PM 579 Final Project

Yue Tu, Dong Yuan, Zeyun Lu

08/05/2020

Introduction

Prediction on the disease outcome of newly enrolled patients based on the historical gene expression database has always been a clinically demanding request from both the physicians and patients. Machine learning, as the emerging powerful classification techniques in statistical fields with better prediction accuracy, becomes a popular tool for supervised classification. In order to compare the performance of different machine learning methods on genetic dataset, we would like to conduct supervised prediction analysis with Random Forest, Elastic Net Regularization and Support Vector Machine using gene expression dataset.

The dataset used for finding the best classification method is from a Kaggle project **Gene expression dataset (Golub et al.): Molecular Classification of Cancer by Gene Expression Monitoring**. It originally comes from a proof-of-concept study published in 1999 by Golub et al. The purpose of this Kaggle project is to use gene expression levels by NDA microarray to predict which kind of leukemia patients would develop. Leukemia is a type of cancer that attacks cells in the bone marrow that make blood. Acute leukemia is a type of the disease that grows quickly. There are two types of acute leukemia: Acute Lymphoblastic Leukemia (ALL), which accounts 80% of children leukemia, and Acute Myeloid Leukemia (AML), which accounts the rest 20%. Therefore, this practice would provide researchers a general approach for assigning tumors to known classes.

All the analyses are performed using R. The dataset is downloaded from *https://www.kaggle.com/crawford/gene-expression*. When we introduce different methods, we use some of the contents from the Internet and books, the reference sources are listed at last. The analyses codes and software version information are attached as appendix for reproducibility purpose.

Method

For the sake of computing complexity and running time, we pick 500 most variable genes based on Median Absolute Deviation (MAD) and standardize gene expression for ALL the analyses.

Before we apply ENET, RF, SVM on our dataset, data visualization is performed, and the technique we use is Multidimensional Scaling (MDS). MDS is a technique that attempts to preserve pairwise distances between each observation. MDS takes a dissimilarity matrix D where D_{ij} represents the dissimilarity between points i and j and produces a mapping on a lower dimension, preserving the dissimilarities as closely as possible. The dissimilarity matrix could be observed or calculated from the given dataset.

The whole dataset will be splitted into training with 50 patients and validation dataset with 22 patients. Pseudo-random seed is set for the purpose of reproducibility. All the prediction methods will be applied on the training dataset and the models' performance will be evaluated on the validation dataset.

Next, we begin the model comparison by performing ENET to predict the cancer type. Since the outcome is a binary variable for two types of cancer, penalized logistic regression model is constructed. λ is a regularization parameter and the regularization path is computed for the elastic net penalty at a grid of values for λ. The parameter λ controls the overall strength of the penalty. α is the proportion between the square of coefficients and the absolute value of the coefficients, which is another parameter that can vary in the model. Ten-fold cross-validation will be performed on the standardized training dataset with α ranging from 0 to 1 with 0.1 increment, using misclassification error as the criterion for model performance. The λ we use from each model is the one showing the error is within one standard error of the minimum mean cross-validated error. The α with the lowest corresponding mean misclassification error on the training dataset is selected.

Then, we build the prediction model with a tree-based method, which is Random Forest. RF is an improved method over bagged trees by de-correlating the trees. That is, we build a number of decision trees on bootstrapped training samples. However, when building these decision trees, instead of considering all p number of predictors in a splitting stage, we only allow a random sample of size m predictors being the split candidates (typically, with size equal to $\sqrt{p}$). This method is extremely helpful when we have a large number of correlated predictors as in our case. A bunch of random forest models with m equal to $\sqrt{p}$, but with different numbers of trees, are performed on a training dataset. And the model with lowest Out-of-Bag error rate (OOB) will be selected as the final model. The importance of each predictor is quantified based on the Mean Decrease Accuracy (MDA) in predictions on the out of bag samples when a given variable is excluded from the model, and on the Mean Decrease Gini (MDG), i.e. deviance, that results from splits over that variable, averaged over all trees.

Finally, the Support Vector Machine is applied. SVM specializes in finding the boundary which is formed by a set of support vectors. We firstly use SVM with a linear kernel, which is assuming that the data is linearly separable. In the linear separation model, the tuning parameter α, which specifies the cost of a violation to the margin, is selected via ten-fold cross validation. When α is large, then the margins will be narrow and there will be a few support vectors on the margin or violating the margin. Next, we apply SVM with a non-linear kernel such as a polynomial kernel, and repeat the process.

