)
# Apply UMAP on the dataset, excluding the first column (assumed to be Infectious status)
um <- umap::umap(data1[,-c(1)], method = 'umap-learn', preserve.seed = T,
                 min_dist = 0.5)

# Convert UMAP results into a data frame
umap_df <- data.frame(um$layout)
umap_df$Group <- as.factor(data1$Infection_0_1pre_2post)
umap_df$ID <- rownames(umap_df)

# Plot UMAP projection
ggplot(umap_df, aes(x = X1, y = X2, color = Group)) +
  geom_point(size = 3) +
  scale_color_manual(labels = c("No Infection", "Infection pre-boost",
                                "Infection post-boost"),
                     values = c("#66a182", "#2e4057", "#edae49")) +
  labs(x = 'UMAP 1', y = 'UMAP 2', title = 'Dimensionality Reduction (UMAP)') +
  theme_minimal()
```



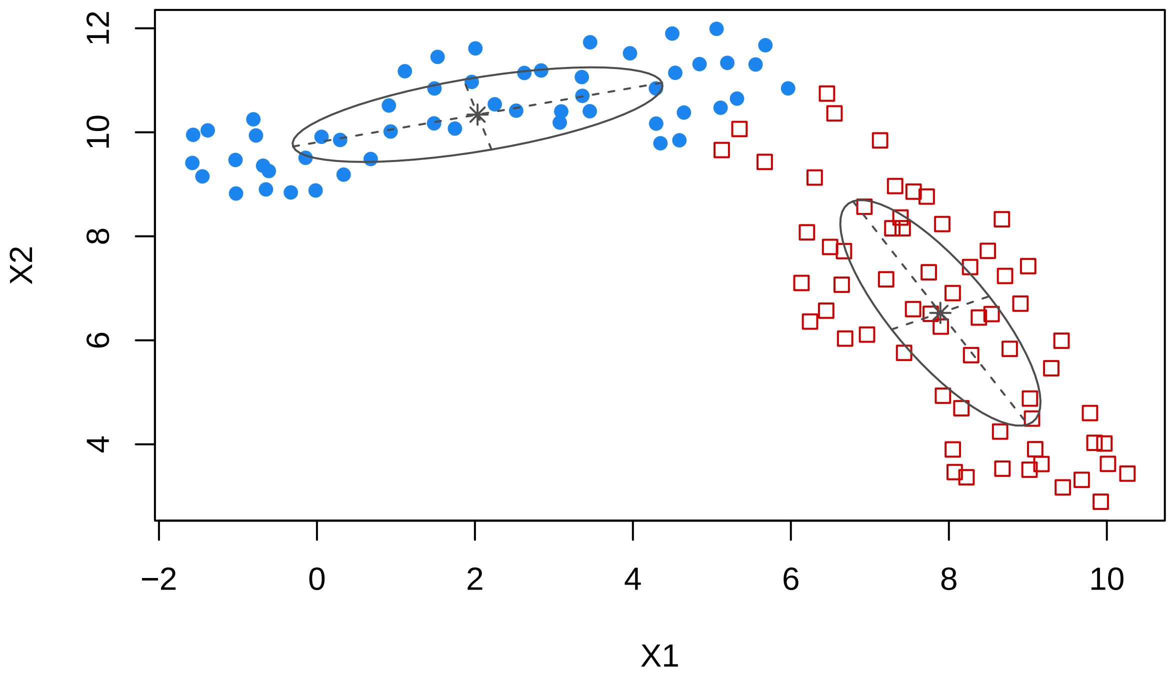
0.2.0.2 UMAP for Dimensionality Reduction

```r
# Clustering with Gaussian Mixture Model (GMM) on UMAP-reduced data (unsupervised)
um_gmm = mclust::Mclust(umap_df[, c(1, 2)])
# Print summary of clustering results
summary(um_gmm)
```

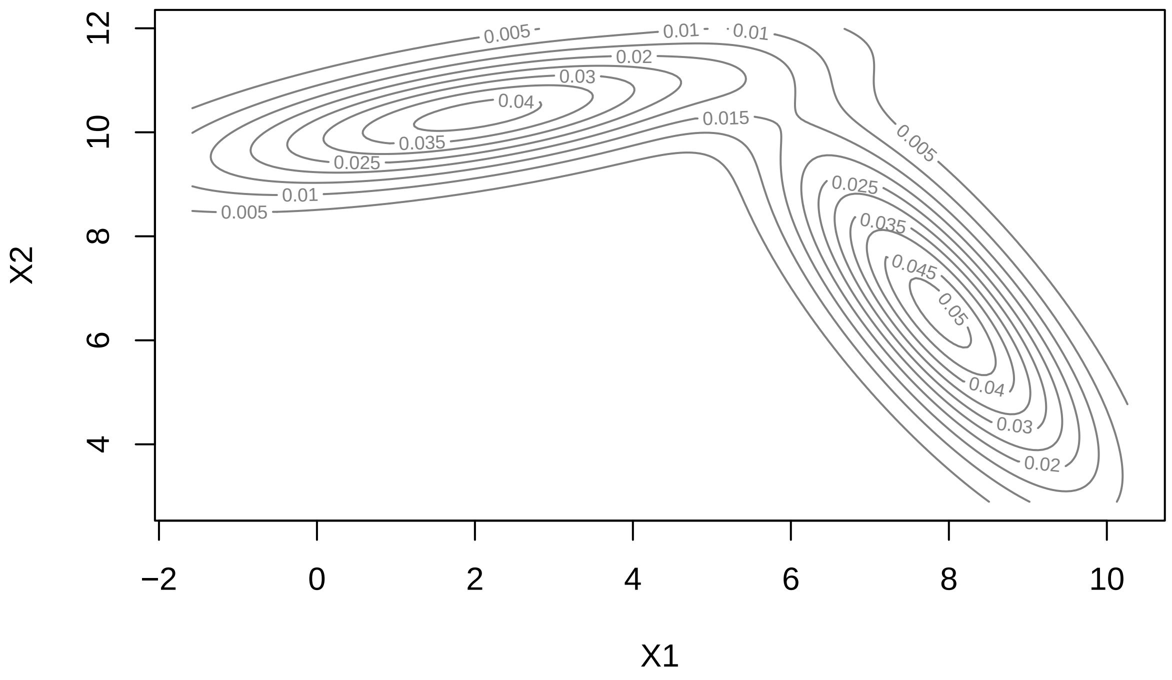
0.2.0.3 Clustering with Gaussian Mixture Model (GMM)

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 2 components:
##
##  log-likelihood   n df       BIC       ICL
##       -459.7582 116  9 -962.2986 -972.0334
##
## Clustering table:
##  1  2
## 53 63
```

