Machine learning classifies ferroptosis and apoptosis cell death modalities with TfR1 immunostaining

supporting information: R code

Jenny Jin, Kenji Schorpp, Daniel Samaga, Kristian Unger, Kamyar Hadian, Brent R. Stockwell

08/02/2022

Contents

Concepts

Machine Learning for Multiclass Prediction

Classification denotes the estimation of class-membership $c \in \{1, .., C\}$ of an object on basis of its attributes $x = (x_1, \ldots, x_p)$, which are given as a feature vector - or feature matrix X in case of n objects ($x_{ij} \in \mathbb{R}^{n \times p}$). In supervised machine learning, algorithms are used to extract information about class-membership from samples with known class (model discovery), which can be applied to predict class membership for samples with unknown class or to estimate quality of model performance on validation data.

Classification concepts are, for instance

- linear classifiers (i.e. The 'machine' learns the coefficients β of a generalized linear model $y = X\beta$, which fit discovery data the 'best'. y is connected to class membership, e.g. the logit (log-odds log(p/(1-p))). Those estimated coefficients $\hat{\beta}_j$ are easy to interpret and can be applied for sample classification to any new feature vector z by calculating $z^T\hat{\beta}$)

- support vector machines (i.e. a hyperplane is constructed in the high-dimensional feature space, which separates two groups the best. Extensions of this idea can be applied for multi-class prediction. However the classifier itself is almost impossible to interpret)
- k-nearest neighbors approach (i.e. classifying new objects z on the basis of their distance $d(z, x_i), i \in \{1, .., n\}$ to the objects of the discovery data. Obviously the choice of distance measure d and k are crucial)
- decision trees (i.e. during discovery, single yes/no decisions are extracted (e.g. $x_{12} < 0.4$) and used to construct a decision tree. Along this tree, new samples can be classified. Drawback: trees can be highly instable, when training samples are small with respect to number of features. Random forests create many decision trees and their prediction is more robust, drawback is, they are hard to analyse/interpret)
- neural networks (i.e. inspired by neuron layers in the brain, layers of artificial neurons are trained to predict c from x. The trained network is a black box)

Multiclass Classification

Some approaches originated in binary decisions. Logistic regression, for instance, only is applicable to two-class prediction.

Besides generalizations of the approach to multiclass prediction, like multinomial logistic regression, there ís always the option of 'one-vs-all' (OvA) or 'one-vs-one' (OvO) strategies. In OvA and OvO approaches a set of binary classificators (class c_1 vs 'the rest' and c_2 vs 'the rest' and ... ; or c_1 vs c_2 and c_1 vs c_3 and ...) is built and the set of single decisions is reduced to a single decision by a rule like 'take the class, which is chosen most often' etc.

Data Transformation

No classification tool is able to do magic tricks. Every tool is dependent on accessibility to informative aspects in the feature matrix. Therefore, it is often recommendable to consider log-transformation of positive and skewed data (e.g. Box-Cox-transformation).

Furthermore, many machine learning algorithms suffer from highly correlated features. (moreover, highly correlated features cannot carry much information, from an information theoretical point of view). Thus, unsupervised feature selection should be considered (note, supervised feature selection must be placed within k-fold-cross-validation strategies).

Another strategy is reduction of dimensionality by principal component analysis (PCA) - but here generalizability of results is affected and interpretation of the classifier at the end is tedious.

Note, that transformation steps as featurewise normalization/scaling can be pitfalls in the anticipated applicability of the classificator. Single objects cannot be normalized/scaled. Furthermore shifts in class-proportions from discovery to training data will crush classificator performance. Therefore, scaling factors and normalization coefficients must be transferred from discovery data to validation data or new data and thus are inseparable from the classifier itself (i.e. most often pointless).

Data Visualization

Visualization techniques (e.g. tSNE) are key in explorative data analysis. For classification they are mostly pointless unless there is a direct connection to the chosen classification method like linear- or quadratic discriminant analysis or clustering-based-approaches.

Regression Models

- full transparency

- easy interpretation
- toolbox of feature selection methods for two-class scenarios
- prediction of class from new independent single samples

```
#install.packages("glmnet")
rm(list = ls())
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

Recap multinomial logistic regression

Multinomial distribution (see Fahrmeir, for $q+1$ classes): $\sum_{i=1}^{q+1} \pi_i = 1$,

$$f(x_1, \ldots, x_q; \pi) = \frac{\mathbf{N!}}{\mathbf{x_1! \ldots x_q!(N - \sum_{j=1}^{q} x_j)!}} \prod_{\mathbf{i=1}}^{\mathbf{q}} \pi_{\mathbf{i}}^{\mathbf{x_i}} (\mathbf{1} - \sum_{\mathbf{j=1}}^{\mathbf{q}} \pi_{\mathbf{j}})^{\mathbf{N} - \sum_{\mathbf{j=1}}^{\mathbf{q}} \mathbf{x_j}}$$

Tutz, Poessnecker and Uhlmann 'Varable Selection in General Multinomial Logit Models': '2.1. Multinomial Logit Model with Category-Specific Covariates'

Model:

$$\pi_{ir} = P(Y = r|\mathbf{x_i}) = \frac{\mathbf{exp}(\beta_{\mathbf{r0}} + \mathbf{x_i}^{\mathbf{T}} \beta_{\mathbf{r}})}{\sum_{\mathbf{s=1}}^{\mathbf{k}} \mathbf{exp}(\beta_{\mathbf{s0}} + \mathbf{x_i}^{\mathbf{T}} \beta_{\mathbf{s}})} = \frac{\mathbf{exp}(\eta_{\mathbf{ir}})}{\sum_{\mathbf{s=1}}^{\mathbf{k}} \mathbf{exp}(\eta_{\mathbf{is}})}$$

Variable selection apparently non-trivial (see Tutz et al.). Therefore two-step approach.

0. data transformation and variable selection (excluding highly correlated features)
1. pairwise logistic lasso regression
2. multinomial logistic lasso regression on subset of variables (e.g. union of features selected by lassos)

also see https://glmnet.stanford.edu/articles/glmnet.html#multinomial-regression-family-multinomial-

---

Data

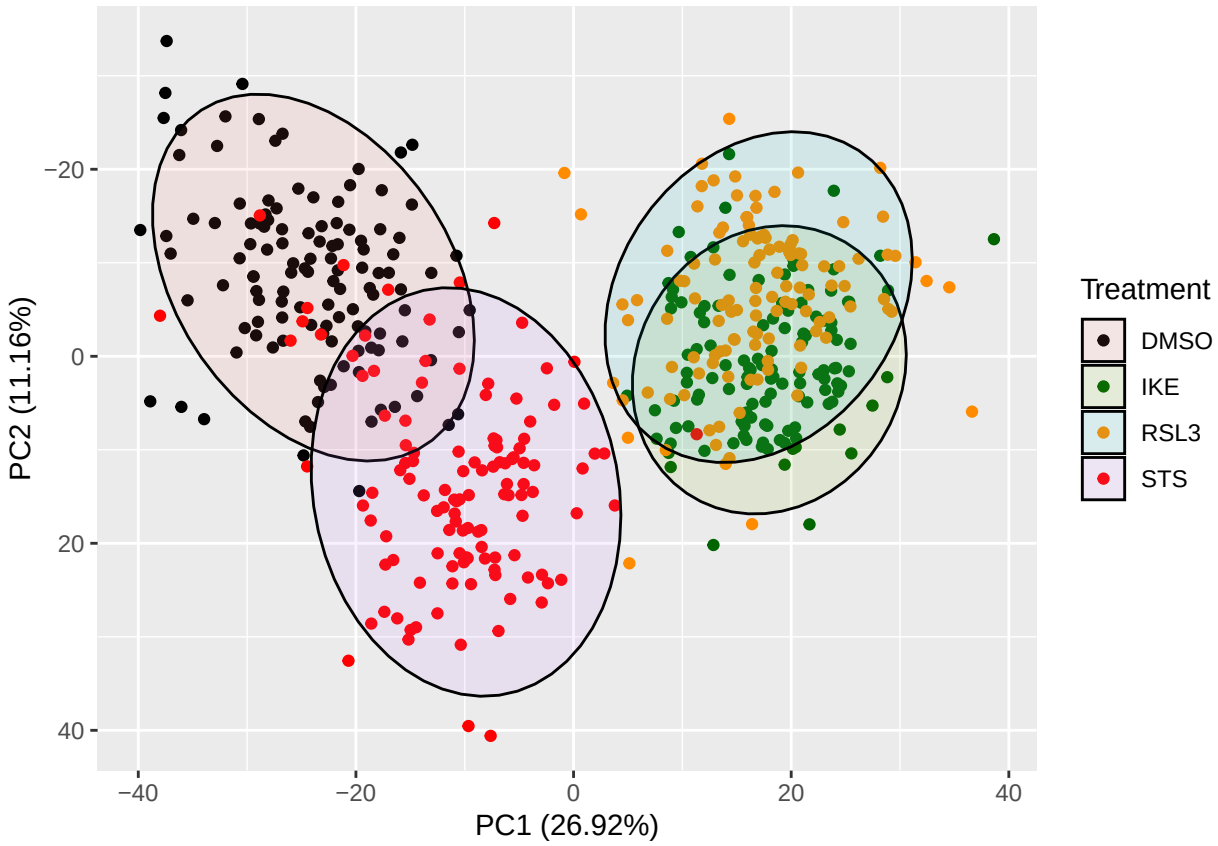
Import