A table that contains classification accuracy on the validation data set of each method will be reported.

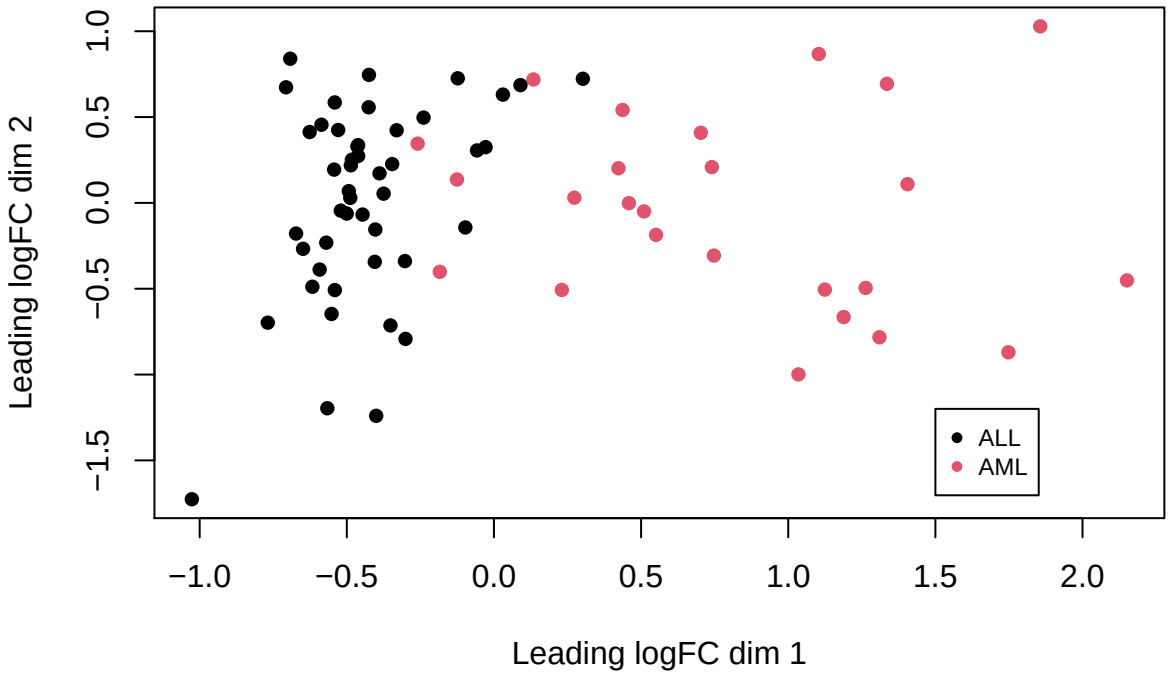
For ENET, we list the top 10 genes with the largest absolute value of coefficients. For RF, we list top 10 genes with highest MDA and highest MDG, respectively. Because of the nature of SVM, we cannot list the top 10 genes that drive the prediction. In addition, for comparison, we perform two-sample t tests between AAL and AML patients on gene expression. Then, we adjust their P-value according to Bonferroni correction and list top 10 significant genes.

Results

The expression data and annotation data are identically sorted. The dataset comes from 72 patients. 47 of them are ALL patients and 25 of them are AML patients. There are 7131 genes. There is no missing value. Furthermore, we check that the 500 most variable genes explain 48% of total variation while the 1000 most variable genes explain 57% of total variation with only 9% increase. As a result, we think these 500 most variable genes are sufficient.

From Figure 1, we can see that ALL patients and AML patients are clearly separated with only a few exceptions. AML patients' points are more sparse than ALL patients'. This plot actually provides us with confidence that gene expression levels between ALL and AML patients are somewhat different and we are able to use sophisticated learning methods to correctly classify them.

Figure 1: MDS plot



For ENET, after generating the models with varying α values, we find that the mean misclassification error on the training dataset is 0.02 when $\alpha = 0$ and 0.08 when $\alpha = 1$. But all other α give 0 error rate, so the performance of $\alpha = 1$ and $\alpha = 0$, which are also called Ridge and LASSO regression, is inferior than other models (Table 1). Also, since the error rate is 0, there is no standard error from the minimum mean cross-validated error. The λ we pick is also the one gives minimum error rate. Since all other models have the same performance under our evaluation criteria, we will pick $\alpha = 0.1$ arbitrarily.