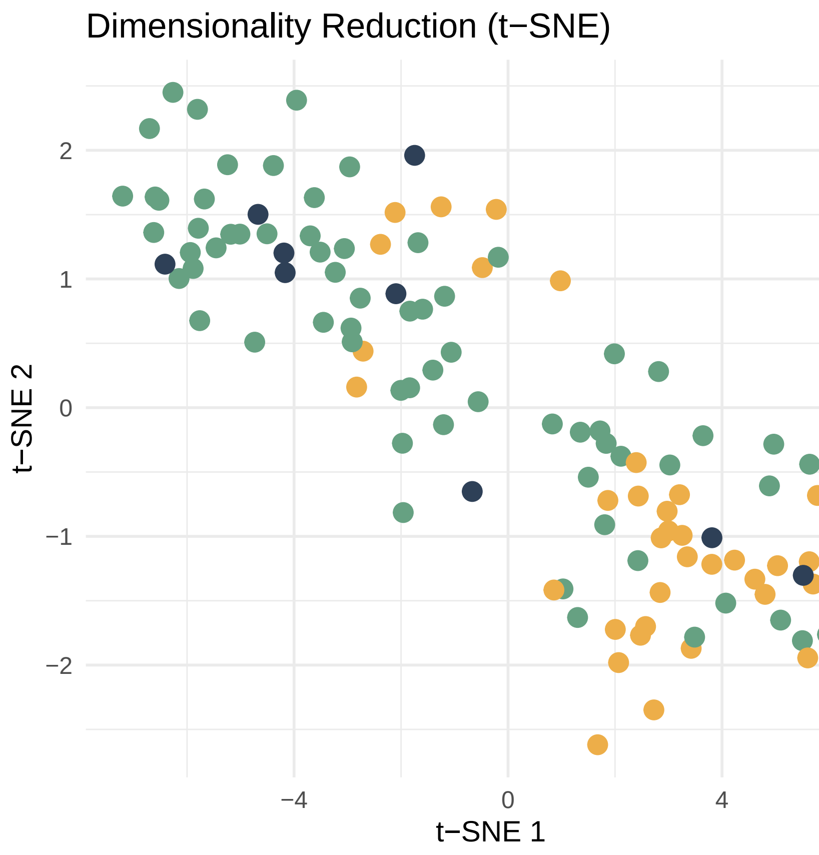
```r
# Plot the clustering results
plot(um_gmm, "classification")
```



```r
plot(um_gmm, "density")
```



```r
set.seed(1848)
# Apply t-SNE on the dataset, excluding the first column (assumed to be Infectious status)
tsne <- Rtsne::Rtsne(data1[, -c(1)], perplexity = 37, normalize=FALSE)

# Convert t-SNE results into a data frame
tsne_df <- data.frame(tsne$Y)
tsne_df$Group <- as.factor(data1$Infection_0_1pre_2post)
tsne_df$ID <- rownames(data1)

# Plot t-SNE projection
ggplot(tsne_df, aes(x = X1, y = X2, color = Group)) +
  geom_point(size = 3) +
  scale_color_manual(labels = c("No Infection", "Infection pre-boost",
                                "Infection post-boost"),
                     values = c("#66a182", "#2e4057", "#edae49")) +
  labs(x = 't-SNE 1', y = 't-SNE 2', title = 'Dimensionality Reduction (t-SNE)') +
  theme_minimal()
```



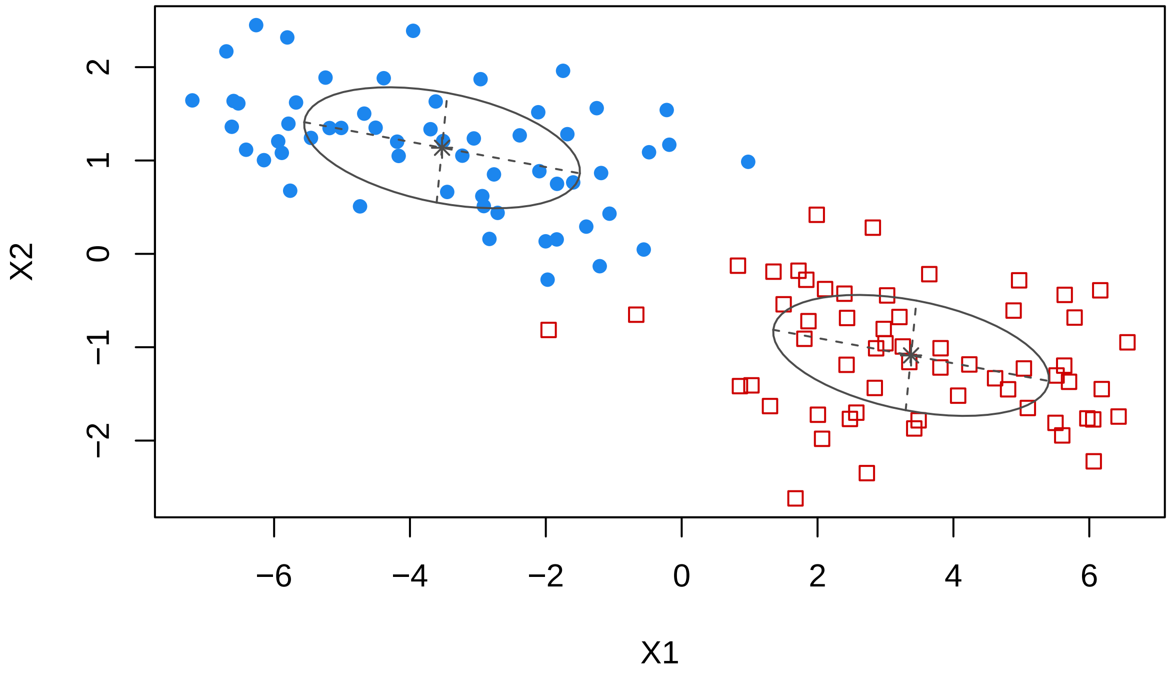
0.2.0.4 t-SNE for Dimensionality Reduction

```r
# Clustering with Gaussian Mixture Model (GMM) on t-SNE-reduced data (unsupervised)
t_gmm <- mclust::Mclust(tsne_df[, c(1, 2)])

# Print summary of clustering results
summary(t_gmm)
```

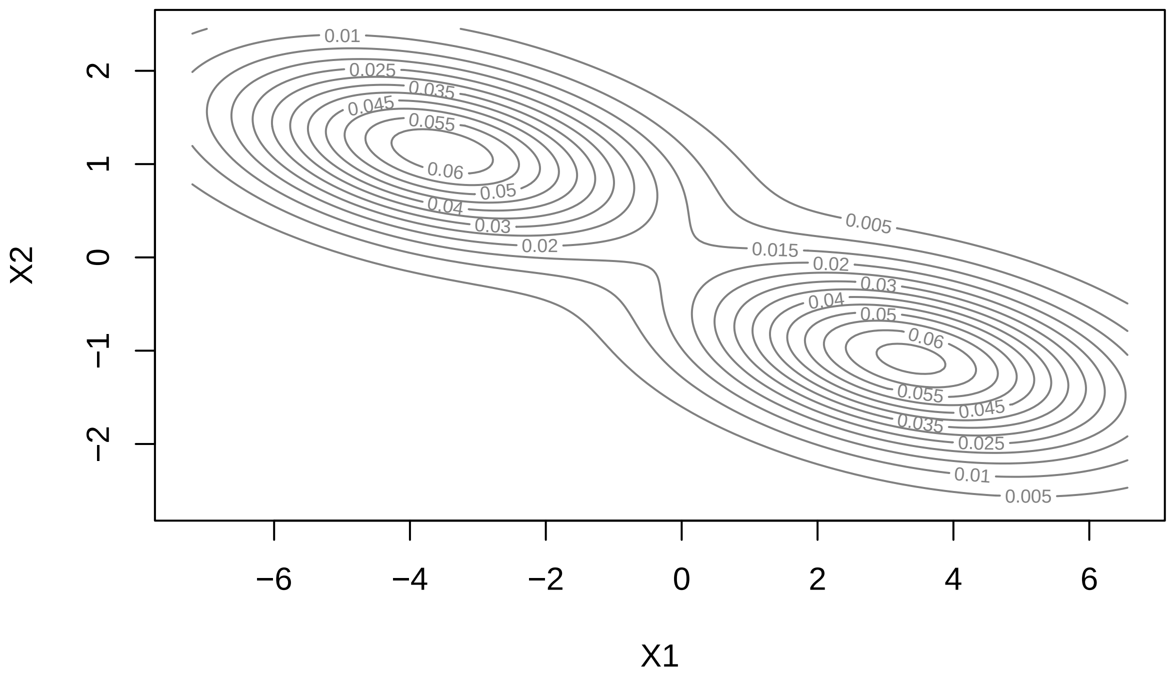
0.2.0.5 Clustering with Gaussian Mixture Model (GMM)

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEE (ellipsoidal, equal volume, shape and orientation) model with 2
## components:
##
##  log-likelihood   n df       BIC       ICL
##       -425.0104 116  8 -888.0496 -893.6646
##
## Clustering table:
##  1  2
## 57 59
```