```
if (file.exists("ML_ferroptosis_data_complete.Rdata")){
  load("ML_ferroptosis_data_complete.Rdata")
} else {
  col.mms <- function(data){
    X.median <- data[,grep("Median.per.Well",colnames(data))]
    colnames(X.median) <- substr(colnames(X.median),start = 28,stop = 999)
    X <- X.median
    return(X)
  }
  # +---------------+
  # | Collect data |
  # +---------------+
```

```
  Exp.list <- c("2021-03-04","2021-04-01")
  treat.list <- c("DMSO","IKE","RSL3","STS")
  X.complete <- NULL
  for (cur.exp in Exp.list){
    for (cur.tre in treat.list){
      if (file.exists(paste0(cur.exp,
           " ",cur.tre,
           " data_red_channel_wo_border_objects_ALL_single_cells.txt"))){
        # import red, blue, green channel data
        d1 <- read.table(file = paste0(cur.exp,
           " ",cur.tre,
           " data_red_channel_wo_border_objects_ALL_single_cells.txt"),
           sep ='\t',header = T)
        d2 <- read.table(file = paste0(cur.exp,
           " ",cur.tre,
           " data_blue-green_channel_wo_border_objects_ALL_single_cells.txt"),
           sep ='\t',header = T)
        # merge features
        d <- merge(x = d1,y = d2,by = "WellName")
        # organize imported data and attach to data.frame
        X.part <- cbind(data.frame("Exp" = rep(cur.exp,nrow(d)),
                                   "Treatment" = cur.tre),
                        col.mms(d))
        X.complete <- rbind(X.complete,X.part)
      }
    }
    rm(X.part,d1,d2,d)
  }
  rm(cur.exp,cur.tre,Exp.list,treat.list,col.mms)
  save(list = c("X.complete"),file = "ML_ferroptosis_data_complete.Rdata")
}
X.complete <- X.complete[,colMeans(is.na(X.complete[X.complete$Exp=="2021-03-04",]))==0]
```

PCA

```
library(ggplot2)
library(ggfortify)
X <- X.complete[X.complete$Exp=="2021-03-04",]
D <- X[,3:ncol(X)]
PCA <- prcomp(D,scale. = T)
X <- cbind(X[,1:2],PCA$x[,1:3])
lab_ap <- autoplot(PCA,plot = F)
ggplot(X, aes(PC1, PC2, col=Treatment, fill= Treatment)) +
  geom_point() + scale_y_continuous(trans = "reverse") +
  stat_ellipse(geom = "polygon", col = "black", alpha = 0.1)+
  scale_color_manual(values=c("black", "darkgreen", "darkorange","red")) +
  theme_grey() + xlab(lab_ap$labels$x) + ylab(lab_ap$labels$y)
```




```
rm(D,X,PCA,lab_ap)
```

Variance Stabilization

```
FM <- as.matrix(X.complete[,3:ncol(X.complete)])

vorher.norm <- 0
N.feature <- ncol(FM)
is.log.feature <- rep(FALSE,N.feature)
for (i in 1:N.feature){
  p.ref <- shapiro.test(FM[X.complete$Exp=="2021-03-04",i])$p.value
  p.tra <- shapiro.test(log10(1+FM[X.complete$Exp=="2021-03-04",i]))$p.value
  if ((p.ref < 0.05) & (p.ref < p.tra)){
    is.log.feature[i] <- TRUE
    FM[,i] <- log10(1+FM[,i])
  } else {
    vorher.norm <- vorher.norm + 1
  }
}
print(vorher.norm)
```

```
## [1] 259
```

```
sum(is.log.feature)
```

```
## [1] 1114
```

```
rm(is.log.feature,N.feature)
rm(p.ref,p.tra,vorher.norm,i)

X.complete[,3:ncol(X.complete)] <- FM
rm(FM)
```

Dimensionality Reduction

Check for Correlations

Goal: identify clusters of highly correlated features.

- for each feature: count number of other features, that have absolute (Pearson) correlation of more than 0.9.
- starting from the feature with highest number of correlated partners, remove all partners from the data set.
- iterate until no pair of features has correlation higher than 0.9