Each curve (Figure 2) visualizes the corresponding coefficients to each gene and coefficients for those non-significance genes will be shrunk to zero. Figure 3 is for the cross-validation error curves for the chosen model as the change of $log(\lambda)$. The dotted line located at the λ we selected.

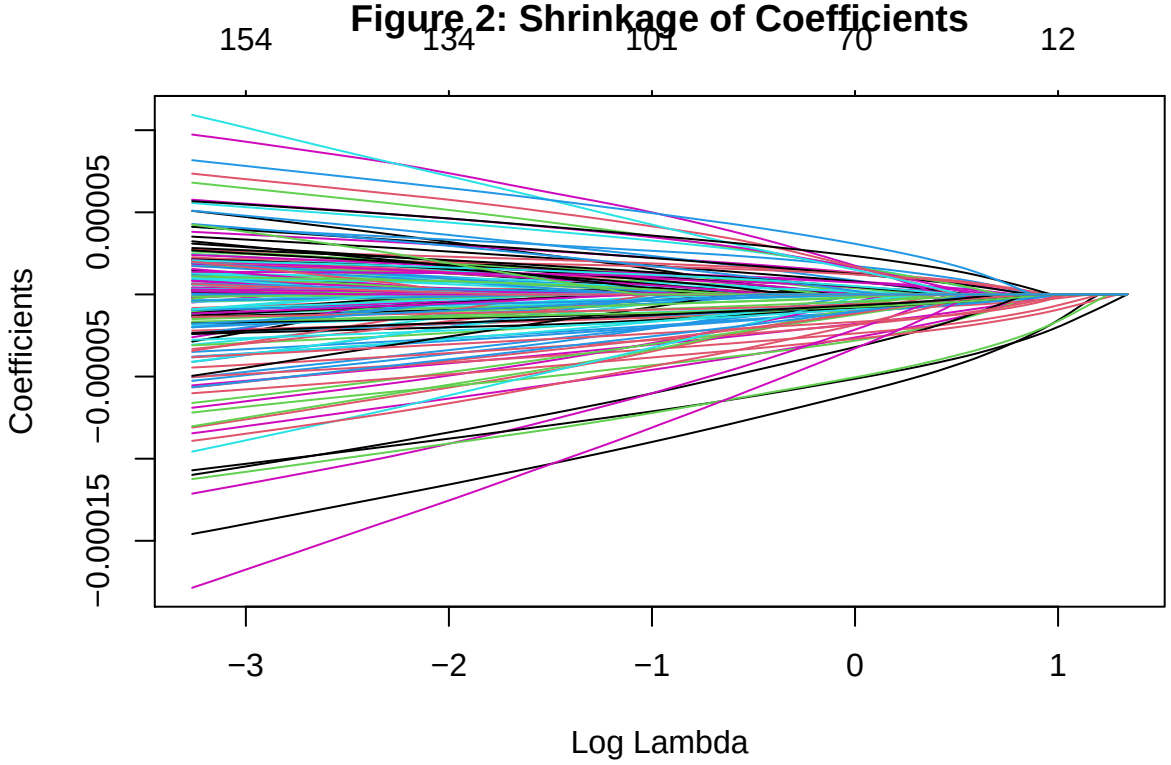
When checking the model performance on a validation dataset, the ENET model with $\alpha = 0.1$ correctly classifies 6 patients with ALL and 13 patients with 13. Out of the 20 patients, 3 are misclassified and the prediction accuracy is 86.36% (Table 2).

Among those 500 genes, 92 genes have non-zero coefficients and the gene 2111, 4847, 6041, 6422, 3183, 6855, 4763, 1615, 1249, and 4196 are the top 10 highest coefficients. Those genes are theoretically the most important gene in predicting the type of cancer using this prediction model (Table 3).