```r
# Plot the clustering results
plot(t_gmm, "classification")
```



```r
plot(t_gmm, "density")
```



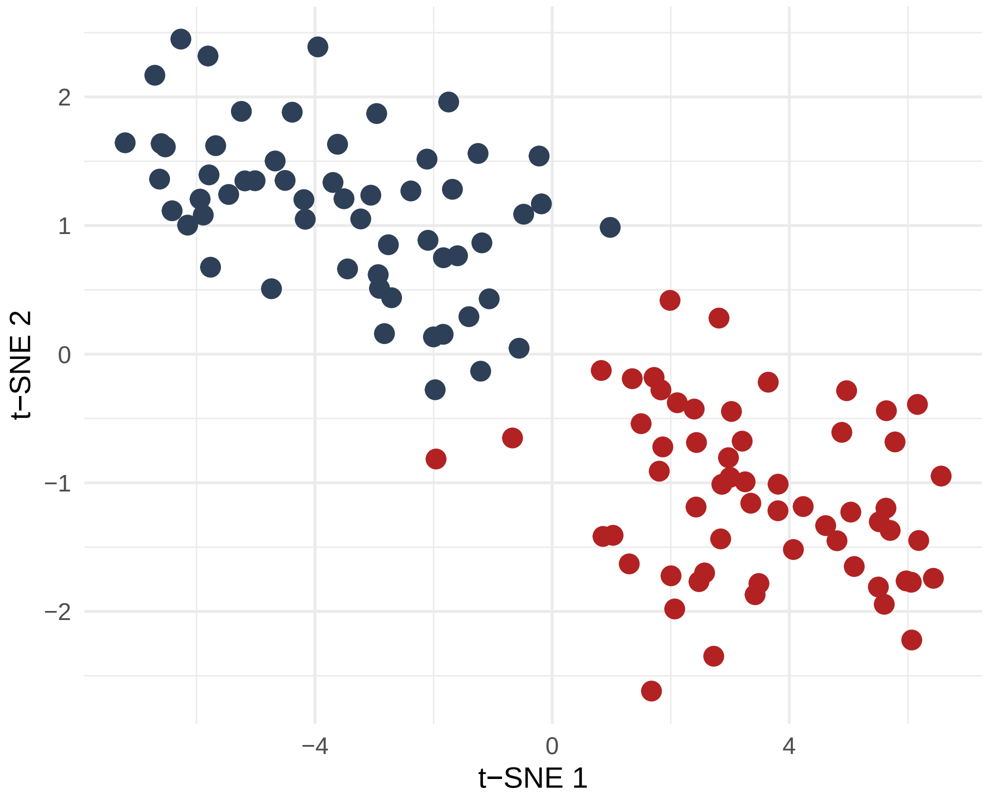
```r
# Add clustering results to the t-SNE data frame
tsne_df$Cluster <- as.factor(t_gmm$classification)
```

```r
# Compute clustering statistics for UMAP-based clustering
cs_um_gmm <- fpc::cluster.stats(dist(umap_df[1:2]), um_gmm$classification)
stats_um_gmm <- cs_um_gmm[c("within.cluster.ss","avg.silwidth")]

# Compute clustering statistics for t-SNE-based clustering
cs_ts_gmm <- fpc::cluster.stats(dist(tsne_df[1:2]), t_gmm$classification)
stats_t_gmm <- cs_ts_gmm[c("within.cluster.ss","avg.silwidth")]

# Combine statistics and print Comparison
stats <- rbind(stats_um_gmm, stats_t_gmm)
rownames(stats) <- c("UMAP", "t-SNE")
stats <- as.data.frame(stats)
stats
```

0.2.0.6 Clustering Evaluation: Within-cluster Sum of Squares & Silhouette Score

```
##       within.cluster.ss avg.silwidth
## UMAP           693.3975    0.5673657
## t-SNE           494.391    0.6305865
```

```r
# Plot t-SNE projection with clustering results as the best-performing method
ggplot(tsne_df, aes(x = X1, y = X2, color = Cluster)) +
  geom_point(size = 3) +
  scale_color_manual(values = c("#2e4057", "firebrick"),
                     labels = c("Cluster 1", "Cluster 2"),
                     name = "GMM Clusters"
                     ) +
  labs(x = 't-SNE 1', y = 't-SNE 2', title = '') +
  theme_minimal()
```



0.2.0.7 Cluster Visualization

0.3 Part 2: Unaware Infection Prediction (Classification Task)

```r
# Load the labeled dataset (data2) for Model Construction phase
load("../data/data2.rda")
# Ensure the target variable is a factor
data2$Class <- as.factor(data2$Class)
# Use ID as row names
rownames(data2) <- data2$ID
```

```r
data2$ID <- NULL
# Transform the data: log2 transformation and scaling
data2[, c(1:8, 13)] <- log2(data2[, c(1:8, 13)] + 1)
# scale the data
data2[, c(1:13)] <- scale(data2[, c(1:13)], center = T, scale = T)

# Display basic dataset structure and summary
str(data2)
```

0.3.0.1 Load and pre-process Data

```
## 'data.frame':    34 obs. of  14 variables:
##  $ wt-spike specific IgG (ng/ml)          : num  1.0048 1.1777 -0.0765 -0.4587 0.4829 ...
##  $ Delta-spike specific IgG (ng/ml)       : num  0.973 1.2 0.168 -0.352 0.479 ...
##  $ Omicron BA.1-spike specific IgG (ng/ml): num  0.76025 0.99163 -0.00842 -0.78798 0.35723 ...
##  $ Omicron BA.2-spike specific IgG (ng/ml): num  1.111 1.233 -0.137 -0.419 0.452 ...
##  $ wt-RBD specific IgG (ng/ml)            : num  0.30867 0.94031 -0.00804 -0.24737 0.71896 ...
##  $ Delta-RBD specific IgG (ng/ml)         : num  0.41904 1.00071 -0.00971 -0.54319 0.53717 ...
##  $ Omicron BA.1-RBD specific IgG (ng/ml)  : num  0.6418 0.9517 -0.0289 -0.3502 0.896 ...
##  $ Omicron BA.2-RBD specific IgG (ng/ml)  : num  0.531 0.801 -0.228 -0.282 0.793 ...
##  $ ACE2/wt RBD binding inhibition (%)     : num  0.338 0.338 0.338 0.338 0.338 ...
##  $ ACE2/Delta RBD binding inhibition (%)  : num  0.467 0.467 0.467 0.392 0.467 ...
##  $ ACE2/BA.1 RBD binding inhibition (%)   : num  1.149 1.203 -1.201 0.455 0.722 ...
##  $ ACE2/BA.2 RBD binding inhibition (%)   : num  0.7412 0.7412 -0.0428 0.5918 0.6665 ...
##  $ BA.2 N-specific IgG (AUC log)          : num  0.647 0.759 -0.749 2.326 1.074 ...
##  $ Class                                  : Factor w/ 2 levels "mcI","mcNI": 1 1 2 1 1 1 1 2 2 2 ...
```