```
list_cors <- function(X){
  N.f <- ncol(X)
  sp <- data.frame("i" = rep(NA,(N.f-1)*N.f/2),
                   "j" = NA,
                   "r" = NA,
                   "i.name" = NA,
                   "j.name" = NA)
  k <- 1
  for (i in 2:(N.f)){
    x <- X[,i]
    x.name <- colnames(X)[i]
    for (j in 1:(i-1)){
      sp$i[k] <- i
      sp$j[k] <- j
      sp$r[k] <- cor(x,X[,j])
      sp$i.name[k] <- x.name
      sp$j.name[k] <- colnames(X)[j]
      k <- k+1
    }
    cat(i)
    cat(".")
  }
  sp$i <- as.numeric(sp$i)
  sp$j <- as.numeric(sp$j)
  sp$r <- as.numeric(sp$r)
  return(sp)
}

X.disc <- X.complete[X.complete$Exp=="2021-03-04",]
if (!file.exists("ML_ferroptosis_temp_skipCorrCalc_1.Rdata")){
  sp.cors <- list_cors(X = X.disc[,3:ncol(X.disc)])
  save(list = c("sp.cors","X.complete"),
       file = "ML_ferroptosis_temp_skipCorrCalc_1.Rdata")
} else {
  load("ML_ferroptosis_temp_skipCorrCalc_1.Rdata")
```

```
}

max(sp.cors$r)
```

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

```
sum(abs(sp.cors$r) > 0.8)
```

```
## [1] 24291
```

```
sum(abs(sp.cors$r) > 0.9)
```

```
## [1] 9079
```

```
sum(abs(sp.cors$r) > 0.95)
```

```
## [1] 2741
```

```
rm(list_cors)
```

Exclude Redundancies

Keep only one representative feature from clusters of highly correlated features. Here, iteratively the feature with the largest number of correlations (> 0.9) to other features is marked as representative and all correlated features are removed from analysis.

```
remove_cors <- function(X,sp,s = 0.9){
  features <- data.frame("Id" = colnames(X),"N.cor" = NA)
  for (i in 1:nrow(features)){
    c.f <- features$Id[i]
    features$N.cor[i] <- sum(abs(sp$r[(sp$i.name == c.f)|(sp$j.name == c.f)]) > s)
  }
  if (max(features$N.cor) >= 1){
    bleibt.index <- which.max(features$N.cor)
    bleibt.name <- features$Id[bleibt.index]
    X.bleibt <- sp[(((sp$i.name == (bleibt.name)) |
                      (sp$j.name == (bleibt.name))) &
                     (abs(sp$r) > s)),]
    raus.namen <- unique(c(X.bleibt[,4],X.bleibt[,5]))
    raus.namen <- raus.namen[raus.namen!=bleibt.name]
    X <- X[,!(colnames(X) %in% raus.namen)]
    sp <- sp[(!(sp$i.name %in% raus.namen) & !(sp$j.name %in% raus.namen)),]
  }
  for (i in 1:nrow(X.bleibt)){
    if (X.bleibt$i.name[i] != bleibt.name){
      X.bleibt[i,1:2] <- X.bleibt[i,2:1]
      X.bleibt[i,4:5] <- X.bleibt[i,5:4]
    }
  }
  result <- list("X" = X,"sp" = sp,
                 "bleibt.i" = bleibt.index,"bleibt.n" = bleibt.name,
                 "raus.namen" = raus.namen,"represents" = X.bleibt)
  return(result)
}

if (!file.exists("ML_ferroptosis_temp_skipCorrCalc_2.Rdata")){
```

```
  aus <- remove_cors(X = X.disc[,3:ncol(X.disc)],sp = sp.cors,s = 0.9)
  represents <- aus$represents
  X.disc <- cbind(X.disc[,1:2],aus$X)
  sp.cors <- aus$sp
  while (max(abs(sp.cors$r)) > 0.9){
    aus <- remove_cors(X = X.disc[,3:ncol(X.disc)],sp = sp.cors,s = 0.9)
    represents <- rbind(represents,aus$represents)
    X.disc <- cbind(X.disc[,1:2],aus$X)
    sp.cors <- aus$sp
    print(c(sum(abs(sp.cors$r)>0.9),max(abs(sp.cors$r))))
  }
  colnames(represents) <- c("kept.i","repr.j","r","kept.name","repr.name")
  rm(aus,sp.cors,remove_cors)
  save(list = ls(),file = "ML_ferroptosis_temp_skipCorrCalc_2.Rdata")
} else {
  rm(remove_cors,sp.cors)
  load(file = "ML_ferroptosis_temp_skipCorrCalc_2.Rdata")
}
```

Binary Prediction

As a proof of principle, we check, if binary prediction is possible.

RSL3 vs DMSO

```
X.full <- X.complete[,colnames(X.disc)]
X.sub <- subset.data.frame(x = X.full,subset = Treatment %in% c("DMSO","RSL3"))
train <- subset.data.frame(x = X.sub,subset = Exp == "2021-03-04")[,-1]
test <- subset.data.frame(x = X.sub,subset = Exp == "2021-04-01")[,-1]
test.full <- subset.data.frame(x = X.full,subset = Exp != "2021-03-04")[,-1]

y.out <- ifelse(test = train[,1]=="RSL3",yes = 1,no = 0)
x.in <- as.matrix(train[,-1])
mod <- cv.glmnet(y = y.out,x = x.in,family = "binomial",alpha = 1)

out <- predict(object = mod,
               newx = as.matrix(test[,-1]),
               s = mod$lambda.1se,
               type = "class")
ttt <- table(out,test[,1])
ttt
```

```
##
## out DMSO RSL3
##   0  107    0
##   1   13  119
```

```
# accuracy:
sum(diag(ttt))/sum(ttt)
```

```
## [1] 0.9456067
```

works perfect! - 95% accuracy in independent validation data.

Prediction of untrained classes:

```
out <- predict(object = mod,
               newx = as.matrix(test.full[,-1]),
               s = mod$lambda.1se,
               type = "class")
table(out,test.full[,1])
```

```
##
## out DMSO IKE RSL3 STS
##   0  107   0    0  30
##   1   13 120  119  90
```

- IKE classified as RSL3.
- STS non-uniform, but rather non-DMSO

RSL3 and STS

train RSL3 vs STS - is IKE classified as RSL3?