Table 1: Error for Each Alpha

Error	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.08
Alpha	0.00	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.00

Figure 2: Shrinkage of Coefficients



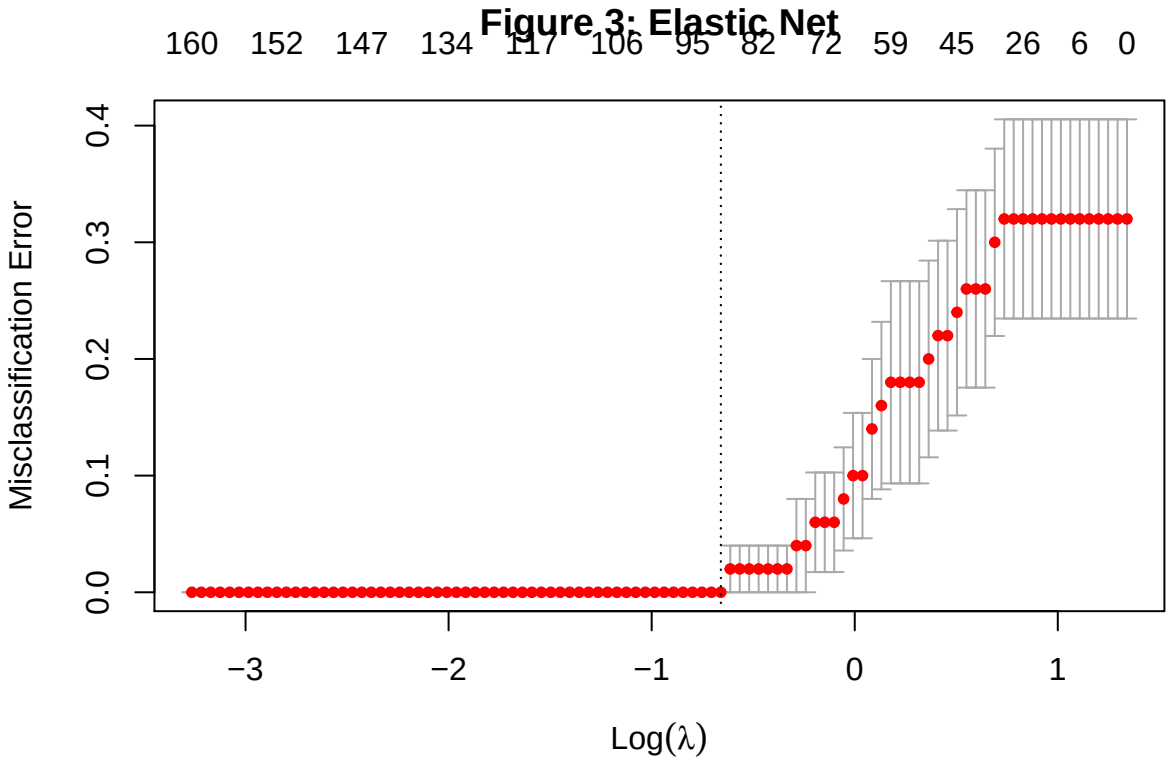


Figure 3: Elastic Net

In the random forest model, we plot the error rate versus number of trees grown (Figure 4). The plot has shown that the Out-of-Bag error rate (OOB) is relatively low when the number of trees are between 100 and 130 but does not improve significantly after 100. We proceed to build a random forest with 130 trees as the final model, which results in the lowest OOB error rate as 2%. By applying it on the validation dataset, the prediction result shows that only one subject with outcome AML being misclassified, and the total prediction accuracy is 95.45% (Table 2).

Furthermore, according to the value of Mean Decrease Accuracy, we find that Gene 6855, 6041, 1685, 1882, 4847, 2121, 5501, 2354, 1674, and 1400 are 10 most influential genes. According to the value of Mean Decrease Gini, we find that Gene 6855, 1685, 6041, 5501, 4847, 1882, 2121, 1400, 1674, and 4196 are the 10 most influential genes (Table 3).