```r
summary(data2)
```

```
##  wt-spike specific IgG (ng/ml) Delta-spike specific IgG (ng/ml)
##  Min.   :-2.0146               Min.   :-2.2006
##  1st Qu.:-0.7670               1st Qu.:-0.6964
##  Median : 0.2199               Median : 0.1617
##  Mean   : 0.0000               Mean   : 0.0000
##  3rd Qu.: 0.7621               3rd Qu.: 0.7602
##  Max.   : 1.6915               Max.   : 1.7304
##  Omicron BA.1-spike specific IgG (ng/ml)
##  Min.   :-2.1883
##  1st Qu.:-0.7992
##  Median : 0.1365
##  Mean   : 0.0000
##  3rd Qu.: 0.7479
##  Max.   : 1.6340
##  Omicron BA.2-spike specific IgG (ng/ml) wt-RBD specific IgG (ng/ml)
##  Min.   :-1.9769                         Min.   :-2.0299
##  1st Qu.:-0.7884                         1st Qu.:-0.6716
##  Median : 0.1306                         Median : 0.2376
##  Mean   : 0.0000                         Mean   : 0.0000
##  3rd Qu.: 0.6174                         3rd Qu.: 0.7056
##  Max.   : 1.6412                         Max.   : 2.0621
##  Delta-RBD specific IgG (ng/ml) Omicron BA.1-RBD specific IgG (ng/ml)
##  Min.   :-1.9696                Min.   :-1.7103
##  1st Qu.:-0.6503                1st Qu.:-0.5869
##  Median : 0.3395               Median : 0.3333
##  Mean   : 0.0000               Mean   : 0.0000
##  3rd Qu.: 0.6497               3rd Qu.: 0.6368
##  Max.   : 1.7894               Max.   : 2.0401
##  Omicron BA.2-RBD specific IgG (ng/ml) ACE2/wt RBD binding inhibition (%)
##  Min.   :-2.5838                       Min.   :-4.9959
##  1st Qu.:-0.5200                       1st Qu.: 0.3376
##  Median : 0.1383                       Median : 0.3376
##  Mean   : 0.0000                       Mean   : 0.0000
##  3rd Qu.: 0.5741                       3rd Qu.: 0.3376
##  Max.   : 1.8811                       Max.   : 0.3376
##  ACE2/Delta RBD binding inhibition (%) ACE2/BA.1 RBD binding inhibition (%)
##  Min.   :-3.1111                       Min.   :-1.4146
##  1st Qu.: 0.3179                       1st Qu.:-1.1876
##  Median : 0.4670                       Median : 0.4281
##  Mean   : 0.0000                       Mean   : 0.0000
##  3rd Qu.: 0.4670                       3rd Qu.: 0.8354
##  Max.   : 0.4670                       Max.   : 1.2026
##  ACE2/BA.2 RBD binding inhibition (%) BA.2 N-specific IgG (AUC log)  Class
##  Min.   :-2.9174                      Min.   :-1.0484              mcI :18
##  1st Qu.:-0.4815                      1st Qu.:-0.8539              mcNI:16
##  Median : 0.5918                      Median :-0.2945
##  Mean   : 0.0000                      Mean   : 0.0000
##  3rd Qu.: 0.6665                      3rd Qu.: 0.7308
##  Max.   : 0.7412                      Max.   : 2.3256
```

```r
# Set a global seed
set.seed(1939)

# Create 5 cross-validation folds to use for all models
cv_folds <- caret::createFolds(data2$Class, k = 5, returnTrain = TRUE)

# Define training control settings
control <- caret::trainControl(
  method = "cv", number = 5,
  classProbs = TRUE,
  summaryFunction = multiClassSummary,
  verboseIter = TRUE,
  index = cv_folds
)
```

```r
# Define a function for hyper parameter tuning and training models
train_model <- function(method, tL) {
  caret::train(Class ~ ., data = data2, method = method,
               trControl = control, tuneLength = tL)
}

# Train models with hyperparameter tuning
knn_model <- train_model("knn", tL = 5)
```

0.3.0.2 Machine Learning Models Construction

```
## + Fold1: k= 5
## - Fold1: k= 5
## + Fold1: k= 7
## - Fold1: k= 7
## + Fold1: k= 9
## - Fold1: k= 9
## + Fold1: k=11
## - Fold1: k=11
## + Fold1: k=13
## - Fold1: k=13
## + Fold2: k= 5
## - Fold2: k= 5
## + Fold2: k= 7
## - Fold2: k= 7
## + Fold2: k= 9
## - Fold2: k= 9
## + Fold2: k=11
## - Fold2: k=11
## + Fold2: k=13
## - Fold2: k=13
## + Fold3: k= 5
## - Fold3: k= 5
## + Fold3: k= 7
## - Fold3: k= 7
## + Fold3: k= 9
## - Fold3: k= 9
## + Fold3: k=11
## - Fold3: k=11
## + Fold3: k=13
## - Fold3: k=13
## + Fold4: k= 5
## - Fold4: k= 5
## + Fold4: k= 7
## - Fold4: k= 7
## + Fold4: k= 9
## - Fold4: k= 9
## + Fold4: k=11
## - Fold4: k=11
## + Fold4: k=13
## - Fold4: k=13
## + Fold5: k= 5
## - Fold5: k= 5
## + Fold5: k= 7
## - Fold5: k= 7
## + Fold5: k= 9
## - Fold5: k= 9
## + Fold5: k=11
## - Fold5: k=11
## + Fold5: k=13
## - Fold5: k=13
## Aggregating results
## Selecting tuning parameters
## Fitting k = 5 on full training set
```

```r
rf_model  <- train_model("rf", tL = 5)
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry= 4
## - Fold1: mtry= 4
## + Fold1: mtry= 7
## - Fold1: mtry= 7
## + Fold1: mtry=10
## - Fold1: mtry=10
## + Fold1: mtry=13
## - Fold1: mtry=13
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry= 4
## - Fold2: mtry= 4
## + Fold2: mtry= 7
## - Fold2: mtry= 7
## + Fold2: mtry=10
## - Fold2: mtry=10
## + Fold2: mtry=13
## - Fold2: mtry=13
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry= 4
## - Fold3: mtry= 4
## + Fold3: mtry= 7
## - Fold3: mtry= 7
## + Fold3: mtry=10
## - Fold3: mtry=10
## + Fold3: mtry=13
## - Fold3: mtry=13
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry= 4
## - Fold4: mtry= 4
## + Fold4: mtry= 7
## - Fold4: mtry= 7
## + Fold4: mtry=10
## - Fold4: mtry=10
## + Fold4: mtry=13
## - Fold4: mtry=13
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry= 4
## - Fold5: mtry= 4
## + Fold5: mtry= 7
## - Fold5: mtry= 7
## + Fold5: mtry=10
## - Fold5: mtry=10
## + Fold5: mtry=13
## - Fold5: mtry=13
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 7 on full training set
```