```
X.sub <- subset.data.frame(x = X.full,subset = Treatment %in% c("RSL3","STS"))
train <- subset.data.frame(x = X.sub,subset = Exp == "2021-03-04")[,-1]
test <- subset.data.frame(x = X.full,subset = Exp == "2021-04-01")[,-1]

y.out <- ifelse(test = train[,1]=="STS",yes = 1,no = 0)
x.in <- as.matrix(train[,-1])
mod <- cv.glmnet(y = y.out,x = x.in,family = "binomial",alpha = 1)

out <- predict(object = mod,
               newx = as.matrix(test[,-1]),
               s = mod$lambda.1se,
               type = "class")
ttt <- table(out,test[,1])
ttt
```

```
##
## out DMSO IKE RSL3 STS
##   0    0   0 113  108   6
##   1  120   7   11 114
```

```
rm(mod,out,train,test.full,test,x.in,X.sub,ttt,y.out)
```

Multiclass Prediction

- binary logistic lasso models pairwise
- take union of all selected variables
- train multinomial logistic regression lasso model

4-class Signature

Discovery and Validation

```
#load(file = "ML_ferroptosis_Xfull.Rdata")
# +-----------------------------------------+
  cur.seed <- 1119
  set.seed(cur.seed)
#  save(list = "cur.seed",file = "CurSeed.Rdata")
# +-----------------------------------------+
classes <- unique(X.full$Treatment)
pairs <- data.frame("A" = classes[c(1,1,1,2,2,3)],
                    "B" = classes[c(2,3,4,3,4,4)])

selected <- NULL
for (j in 1:nrow(pairs)){
  X.sub <- subset.data.frame(x = X.full,subset = Treatment %in% pairs[j,])
  train <- subset.data.frame(x = X.sub,subset = Exp == "2021-03-04")[,-1]
  test <- subset.data.frame(x = X.sub,subset = Exp == "2021-04-01")[,-1]
  y.out <- ifelse(test = train[,1]==pairs[j,2],yes = 1,no = 0)
  x.in <- as.matrix(train[,-1])
  mod <- cv.glmnet(y = y.out,x = x.in,family = "binomial",alpha = 1)
  selected <- unique(c(selected,
                as.character(rownames(coef(mod)))[as.numeric(coef(mod))!=0]))
}
selected <- selected[-1]

X.fs <- X.full[,c("Exp","Treatment",selected)]
N.fs <- ncol(X.fs)

train <- subset.data.frame(x = X.fs,subset = Exp == "2021-03-04")[,-1]
test <- subset.data.frame(x = X.fs,subset = Exp == "2021-04-01")[,-1]
y.out <- train[,1]
x.in <- as.matrix(train[,-1])
mod <- cv.glmnet(y = y.out,
                 x = x.in,
                 family = "multinomial",
                 type.multinomial = "ungrouped",
                 alpha = 1,
                 type.measure = "class")
out <- predict(object = mod,
               newx = as.matrix(test[,-1]),
               s = mod$lambda.1se,
               type = "class")

table(out,test[,1])
```

```
##
## out    DMSO IKE RSL3 STS
##   DMSO  101   0    0   0
##   IKE    18  89   89   8
##   RSL3    0  31   29   0
##   STS     1   0    1 112
```

```
mean(out == test[,1])
```

```
## [1] 0.691023
```

```
ttt <- table(out,test[,1])
ttt3 <- cbind(ttt[,1],ttt[,2]+ttt[,3],ttt[,4])
ttt3 <- rbind(ttt3[1,],ttt3[2,]+ttt3[3,],ttt3[4,])
ttt3
```

```
##      [,1] [,2] [,3]
## [1,]  101    0    0
## [2,]   18  238    8
## [3,]    1    1  112
```

```
sum(diag(ttt3))/sum(ttt3)
```

```
## [1] 0.9415449
```

```
rm(mod,out,pairs,test,train,ttt3,x.in,y.out,ttt,selected,X.fs,N.fs,classes,j,X.sub)
```

3-class Signature (Ferroptosis, Apoptosis)

```
X.full$Treatment[X.full$Treatment %in% c("IKE", "RSL3")] <- "ferroptosis"
X.full$Treatment[X.full$Treatment %in% c("STS")] <- "apoptosis"
X.full$Treatment[X.full$Treatment %in% c("DMSO")] <- "healthy"
classes <- unique(X.full$Treatment)
pairs <- data.frame("A" = classes[c(1,1,2)], "B" = classes[c(2,3,3)])

selected <- NULL
for (j in 1:nrow(pairs)){
  X.sub <- subset.data.frame(x = X.full,subset = Treatment %in% pairs[j,])
  train <- subset.data.frame(x = X.sub,subset = Exp == "2021-03-04")[,-1]
  test <- subset.data.frame(x = X.sub,subset = Exp == "2021-04-01")[,-1]
  y.out <- ifelse(test = train[,1]==pairs[j,2],yes = 1,no = 0)
  x.in <- as.matrix(train[,-1])
  mod <- cv.glmnet(y = y.out,x = x.in,family = "binomial",alpha = 1)
  selected <- unique(c(selected,
                as.character(rownames(coef(mod)))[as.numeric(coef(mod))!=0]))
}
pairwise_features <- selected[-1]

X.fs <- X.full[,c("Exp","Treatment",selected[-1])]
N.fs <- ncol(X.fs)

train <- subset.data.frame(x = X.fs,subset = Exp == "2021-03-04")[,-1]
test <- subset.data.frame(x = X.fs,subset = Exp == "2021-04-01")[,-1]
y.out <- train[,1]
x.in <- as.matrix(train[,-1])
mod <- cv.glmnet(y = y.out,
                 x = x.in,
                 family = "multinomial",
                 type.multinomial = "ungrouped",
                 alpha = 1,
                 type.measure = "class")
```

```
out <- predict(object = mod,
               newx = as.matrix(test[,-1]),
               s = mod$lambda.1se,
               type = "class")

table(out,test[,1])
```

```
##
## out           apoptosis ferroptosis healthy
##   apoptosis         103           0       0
##   ferroptosis        17         239      15
##   healthy             0           0     105
```