Figure 4: RF with Increasing Number of Trees

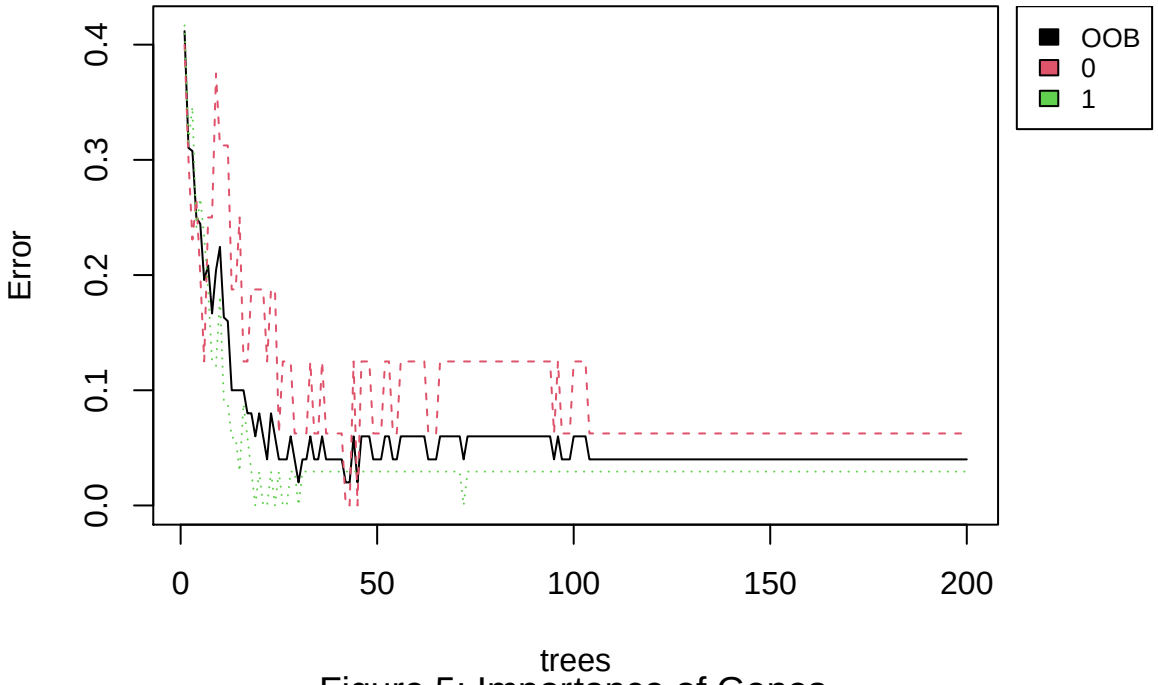
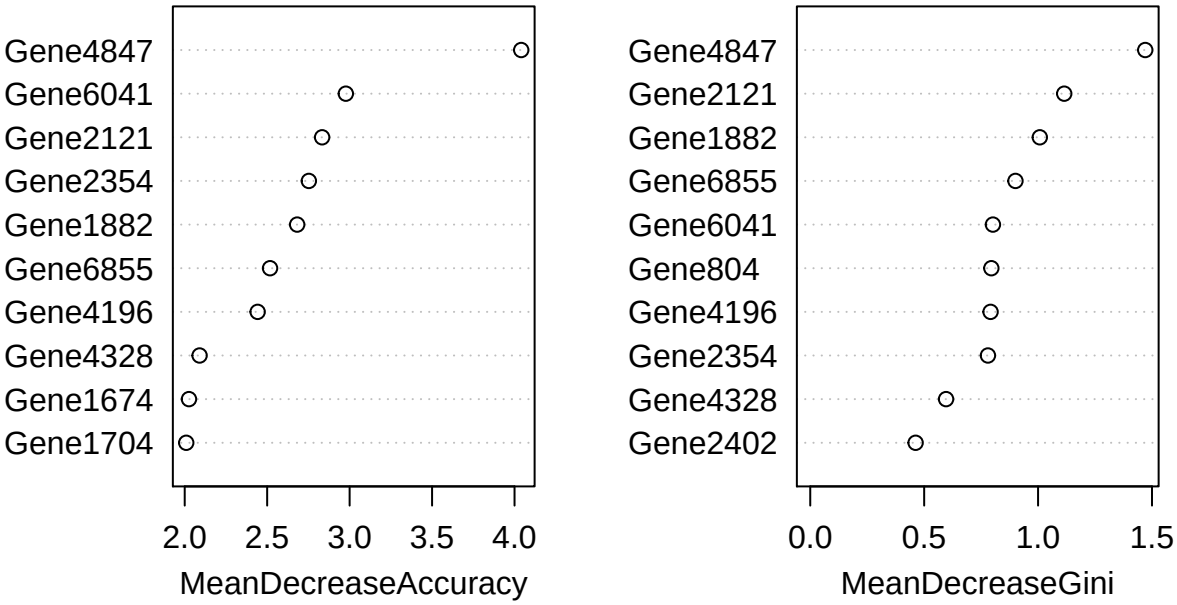


Figure 5: Importance of Genes



Finally, in the Support Vector Machine with linear kernel, the model with $\alpha = 0.01$ gives lowest cross validation error and is therefore selected as the final model. The number of support vectors in AML patients is 11, and in ALL patients is 15. The prediction on validation dataset gives two misclassifications on AML

patients, with prediction accuracy as 90.91%. However, the SVM with polynomial kernel and degree equal to 3 result in a 20% error rate on training dataset and 23% error rate on validation dataset, indicating the linear boundary is well established for this dataset (Table 2).