```
svm_model <- train_model("svmRadial", tL = 5)
```

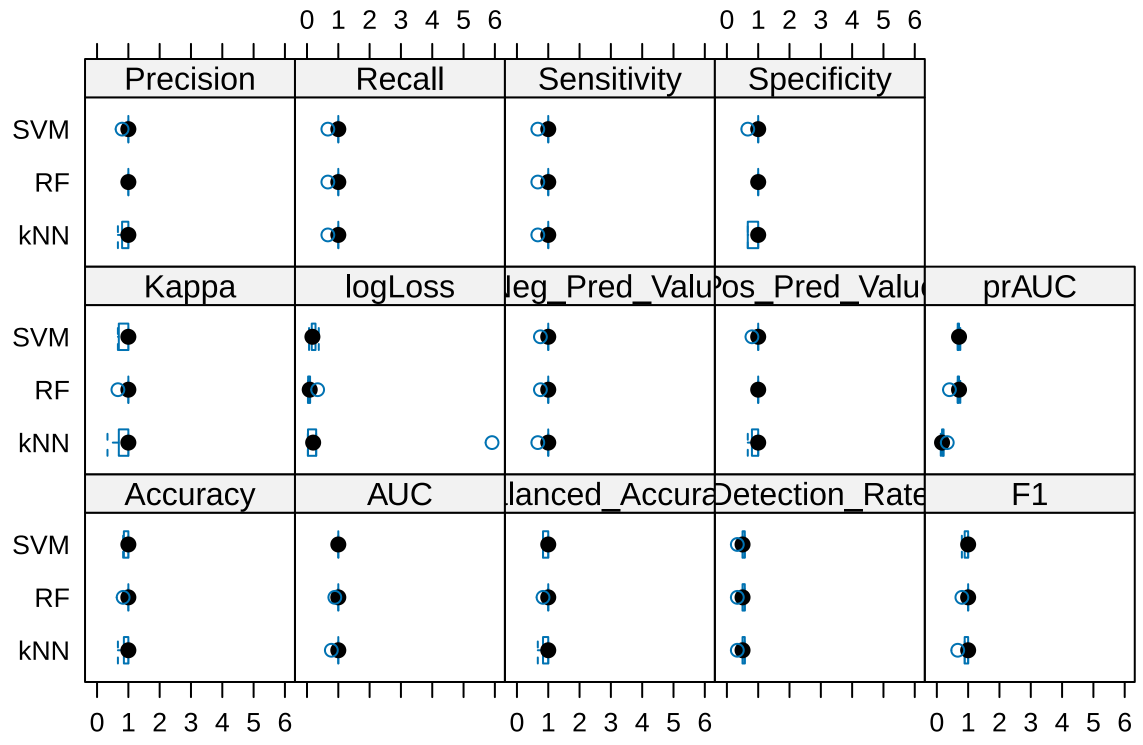
```
## + Fold1: sigma=0.2245, C=0.25
## - Fold1: sigma=0.2245, C=0.25
## + Fold1: sigma=0.2245, C=0.50
## - Fold1: sigma=0.2245, C=0.50
## + Fold1: sigma=0.2245, C=1.00
## - Fold1: sigma=0.2245, C=1.00
## + Fold1: sigma=0.2245, C=2.00
## - Fold1: sigma=0.2245, C=2.00
## + Fold1: sigma=0.2245, C=4.00
## - Fold1: sigma=0.2245, C=4.00
## + Fold2: sigma=0.2245, C=0.25
## - Fold2: sigma=0.2245, C=0.25
## + Fold2: sigma=0.2245, C=0.50
## - Fold2: sigma=0.2245, C=0.50
## + Fold2: sigma=0.2245, C=1.00
## - Fold2: sigma=0.2245, C=1.00
## + Fold2: sigma=0.2245, C=2.00
## - Fold2: sigma=0.2245, C=2.00
## + Fold2: sigma=0.2245, C=4.00
## - Fold2: sigma=0.2245, C=4.00
## + Fold3: sigma=0.2245, C=0.25
## - Fold3: sigma=0.2245, C=0.25
## + Fold3: sigma=0.2245, C=0.50
## - Fold3: sigma=0.2245, C=0.50
## + Fold3: sigma=0.2245, C=1.00
## - Fold3: sigma=0.2245, C=1.00
## + Fold3: sigma=0.2245, C=2.00
## - Fold3: sigma=0.2245, C=2.00
## + Fold3: sigma=0.2245, C=4.00
## - Fold3: sigma=0.2245, C=4.00
## + Fold4: sigma=0.2245, C=0.25
## - Fold4: sigma=0.2245, C=0.25
## + Fold4: sigma=0.2245, C=0.50
## - Fold4: sigma=0.2245, C=0.50
## + Fold4: sigma=0.2245, C=1.00
## - Fold4: sigma=0.2245, C=1.00
## + Fold4: sigma=0.2245, C=2.00
## - Fold4: sigma=0.2245, C=2.00
## + Fold4: sigma=0.2245, C=4.00
## - Fold4: sigma=0.2245, C=4.00
## + Fold5: sigma=0.2245, C=0.25
## - Fold5: sigma=0.2245, C=0.25
## + Fold5: sigma=0.2245, C=0.50
## - Fold5: sigma=0.2245, C=0.50
## + Fold5: sigma=0.2245, C=1.00
## - Fold5: sigma=0.2245, C=1.00
## + Fold5: sigma=0.2245, C=2.00
## - Fold5: sigma=0.2245, C=2.00
## + Fold5: sigma=0.2245, C=4.00
## - Fold5: sigma=0.2245, C=4.00
## Aggregating results
## Selecting tuning parameters
## Fitting sigma = 0.224, C = 4 on full training set
```

```
# Collect and compare model results
results <- resamples(list(kNN = knn_model, RF = rf_model, SVM = svm_model))

# Select metrics of interest
selected_metrics <- results$values %>%
  select(contains(c("Accuracy", "Precision", "Recall", "F1")))

# Display performance summary
summary(selected_metrics)
```

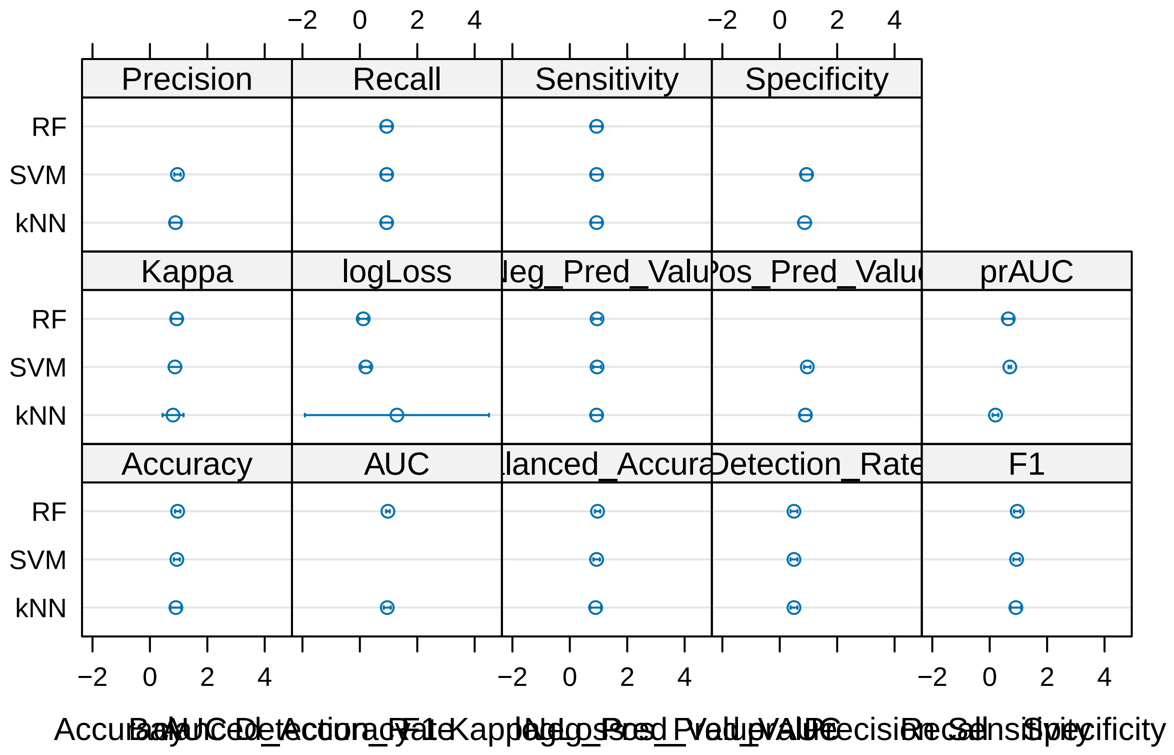
0.3.0.3 Model Performance Evaluation

```
##   kNN~Accuracy    kNN~Balanced_Accuracy  RF~Accuracy     RF~Balanced_Accuracy
##  Min.   :0.6667   Min.   :0.6667        Min.   :0.8333   Min.   :0.8333
##  1st Qu.:0.8571   1st Qu.:0.8333        1st Qu.:1.0000   1st Qu.:1.0000
##  Median :1.0000   Median :1.0000        Median :1.0000   Median :1.0000
##  Mean   :0.9048   Mean   :0.9000        Mean   :0.9667   Mean   :0.9667
##  3rd Qu.:1.0000   3rd Qu.:1.0000        3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000        Max.   :1.0000   Max.   :1.0000
##   SVM~Accuracy    SVM~Balanced_Accuracy kNN~Precision     RF~Precision
##  Min.   :0.8333   Min.   :0.8333        Min.   :0.6667   Min.   :1
##  1st Qu.:0.8571   1st Qu.:0.8333        1st Qu.:0.8000   1st Qu.:1
##  Median :1.0000   Median :1.0000        Median :1.0000   Median :1
##  Mean   :0.9381   Mean   :0.9333        Mean   :0.8933   Mean   :1
##  3rd Qu.:1.0000   3rd Qu.:1.0000        3rd Qu.:1.0000   3rd Qu.:1
##  Max.   :1.0000   Max.   :1.0000        Max.   :1.0000   Max.   :1
##  SVM~Precision    kNN~Recall       RF~Recall        SVM~Recall
##  Min.   :0.80   Min.   :0.6667   Min.   :0.6667   Min.   :0.6667
##  1st Qu.:1.00   1st Qu.:1.0000   1st Qu.:1.0000   1st Qu.:1.0000
##  Median :1.00   Median :1.0000   Median :1.0000   Median :1.0000
##  Mean   :0.96   Mean   :0.9333   Mean   :0.9333   Mean   :0.9333
##  3rd Qu.:1.00   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      kNN~F1           RF~F1          SVM~F1
##  Min.   :0.6667   Min.   :0.80   Min.   :0.8000
##  1st Qu.:0.8889   1st Qu.:1.00   1st Qu.:0.8889
##  Median :1.0000   Median :1.00   Median :1.0000
##  Mean   :0.9111   Mean   :0.96   Mean   :0.9378
##  3rd Qu.:1.0000   3rd Qu.:1.00   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00   Max.   :1.0000
```