```
mean(out == test[,1])
```

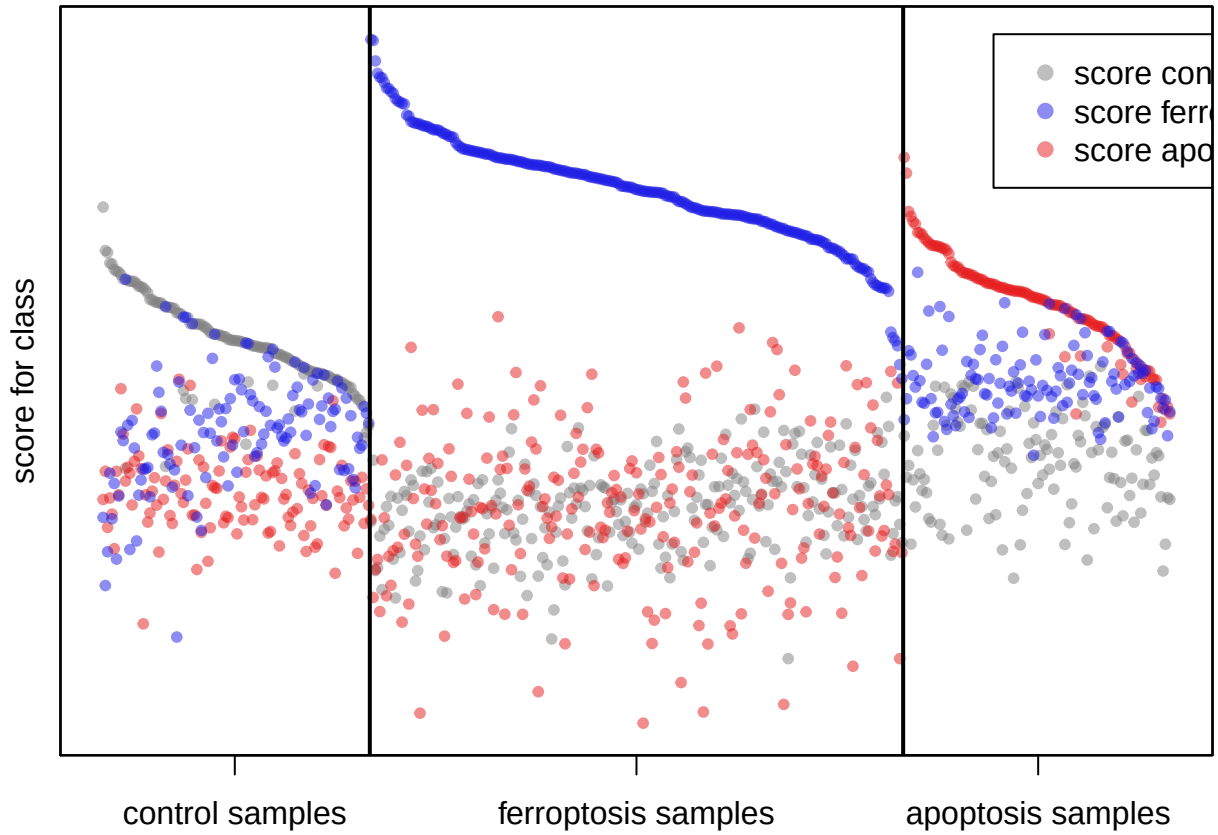
```
## [1] 0.9331942
```

```
rm(out,pairs,train,x.in,X.sub,classes,j,N.fs,selected,y.out)
```

figure scores

```
out.score <- predict(object = mod,
                     newx = as.matrix(test[,-1]),
                     s = mod$lambda.1se)

part1 <- test[,1] == "healthy"
part2 <- test[,1] == "ferroptosis"
part3 <- test[,1] == "apoptosis"
sicher <- apply(X = out.score,MARGIN = 1,FUN = max)
sicher1 <- sicher[part1]
sicher2 <- sicher[part2]
sicher3 <- sicher[part3]
reihenfolge1 <- order(sicher1,decreasing = T)
reihenfolge2 <- order(sicher2,decreasing = T)
reihenfolge3 <- order(sicher3,decreasing = T)
rm(sicher,sicher1,sicher2,sicher3,part1,part2,part3)

reihenfolge <- c(reihenfolge1,
                 reihenfolge2+length(reihenfolge1),
                 reihenfolge3+length(reihenfolge1)+length(reihenfolge2))
rm(reihenfolge1,reihenfolge2,reihenfolge3)
#pdf(file = "Fig_Validation_Scores.pdf",width = 10,height = 3.3)
par(mar = c(2.5,2,0.5,0.5))
plot(out.score[reihenfolge,3,1],pch = 20,
     col = rgb(red = 0.5,green = 0.5,blue = 0.5,alpha = 0.5,maxColorValue = 1),
     ylim = c(-3.3,4.1),xaxt = 'n',yaxt = 'n',ann = F)
par(new = T)
plot(out.score[reihenfolge,1,1],pch = 20,
     col = rgb(red = 0.9,green = 0.1,blue = 0.1,alpha = 0.5,maxColorValue = 1),
     ylim = c(-3.3,4.1),xaxt = 'n',yaxt = 'n',ann = F)
par(new = T)
plot(out.score[reihenfolge,2,1],pch = 20,
     col = rgb(red = 0.1,green = 0.1,blue = 0.9,alpha = 0.5,maxColorValue = 1),
```

```
     ylim = c(-3.3,4.1),xaxt = 'n',yaxt = 'n',
     ylab = "score for class",font = 12,
     mgp = c(0.5,2,1),xlab = "")
legend(x = 400,
       y = 4.1,
       legend = c("score control","score ferroptosis","score apoptosis"),
       fill = F,
       col = c(rgb(red = 0.5,green = 0.5,blue = 0.5,alpha = 0.5,
                   maxColorValue = 1),
               rgb(red = 0.1,green = 0.1,blue = 0.9,alpha = 0.5,
                   maxColorValue = 1),
               rgb(red = 0.9,green = 0.1,blue = 0.1,alpha = 0.5,
                   maxColorValue = 1)),
       pch = 19,density = F,lty = 0,border = F,text.font = 12)
axis(side = 1,at = c(60,240,420),font = 12,
     labels = c("control samples","ferroptosis samples","apoptosis samples"))
abline(v = 120.5,lwd = 2)
abline(v = 359.5,lwd = 2)
```




```
rm(out.score,reihenfolge)
#dev.off()
```

Summary Variables

Coefficients used (separately for classes) - summed up in table:

```
mod.full <- mod
coef.mod <- coef(mod.full)
mod.1 <- coef.mod$healthy
P.h <- cbind(as.character(rownames(mod.1))[as.numeric(mod.1)!=0],
      round(as.numeric(mod.1)[as.numeric(mod.1)!=0],digits = 3))
mod.1 <- coef.mod$ferroptosis
P.f <- cbind(as.character(rownames(mod.1))[as.numeric(mod.1)!=0],
      round(as.numeric(mod.1)[as.numeric(mod.1)!=0],digits = 3))
mod.1 <- coef.mod$apoptosis
P.a <- cbind(as.character(rownames(mod.1))[as.numeric(mod.1)!=0],
      round(as.numeric(mod.1)[as.numeric(mod.1)!=0],digits = 3))
selected <- data.frame("Var" = unique(c(P.h[,1],P.f[,1],P.a[,1])),
                       "h" = 0, "f" = 0, "a" = 0)
for (i in 1:nrow(P.h)){
  selected$h[which(selected$Var == P.h[i,1])] <- as.numeric(P.h[i,2])
}
for (i in 1:nrow(P.f)){
  selected$f[which(selected$Var == P.f[i,1])] <- as.numeric(P.f[i,2])
}
for (i in 1:nrow(P.a)){
  selected$a[which(selected$Var == P.a[i,1])] <- as.numeric(P.a[i,2])
}
Ferroptosis.Signature <- selected
rm(coef.mod,mod.1,mod.full,P.a,P.f,P.h,i)
```

table S2

Table S2 from the supporting information is exported from the object `represents`.