Table 2: Overall Performance

Classified	True	ENET	RF	SVM (Linear)	SVM (Polynomial)
AML	AML	6	8	7	4
ALL	AML	3	1	2	5
AML	ALL	0	0	0	0
ALL	ALL	13	13	13	13
Error Rate		13.63%	4.54%	9.09%	22.72%

RF method performs best with 4.54% classification error rate (CRR); SVM with linear kernel perform the second best, which gives us 9.09% CRR, CRR of ENET is 13.63% while SVM with polynomial kernel performs the worst with 22.72% CRR.

We have 60 genes out of 500 that show significance between AAL and AML patients. We also list 10 most significant genes (Table 3).

For the top-10-gene comparison across methods, Gene 6041 and 4847 show up in all the 4 groups; Gene 4196, 6855, 2121, 1674, and 1882 are overlapping in 3 of the 4 groups while Gene 1400, 1685, 1882, 2111, and 5501 are appearing across 2 of the 4 groups (Table 3).

Table 3: Top Genes

ENET	RF MDA	RF MDG	t test
Gene2111	Gene6855	Gene6855	Gene4847
Gene6422	Gene6041	Gene1685	Gene4196
Gene4847	Gene1685	Gene6041	Gene6041
Gene6041	Gene1882	Gene5501	Gene1882
Gene3183	Gene4847	Gene4847	Gene2121
Gene4763	Gene2121	Gene1882	Gene2111
Gene6855	Gene5501	Gene2121	Gene1674
Gene3684	Gene2354	Gene1400	Gene1779

ENET	RF MDA	RF MDG	t test
Gene1249	Gene1674	Gene1674	Gene4328
Gene1615	Gene1400	Gene4196	Gene6806

Discussion

In conclusion, Random Forest has the best performance in the validation dataset while SVM with linear kernel is the second best. The reason why RF performs the best is that it accounts for the correlation among the predictors and we know that gene expression data is somewhat non-independent because of the linkage disequilibrium; meanwhile, the reason why ENET is relatively inferior is that it requires independence assumption between predictors. SVM works best when there exist clear margins of separation between ALL and AML, which is shown in the MDS plot. Because of the roughly straight line of separation in the MDS plot, SVM with a linear kernel performs better than the one with a polynomial kernel.

We see a large number of overlapping genes from top 10 genes derived from ENET, RF, and 2-sample t test with multiple testing correction, which gives us preliminary speculation that these are the potential causal genes for a patient to have AAL or AML. We haven't mapped those selected influential genes back to their original names and haven't checked for their actual biological function due to lack of expertise. However, we believe that there should be genetic reasons behind the scene. And the biological function of those most predictive genes for ALL and AML is worth further investigation.

As for limitation, we only have 72 patients, which is relatively a small sample size. The decrease in power may have prevented us from finding more significant driven genes in ALL/AML classification. If we were able to have several hundred patients, our results would be more robust and clinical sense-making.

Reference

1. What Is Leukemia (ALL and AML)? (https://www.nationwidechildrens.org/conditions/leukemia-all-and-aml)
2. Multi-Dimension Scaling (MDS) (https://blog.paperspace.com/dimension-reduction-with-multi-dimension-scaling/)
3. Wikipedia (https://en.wikipedia.org/wiki/Principal_component_analysis)

4. A One-Stop Shop for Principal Component Analysis (https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c)
5. An Introduction to Statistical Learning with Application in R, Robert Tibshirani
6. https://cran.r-project.org/web/packages/glmnet/vignettes/glmnet.pdf

Session Info

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.29          e1071_1.7-3         randomForest_4.6-14
##  [4] glmnet_4.0-2        Matrix_1.2-18       qvalue_2.20.0
##  [7] genefilter_1.70.0   forcats_0.5.0       stringr_1.4.0
## [10] dplyr_1.0.0         purrr_0.3.4         readr_1.3.1
## [13] tidyr_1.1.0         tibble_3.0.3        tidyverse_1.3.0
## [16] matrixStats_0.56.0  ggplot2_3.3.2       limma_3.44.3
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6         fs_1.4.2             lubridate_1.7.9
##  [4] bit64_0.9-7.1        httr_1.4.1           tools_4.0.2
```