```
# Visualization of model performance
bwplot(results)
```

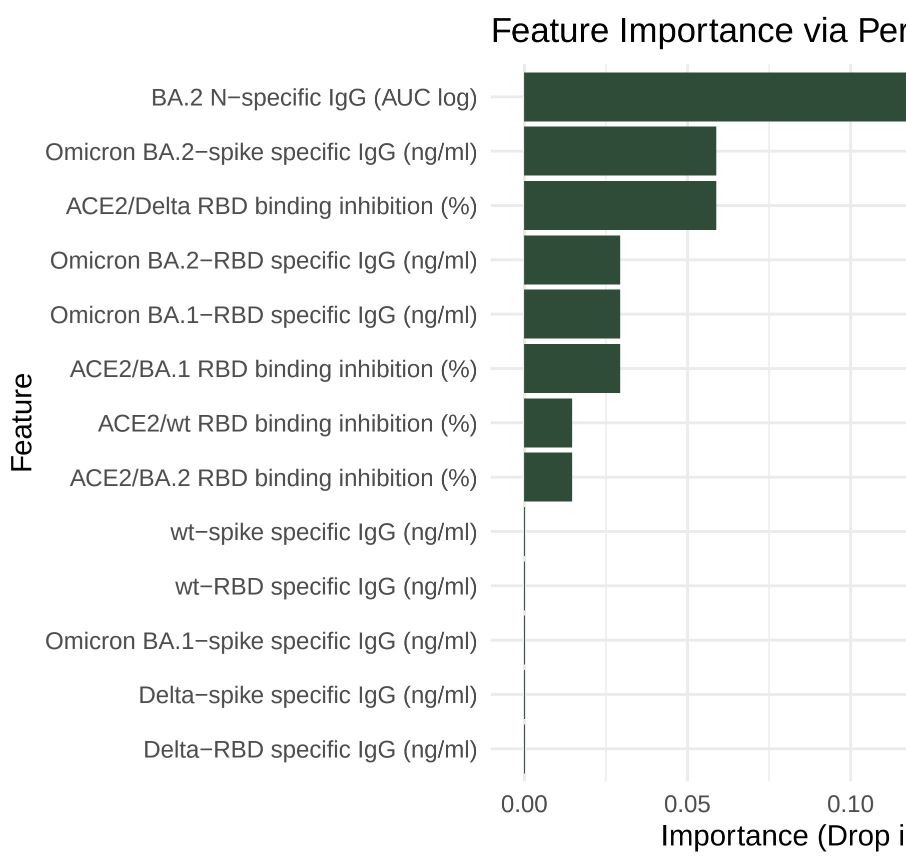


```r
dotplot(results)
```



Confidence Level: 0.95

```r
# Define a function to compute permutation-based feature importance for SVM and k-NN
permute_importance <- function(model, data, target_col, metric = "Accuracy",
                               n_permutations = 10, parallel = TRUE) {
  set.seed(0306)

  y <- data[[target_col]]
  if (!is.factor(y)) y <- as.factor(y)  # Converte il target in fattore se necessario
  X <- data[, colnames(data) != target_col, drop = FALSE]

  # Check input data
  stopifnot(is.data.frame(data))
  stopifnot(target_col %in% colnames(data))
  stopifnot(nrow(data) > 10)

  # Compute original accuracy
  original_preds <- predict(model, newdata = X)
  if (is.numeric(original_preds)) {
    original_preds <- ifelse(original_preds > 0.5, levels(y)[2], levels(y)[1])}
  original_acc <- mean(original_preds == y)

  # Prepare data frame to store importances
  importances <- data.frame(Feature = colnames(X), Importance = 0)
  # Allow parallel computation
  if (parallel) {
    registerDoParallel(cores = detectCores() - 1)}

  # Loop on each feature and compute importances
  results <- foreach(feature = colnames(X), .combine = rbind, .packages = "caret") %dopar% {
    acc_drops <- numeric(n_permutations)

    for (i in 1:n_permutations) {
      X_permuted <- X
      X_permuted[[feature]] <- sample(na.omit(X_permuted[[feature]]), replace = TRUE)

      permuted_preds <- predict(model, newdata = X_permuted)
      permuted_acc <- mean(permuted_preds == y)

      acc_drops[i] <- original_acc - permuted_acc}
    data.frame(Feature = feature, Importance = median(acc_drops))}

  # End parallel computation
  if (parallel) {
    stopImplicitCluster()}

  # Organize results
  results <- results[order(-results$Importance), ]

  # Plot importances
  p <- ggplot(results, aes(x = reorder(Feature, Importance), y = Importance)) +
    geom_col(fill = "#2F4C39") +
    coord_flip() +
    labs(title = "Feature Importance via Permutation",
         x = "Feature",
         y = "Importance (Drop in Accuracy)") +
    theme_minimal()
  print(p)

  return(results)
}

# Compute permutation-based importances for SVM-Radial
importance_svm <- permute_importance(svm_model, data2, "Class", metric = "Accuracy", n_permutations = 1
```

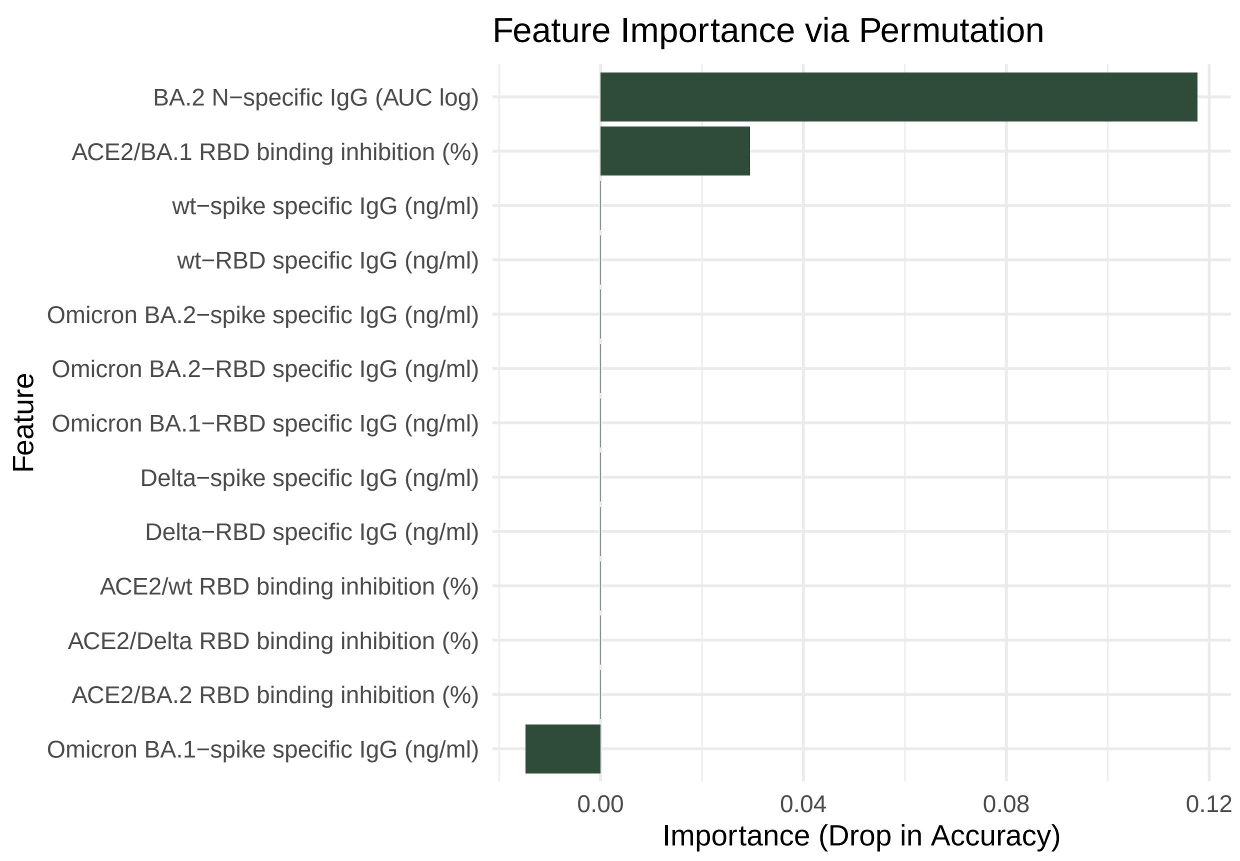


0.3.0.4 Variable Importance Analysis