```
saf <- represents[represents$kept.name %in% Ferroptosis.Signature$Var,]
saf$kept.name <- gsub(x = saf$kept.name,pattern = "..Median.per.Well.x",replacement = "red")
saf$repr.name <- gsub(x = saf$repr.name,pattern = "..Median.per.Well.x",replacement = "red")
saf$kept.name <- gsub(x = saf$kept.name,pattern = "..Median.per.Well.y",replacement = "bluegreen")
saf$repr.name <- gsub(x = saf$repr.name,pattern = "..Median.per.Well.y",replacement = "bluegreen")
saf$kept.name <- gsub(x = saf$kept.name,pattern = "..Median.per.Well",replacement = "")
saf$repr.name <- gsub(x = saf$repr.name,pattern = "..Median.per.Well",replacement = "")
for (feat in unique(saf$kept.name)){
  block <- subset.data.frame(saf,subset = kept.name == feat)
  saf[which(saf$kept.name == feat),] <- block[order(block$r,decreasing = T),]
}
rownames(saf) <- 2:(1+nrow(saf))
colnames(saf) <- c("i","j","correlation coefficient","feature in signature","correlated feature (|r| > 0
saf$`correlation coefficient` <- round(saf$`correlation coefficient`,digits = 3)
head(saf[,3:5])
```

```
##   correlation coefficient                          feature in signature
## 2                   0.997 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
## 3                   0.997 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
## 4                   0.996 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
## 5                   0.996 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
## 6                   0.994 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
## 7                   0.994 Cytoplasm.Region.Radial.Mean.SER.Edge.bluegreen
##                     correlated feature (|r| > 0.9)
```

```
## 2 Cytoplasm.Region.Radial.Mean.SER.Valley.bluegreen
## 3   Cytoplasm.Region.Radial.Mean.SER.Dark.bluegreen
## 4 Cytoplasm.Region.Radial.Mean.SER.Saddle.bluegreen
## 5   Cytoplasm.Region.Radial.Mean.SER.Hole.bluegreen
## 6           Cytoplasm.Region.Radial.Mean..bluegreen
## 7  Cytoplasm.Region.Radial.Mean.SP.Filter.bluegreen
```

Variable Importance

```
selected$sd <- NA
train <- subset.data.frame(x = X.fs,subset = Exp == "2021-03-04")[,-1]
for (i in 2:nrow(selected)){
  selected$sd[i] <- round(sd(train[,selected$Var[i]]),digits = 3)
}
selected$importance <- round(apply(X = abs(selected[,2:5]),
                                   MARGIN = 1,
                                   FUN = max) * selected$sd,digits = 3)

Ferroptosis.Signature.sortiert <- selected[order(selected$importance,decreasing = T),]
Ferroptosis.Signature.sortiert
```

```
##                                                                      Var
## 21                   Nucleus.Region.Blue.SER.Saddle.2.px...Median.per.Well
## 18               Membrane.Region.Profile.1.5.SP.Filter...Median.per.Well.y
## 19                 Intensity.Nucleus.Region.Red.Contrast...Median.per.Well
## 7                            Cytoplasm.Region.Roundness...Median.per.Well
## 3                           Cytoplasm.Region.Width..px....Median.per.Well
## 9                     Cytoplasm.Region.Red.SER.Ridge.0.px...Median.per.Well
## 12               Cytoplasm.Region.Radial.Mean.SER.Edge...Median.per.Well.y
## 5                   Membrane.Region.Green.SER.Ridge.2.px...Median.per.Well
## 4                    Membrane.Region.Green.SER.Hole.2.px...Median.per.Well
## 2                  Intensity.Membrane.Region.Red.Contrast...Median.per.Well
## 13                    Nucleus.Region.Red.SER.Bright.1.px...Median.per.Well
## 14                    Nucleus.Region.Red.SER.Valley.2.px...Median.per.Well
## 10 Membrane.Region.Threshold.Compactness.30..SER.Bright...Median.per.Well.x
## 20                     Nucleus.Region.Blue.SER.Edge.2.px...Median.per.Well
## 6          Cytoplasm.Region.Radial.Mean.Ratio.SER.Spot...Median.per.Well.x
## 11   Nucleus.Region.Threshold.Compactness.50..SER.Edge...Median.per.Well.y
## 24                     Nucleus.Region.Blue.SER.Dark.2.px...Median.per.Well
## 15                Nucleus.Region.Profile.5.5.SP.Filter...Median.per.Well.x
## 8                    Membrane.Region.Red.SER.Valley.0.px...Median.per.Well
## 22  Membrane.Region.Threshold.Compactness.40..SER.Dark...Median.per.Well.y
## 17                       Cytoplasm.Region.Profile.4.5....Median.per.Well.x
## 23                        Membrane.Region.Profile.1.5....Median.per.Well.x
## 16         Nucleus.Region.Radial.Mean.Ratio.SER.Bright...Median.per.Well.x
## 1                                                                (Intercept)
##          h       f        a    sd importance
## 21   0.000   0.000 1306.847 0.001      1.307
## 18   0.000 -25.089    0.000 0.022      0.552
## 19   0.000   0.000    3.994 0.099      0.395
## 7   22.679   0.000    0.000 0.016      0.363
## 3    2.331   0.000    0.000 0.152      0.354
## 9    0.000 -33.494    0.000 0.010      0.335
```

```
## 12   0.000  -2.309    0.000 0.102      0.236
## 5  103.084   0.000    0.000 0.002      0.206
## 4  197.007   0.000    0.000 0.001      0.197
## 2   -1.413   0.000    0.000 0.137      0.194
## 13   0.000 -25.307    0.000 0.007      0.177
## 14   0.000  46.000    0.000 0.003      0.138
## 10   0.000   5.208    0.000 0.025      0.130
## 20   0.000   0.000   27.524 0.003      0.083
## 6    4.707   0.000    0.000 0.016      0.075
## 11   0.000   3.162    0.000 0.023      0.073
## 24   0.000   0.000   71.024 0.001      0.071
## 15   0.000  -2.158    0.000 0.029      0.063
## 8    0.000  -0.665    0.000 0.085      0.057
## 22   0.000   0.000   -1.932 0.018      0.035
## 17   0.000  -0.664    0.000 0.041      0.027
## 23   0.000   0.000   -2.453 0.004      0.010
## 16   0.000  -0.002    0.000 0.010      0.000
## 1   -7.590  10.716   -3.126    NA         NA
```