```
##  [7] backports_1.1.8      R6_2.4.1             DBI_1.1.0
## [10] BiocGenerics_0.34.0  colorspace_1.4-1     withr_2.2.0
## [13] tidyselect_1.1.0     bit_1.1-15.2         compiler_4.0.2
## [16] cli_2.0.2            rvest_0.3.5          Biobase_2.48.0
## [19] xml2_1.3.2           scales_1.1.1         digest_0.6.25
## [22] rmarkdown_2.3        pkgconfig_2.0.3      htmltools_0.5.0
## [25] highr_0.8            dbplyr_1.4.4         rlang_0.4.7
## [28] readxl_1.3.1         rstudioapi_0.11      RSQLite_2.2.0
## [31] shape_1.4.4          generics_0.0.2       jsonlite_1.7.0
## [34] RCurl_1.98-1.2       magrittr_1.5         Rcpp_1.0.5
## [37] munsell_0.5.0        S4Vectors_0.26.1     fansi_0.4.1
## [40] lifecycle_0.2.0      stringi_1.4.6        yaml_2.2.1
## [43] plyr_1.8.6           grid_4.0.2           blob_1.2.1
## [46] parallel_4.0.2       crayon_1.3.4         lattice_0.20-41
## [49] haven_2.3.1          splines_4.0.2        annotate_1.66.0
## [52] hms_0.5.3            pillar_1.4.6         reshape2_1.4.4
## [55] codetools_0.2-16     stats4_4.0.2         reprex_0.3.0
## [58] XML_3.99-0.5         glue_1.4.1           evaluate_0.14
## [61] modelr_0.1.8         vctrs_0.3.1          foreach_1.5.0
## [64] cellranger_1.1.0     gtable_0.3.0         assertthat_0.2.1
## [67] xfun_0.15            xtable_1.8-4         broom_0.7.0
## [70] class_7.3-17         survival_3.1-12      iterators_1.0.12
## [73] AnnotationDbi_1.50.3 memoise_1.1.0        IRanges_2.22.2
## [76] ellipsis_0.3.1
```

Codes

```
knitr::opts_chunk$set(echo = TRUE)
library(limma)
library(ggplot2)
library(matrixStats)
library(tidyverse)
library(genefilter)
```

```
library(qvalue)
library(glmnet)
library(randomForest)
library(e1071)
library(knitr)

dd1 = read_csv("data/data_set_ALL_AML_independent.csv")
dd2 = read_csv("data/data_set_ALL_AML_train.csv")
alldd = left_join(dd1, dd2, by = c("Gene Description", "Gene Accession Number")) %>%
  select(-`Gene Accession Number`)
dropcol <- which(!grepl("call", colnames(alldd)))
tmpdd <- t(as.matrix(alldd[dropcol]))
colN <- tmpdd[1, ]
tmpdd <- tmpdd[-1, ]

tmpdd <- cbind(rownames(tmpdd), tmpdd)
tmpdd <- as.data.frame(tmpdd)
colnames(tmpdd) <- c("patient", paste0("Gene", 1:7129))
rownames(tmpdd) <- NULL

ddWIDE <- read_csv("data/datasets_1868_3249_actual.csv") %>%
  mutate(patient = as.character(patient)) %>%
  left_join(tmpdd, by = "patient") %>%
  mutate(cancer = ifelse(cancer == "ALL", 1, ifelse(cancer == "AML", 0, NA))) %>%
  mutate_at(vars(-cancer), as.numeric) %>%
  as.data.frame

ddLONG <- mutate(alldd[dropcol], `Gene Description` = paste0("Gene", 1:7129)) %>%
  rename(Var = `Gene Description`) %>%
  as.data.frame

ddLONG <- ddLONG[c(1, order(as.numeric(colnames(ddLONG[, -1]))) + 1)]
```

```r
tmp <- ddLONG[, -1]
fmad  <- matrixStats::rowMads(as.matrix(tmp))
rfmad <- rank(-fmad)
fidx <- which(rfmad <= 500)