```
print(importance_svm)
```

```
##                                    Feature Importance
## 13           BA.2 N-specific IgG (AUC log) 0.20588235
## 4  Omicron BA.2-spike specific IgG (ng/ml) 0.05882353
## 10   ACE2/Delta RBD binding inhibition (%) 0.05882353
## 7    Omicron BA.1-RBD specific IgG (ng/ml) 0.02941176
## 8    Omicron BA.2-RBD specific IgG (ng/ml) 0.02941176
## 11    ACE2/BA.1 RBD binding inhibition (%) 0.02941176
## 9       ACE2/wt RBD binding inhibition (%) 0.01470588
## 12    ACE2/BA.2 RBD binding inhibition (%) 0.01470588
## 1            wt-spike specific IgG (ng/ml) 0.00000000
## 2         Delta-spike specific IgG (ng/ml) 0.00000000
## 3  Omicron BA.1-spike specific IgG (ng/ml) 0.00000000
## 5              wt-RBD specific IgG (ng/ml) 0.00000000
## 6           Delta-RBD specific IgG (ng/ml) 0.00000000
```

```
# Compute permutation-based importances for k-NN
importance_knn <- permute_importance(knn_model, data2, "Class", metric = "Accuracy", n_permutations = 3
```

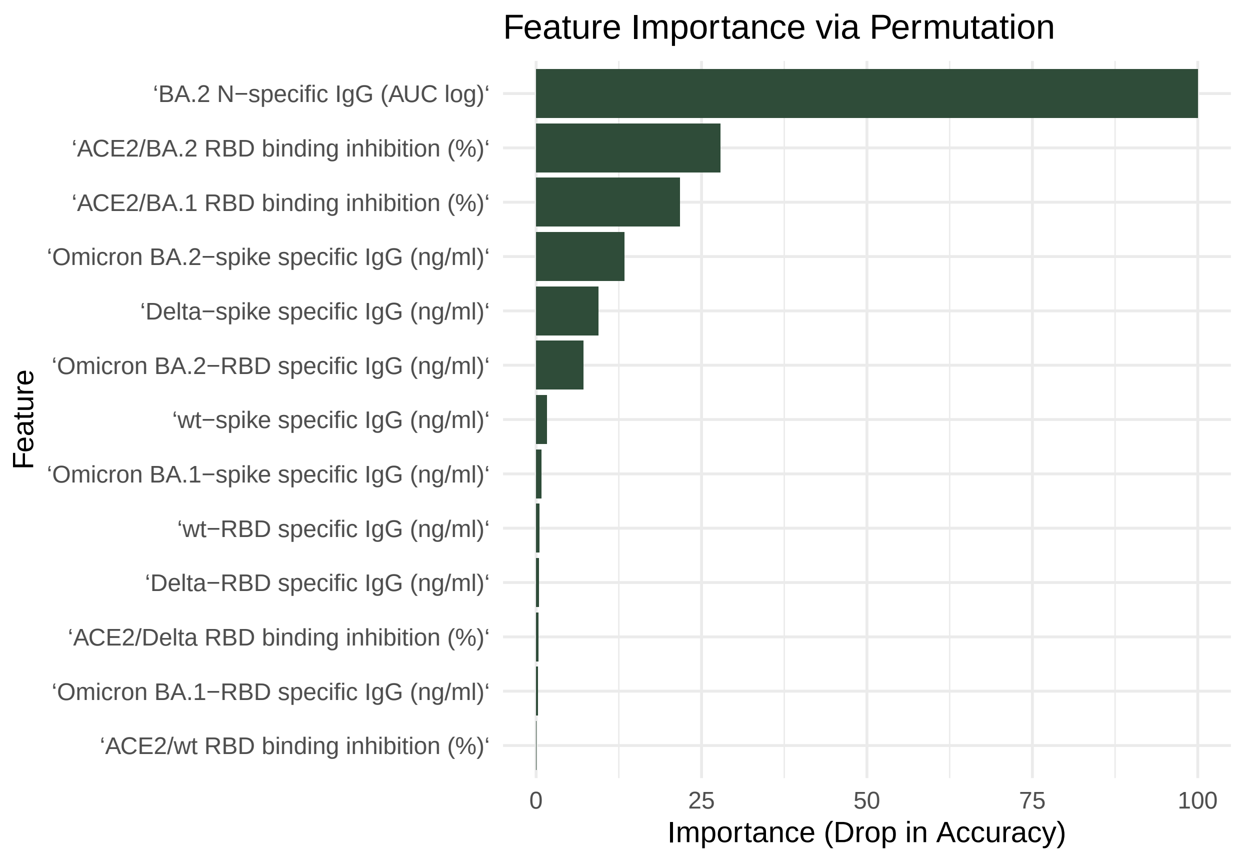


```
print(importance_knn)
```

```
##                                     Feature  Importance
## 13           BA.2 N-specific IgG (AUC log)  0.11764706
## 11     ACE2/BA.1 RBD binding inhibition (%)  0.02941176
## 1            wt-spike specific IgG (ng/ml)  0.00000000
## 2         Delta-spike specific IgG (ng/ml)  0.00000000
## 4  Omicron BA.2-spike specific IgG (ng/ml)  0.00000000
## 5              wt-RBD specific IgG (ng/ml)  0.00000000
## 6           Delta-RBD specific IgG (ng/ml)  0.00000000
## 7    Omicron BA.1-RBD specific IgG (ng/ml)  0.00000000
## 8    Omicron BA.2-RBD specific IgG (ng/ml)  0.00000000
## 9       ACE2/wt RBD binding inhibition (%)  0.00000000
## 10   ACE2/Delta RBD binding inhibition (%)  0.00000000
## 12     ACE2/BA.2 RBD binding inhibition (%)  0.00000000
## 3  Omicron BA.1-spike specific IgG (ng/ml) -0.01470588
```

```
# Compute vip based importances for Random Forest
importance_rf <- caret::varImp(rf_model, scale = TRUE)
importance_rf <- importance_rf$importance
importance_rf$Feature <- rownames(importance_rf)
ggplot(importance_rf, aes(x = reorder(Feature, Overall), y = Overall)) +
  geom_col(fill = "#2F4C39") +
  coord_flip() +
  labs(title = "Feature Importance via Permutation",
```

```r
       x = "Feature",
       y = "Importance (Drop in Accuracy)") +
  theme_minimal()
```