```
rm(test,train,X.fs,i)
```

heatmaps

```
library(circlize)
library(ComplexHeatmap)
library(glmnet)
```

discovery data

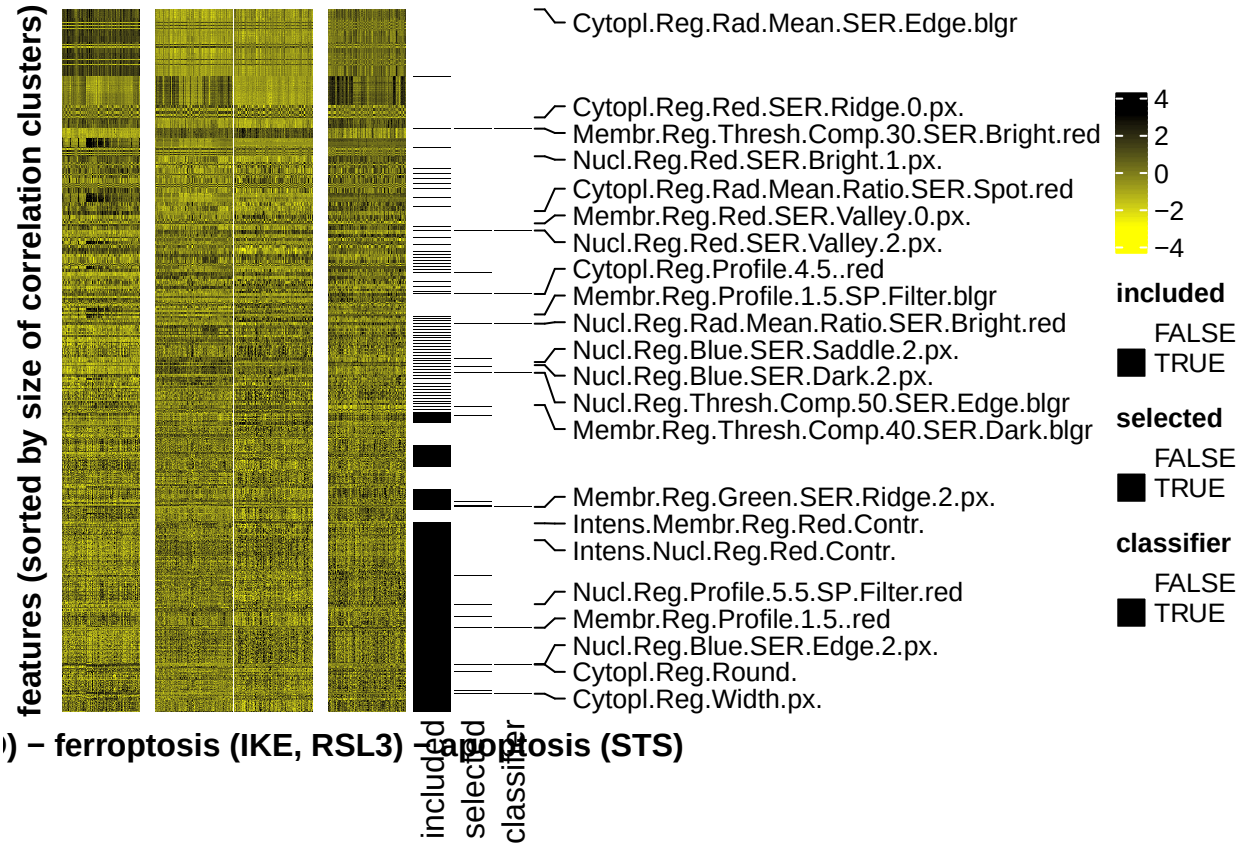
```
# Full scaled discovery-matrix
# ----------------------------
train <- subset.data.frame(x = X.complete,subset = Exp == "2021-03-04")[,-c(1,2)]

X.uncor <- X.full
Add.Info <- data.frame("ID" = colnames(train),
                       "is.cor.excluded" = TRUE,
                       "is.pariwise.lasso.selected" = FALSE,
                       "is.signature" = FALSE)
Add.Info$is.cor.excluded[Add.Info$ID %in% colnames(X.uncor)] <- FALSE
Add.Info$is.pariwise.lasso.selected[Add.Info$ID %in% pairwise_features] <- TRUE
Add.Info$is.signature[Add.Info$ID %in% Ferroptosis.Signature$Var] <- TRUE
summary(Add.Info)
```

```
##       ID            is.cor.excluded is.pariwise.lasso.selected is.signature
##  Length:1373        Mode :logical   Mode :logical              Mode :logical
##  Class :character   FALSE:595       FALSE:1328                 FALSE:1350
##  Mode  :character   TRUE :778       TRUE :45                   TRUE :23
```

```
# ---------
# load("ML_Ferroptosis_temp_skipCorrCalc_2.Rdata")
# order features by appearance in represents[,1:2]
reprs <- unique(represents$kept.name)
```

```
N.cluster <- length(reprs)
features <- NULL
for (i in 1:N.cluster){
  features <- c(features,
                reprs[i],
                represents$repr.name[represents$kept.name==reprs[i]])
}

X.scale <- scale(train)
features <- c(features,colnames(train)[!(colnames(train) %in% features)])

X_mat <- X.scale[,features] # 480 x 1373
X_mat <- t(X_mat)

rownames(Add.Info) <- Add.Info$ID
Add.Info <- Add.Info[features,]

varvektor <- features[features %in% Ferroptosis.Signature$Var[2:24]]
varvektor <- gsub(pattern = "Median.per.Well",replacement = "",x = varvektor)
varvektor <- gsub(pattern = "Membrane",replacement = "Membr",x = varvektor)
varvektor <- gsub(pattern = "Cytoplasm",replacement = "Cytopl",x = varvektor)
varvektor <- gsub(pattern = "Nucleus",replacement = "Nucl",x = varvektor)
varvektor <- gsub(pattern = "Region",replacement = "Reg",x = varvektor)
varvektor <- gsub(pattern = "Intensity",replacement = "Intens",x = varvektor)
varvektor <- gsub(pattern = "Threshold",replacement = "Thresh",x = varvektor)
varvektor <- gsub(pattern = "Compactness",replacement = "Comp",x = varvektor)
varvektor <- gsub(pattern = "Roundness",replacement = "Round",x = varvektor)
varvektor <- gsub(pattern = "Contrast",replacement = "Contr",x = varvektor)
varvektor <- gsub(pattern = "\\.\\.",replacement = ".",x = varvektor)
varvektor <- gsub(pattern = "\\.\\.",replacement = ".",x = varvektor)
varvektor <- gsub(pattern = "Radial",replacement = "Rad",x = varvektor)
varvektor <- gsub(pattern = "\\.x",replacement = ".red",x = varvektor)
varvektor <- gsub(pattern = "\\.y",replacement = ".blgr",x = varvektor)

anno_df <- data.frame(included = !Add.Info$is.cor.excluded,
                      selected = Add.Info$is.pariwise.lasso.selected,
                      classifier = Add.Info$is.signature)
anno_mat <- as.matrix(anno_df)

col_fun <- colorRamp2(c(-3, 3), c("yellow", "black"))

N.sig <- nrow(Ferroptosis.Signature)-1
dort <- rep(NA,N.sig)
for (i in 1:N.sig){
  cur.feat <- Ferroptosis.Signature$Var[i+1]
  dort[i-1] <- which(features==cur.feat)
}

ha <- HeatmapAnnotation(df = anno_df,which = "row",
                col = list(included = c("TRUE" = "black","FALSE" = "white"),
                           selected = c("TRUE" = "black","FALSE" = "white"),
                           classifier = c("TRUE" = "black","FALSE" = "white")),
                foo = anno_mark(at = sort(dort),
```

```
                          labels = varvektor,
                          labels_gp = gpar(fontsize = 10)))
#pdf(file = "Fig_discovery_yb.pdf",width = 10,height = 7)
Heatmap(matrix = X_mat,name = " ",col = col_fun,
    row_order = 1:1373,
    column_order = 1:480,
    column_split = rep(c("DMSO","IKE","RSL3","STS"),each = 120),
    column_gap = unit(c(2,0.2,2), "mm"),
    row_title = "features (sorted by size of correlation clusters)",
    row_title_gp = gpar(fontsize = 12, fontface = "bold"),
    column_title = "control (DMSO) - ferroptosis (IKE, RSL3) - apoptosis (STS)",
    column_title_side = "bottom",
    column_title_gp = gpar(fontsize = 12, fontface = "bold"),
    show_row_names = FALSE,
    show_column_names = FALSE,
    right_annotation = ha)
```

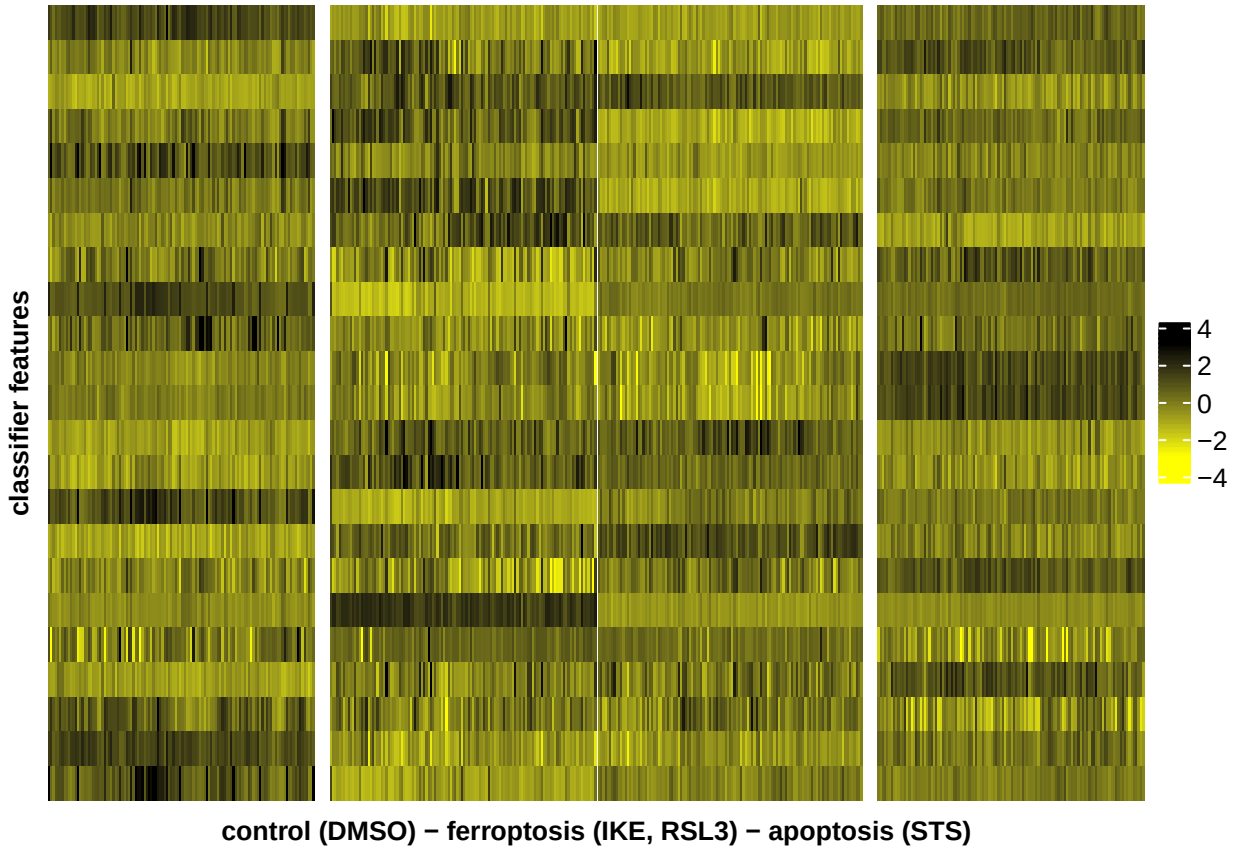



```
rm(ha,X_mat,dort)
#dev.off()
```

validation data

```
valData <- subset.data.frame(x = X.complete,subset = Exp == "2021-04-01")[,-c(1,2)]
VD.scale <- scale(valData)
SignaturSortiertFigure <- c(Ferroptosis.Signature$Var[1],
```

```
                              features[features %in% Ferroptosis.Signature$Var])
VD_mat <- VD.scale[,SignaturSortiertFigure[2:24]] # 480 x 1373
VD_mat <- t(VD_mat)
col_gr <- colorRamp2(c(-3, 3), c("yellow", "black"))

#pdf(file = "Fig_valdata_yb.pdf",width = 10,height = 1.75)
Heatmap(matrix = VD_mat,name = " ",col = col_gr,
    row_order = 1:23,
    column_order = 1:479,
    column_split = rep(c("DMSO","IKE","RSL3","STS"),c(120,120,119,120)),
    column_gap = unit(c(2,0.2,2), "mm"),
    row_title = "classifier features",
    row_title_gp = gpar(fontsize = 10, fontface = "bold"),
    column_title = "control (DMSO) - ferroptosis (IKE, RSL3) - apoptosis (STS)",
    column_title_side = "bottom",
    column_title_gp = gpar(fontsize = 10, fontface = "bold"),
    show_row_names = FALSE,
    show_column_names = FALSE)
```




```
#dev.off()
```