```r
print(importance_rf)
```

```
##                                                 Overall
## `wt-spike specific IgG (ng/ml)`               1.6288847
## `Delta-spike specific IgG (ng/ml)`            9.3904108
## `Omicron BA.1-spike specific IgG (ng/ml)`     0.8218825
## `Omicron BA.2-spike specific IgG (ng/ml)`    13.3490968
## `wt-RBD specific IgG (ng/ml)`                 0.4546349
## `Delta-RBD specific IgG (ng/ml)`              0.4503599
## `Omicron BA.1-RBD specific IgG (ng/ml)`       0.2931158
## `Omicron BA.2-RBD specific IgG (ng/ml)`       7.1791209
## `ACE2/wt RBD binding inhibition (%)`          0.0000000
## `ACE2/Delta RBD binding inhibition (%)`       0.3527592
## `ACE2/BA.1 RBD binding inhibition (%)`       21.6929332
## `ACE2/BA.2 RBD binding inhibition (%)`       27.8112087
## `BA.2 N-specific IgG (AUC log)`             100.0000000
##                                                                              Feature
## `wt-spike specific IgG (ng/ml)`                         `wt-spike specific IgG (ng/ml)`
## `Delta-spike specific IgG (ng/ml)`                   `Delta-spike specific IgG (ng/ml)`
## `Omicron BA.1-spike specific IgG (ng/ml)` `Omicron BA.1-spike specific IgG (ng/ml)`
## `Omicron BA.2-spike specific IgG (ng/ml)` `Omicron BA.2-spike specific IgG (ng/ml)`
## `wt-RBD specific IgG (ng/ml)`                           `wt-RBD specific IgG (ng/ml)`
## `Delta-RBD specific IgG (ng/ml)`                     `Delta-RBD specific IgG (ng/ml)`
## `Omicron BA.1-RBD specific IgG (ng/ml)`     `Omicron BA.1-RBD specific IgG (ng/ml)`
## `Omicron BA.2-RBD specific IgG (ng/ml)`     `Omicron BA.2-RBD specific IgG (ng/ml)`
## `ACE2/wt RBD binding inhibition (%)`           `ACE2/wt RBD binding inhibition (%)`
## `ACE2/Delta RBD binding inhibition (%)`     `ACE2/Delta RBD binding inhibition (%)`
## `ACE2/BA.1 RBD binding inhibition (%)`       `ACE2/BA.1 RBD binding inhibition (%)`
## `ACE2/BA.2 RBD binding inhibition (%)`       `ACE2/BA.2 RBD binding inhibition (%)`
## `BA.2 N-specific IgG (AUC log)`                       `BA.2 N-specific IgG (AUC log)`
```

```r
# Load unlabeled data (data3) for Model Application phase
load("../data/data3.rda")
rownames(data3) <- data3$ID
data3$ID <- NULL
# save ID and self-declared class for later
self_df <- data.frame(self_status = data3$Infection_0_1pre_2post,
                      row.names = rownames(data3))
data3$Infection_0_1pre_2post <- NULL

data3[, c(1:8, 13)] <- log2(data3[, c(1:8, 13)] + 1)
# scale the data
data3[, c(1:13)] <- scale(data3[, c(1:13)], center = T, scale = T)

# Ensure feature consistency between data2 and data3
common_features <- intersect(names(data2), names(data3))
data3 <- data3[, common_features, drop = FALSE]

# Make predictions using trained models
self_df$kNN <- predict(knn_model, data3)
self_df$RF  <- predict(rf_model, data3)
self_df$SVM <- predict(svm_model, data3)

# Compare model predictions
table(self_df$kNN)
```

0.3.0.5 Model Application: Prediction on Unlabelled Data

```
##
##  mcI mcNI
##   29   28
```

```r
table(self_df$RF)
```

```
##
##  mcI mcNI
##   36   21
```

```r
table(self_df$SVM)
```

```
##
##  mcI mcNI
##   28   29
```

```r
# Clean up the predictions by removing "mc" prefix and find consensus
self_df <- self_df %>%
  mutate(
    kNN = str_remove(kNN, "mc"),
    RF = str_remove(RF, "mc"),
    SVM = str_remove(SVM, "mc"),
    consensus = pmap_chr(list(kNN, RF, SVM), function(a, b, c) {
      votes <- c(a, b, c)
      vote_table <- table(votes)
      winner <- names(vote_table)[which.max(vote_table)]
      if (max(vote_table) >= 2) winner else NA_character_
    })
  )
```

0.3.0.6 Model Consensus

```r
# Compare new predictions with self-declared status
self_df <- self_df %>%
  # convert self_status labels
  mutate(self_status = ifelse(self_status %in% c(1, 2), "I", "NI")) %>%

  # Assign new class based on self-declared status and consensus results
  mutate(new_class = case_when(
    self_status == "NI" & consensus == "NI" ~ "NI",
    self_status == "NI" & consensus == "I"  ~ "UI",
    self_status == "I"  & consensus == "I"  ~ "I",
    self_status == "I"  & consensus == "NI" ~ "exclude",
    TRUE ~ NA_character_  # for NAs or other cases
  ))

# Display the final classification results
table(self_df$new_class)
```

0.3.0.7 Assign predicted Labels

```
##
## exclude       I      NI      UI
##       2      16      25      14
```

0.4 Session Informations

```r
sessionInfo()
```

```
## R version 4.5.0 (2025-04-11)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sequoia 15.4.1
```

```
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Rome
## tzcode source: internal
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] purrr_1.0.4       stringr_1.5.1     MLmetrics_1.1.3   doParallel_1.0.17
##  [5] iterators_1.0.14  foreach_1.5.2     dplyr_1.1.4       caret_7.0-1
##  [9] lattice_0.22-7    fpc_2.2-13        ggplot2_3.5.2     Rtsne_0.17
## [13] umap_0.2.10.0     mclust_6.1.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1     timeDate_4041.110    farver_2.1.2
##  [4] fastmap_1.2.0        pROC_1.18.5          digest_0.6.37
##  [7] rpart_4.1.24         timechange_0.3.0     lifecycle_1.0.4
## [10] cluster_2.1.8.1      survival_3.8-3       ROCR_1.0-11
## [13] magrittr_2.0.3       kernlab_0.9-33       compiler_4.5.0
## [16] rlang_1.1.6          tools_4.5.0          yaml_2.3.10
## [19] data.table_1.17.2    knitr_1.50           labeling_0.4.3
## [22] askpass_1.2.1        here_1.0.1           reticulate_1.42.0
## [25] plyr_1.8.9           RColorBrewer_1.1-3   withr_3.0.2
## [28] nnet_7.3-20          grid_4.5.0           stats4_4.5.0
## [31] e1071_1.7-16         future_1.49.0        globals_0.18.0
## [34] scales_1.4.0         MASS_7.3-65          prabclus_2.3-4
## [37] cli_3.6.5            rmarkdown_2.29       generics_0.1.4
## [40] rstudioapi_0.17.1    future.apply_1.11.3  robustbase_0.99-4-1
## [43] RSpectra_0.16-2      reshape2_1.4.4       proxy_0.4-27
## [46] modeltools_0.2-24    splines_4.5.0        vctrs_0.6.5
## [49] hardhat_1.4.1        Matrix_1.7-3         jsonlite_2.0.0
## [52] listenv_0.9.1        diptest_0.77-1       gower_1.0.2
## [55] recipes_1.3.0        glue_1.8.0           parallelly_1.44.0
## [58] DEoptimR_1.1-3-1     codetools_0.2-20     lubridate_1.9.4
## [61] stringi_1.8.7        gtable_0.3.6         tibble_3.2.1
## [64] pillar_1.10.2        htmltools_0.5.8.1    randomForest_4.7-1.2
## [67] ipred_0.9-15         openssl_2.3.2        lava_1.8.1
## [70] R6_2.6.1             rprojroot_2.0.4      evaluate_1.0.3
## [73] png_0.1-8            class_7.3-23         Rcpp_1.0.14
## [76] flexmix_2.3-20       nlme_3.1-168         prodlim_2025.04.28
## [79] xfun_0.52            pkgconfig_2.0.3      ModelMetrics_1.2.2.2
```