ddWIDE500 <- ddWIDE[c(1, 2, fidx + 2)]
ddLONG500 <- ddLONG[fidx, ]
ddPatient <- read_csv("data/datasets_1868_3249_actual.csv") %>%
  mutate(patient = as.character(patient),
    cancer = factor(cancer, levels = c("ALL", "AML")))
allN <- sum(ddPatient$cancer =="ALL")
amlN <- sum(ddPatient$cancer =="AML")
mds <- as.matrix(ddWIDE500[, -c(1, 2)])
sdMDS <- scale(mds, center = TRUE, scale = matrixStats::colMads(mds))

plotMDS(t(sdMDS), pch=16, col=unclass(ddPatient$cancer),
  main = "Figure 1: MDS plot")
legend(1.5, -1.2, levels(ddPatient$cancer), pch = 16,
  col = order(levels(ddPatient$cancer)), cex = .75)
dat1 <- ddWIDE500 %>%
  mutate(y = as.factor(cancer)) %>%
  select(-c(1:2)) %>%
  data.matrix

#glmnet prefer data.matrix as for as.matrix
x <- dat1[, -ncol(dat1)]
y <- dat1[, ncol(dat1)]

set.seed(666)
train_rows <- sample(1:nrow(dat1), 50)
x.train <- x[train_rows, ]
x.test <- x[-train_rows, ]
```

```r
y.train <- y[train_rows]
y.test <- y[-train_rows]

col1 <- vector()

for (i in 0:10) {
  temp <-  cv.glmnet(x.train, y.train, type.measure = "class",
    alpha = i/10, family = "binomial")
  err.min <- temp$cvm[temp$lambda == temp$lambda.1se]
  col1[i+1] = err.min
}

col2 <- seq(0,1,0.1)
errout <- rbind(col1,col2)
rownames(errout) <- c("Error", "Alpha")
kable(errout, caption = "Error for Each Alpha")

fit.elnet <- glmnet(x.train, y.train, family = "binomial", alpha = 0.1)
plot(fit.elnet, xvar = "lambda", main = "Figure 2: Shrinkage of Coefficients")

fit1 <- cv.glmnet(x.train, y.train, type.measure = "class",
  alpha = 0.1,family = "binomial")
plot(fit1, main = "Figure 3: Elastic Net")



yhat <- predict(fit1, s=fit1$lambda.1se,type = "class", newx = x.test) %>%
  as.factor

enetT <- table(yhat, y.test)
rownames(enetT) <- c("Classified as AML", "Classified as ALL")
colnames(enetT) <- c("True AML", "True ALL")
#kable(enetT, caption = "ENET Classification")
```

```r
#coef(fit1, s="lambda.min")[which(coef(fit1, s="lambda.min") != 0)]

myCoefs <- coef(fit1, s = "lambda.min")
#myCoefs[which(myCoefs != 0 ) ]

#myCoefs@Dimnames[[1]][which(myCoefs != 0 ) ]

myResults <- data.frame(features = myCoefs@Dimnames[[1]][which(myCoefs != 0)],
  coefs = myCoefs[which(myCoefs != 0 )])
myResults$coefs <- abs(myResults$coefs)

pp <- myResults[order(myResults$coefs, decreasing = TRUE), ][2:11, 1]
set.seed(666)
train_id <- sample(nrow(ddWIDE),50)
dat500 <- mutate(ddWIDE500,cancer = as.factor(cancer))

dat_model <- dat500[,-1]
train <- dat_model[train_id,]
test <- dat_model[-train_id,]

rf <- randomForest(cancer~., data = dat_model,
  subset = train_id, ntree = 100, importance = TRUE)
rf2 <- randomForest(cancer~., data = dat_model,
  subset = train_id, ntree = 200, importance = TRUE)
rf3 <- randomForest(cancer~., data = dat_model,
  subset = train_id, ntree = 130, importance = TRUE)

layout(matrix(c(1,2), nrow = 1), width = c(4, 1))
par(mar = c(5, 4, 4, 0))
plot(rf2, main = "Figure 4: RF with Increasing Number of Trees")
par(mar = c(5, 0, 4, 2))
plot(c(0, 1), type = "n", axes = FALSE, xlab = "", ylab = "", main = "")
```

```r
legend("top", colnames(rf2$err.rate), col = 1:4, cex = 0.8, fill = 1:4)


varImpPlot(rf3, main = "Figure 5: Importance of Genes", n.var = 10)
yhat_rf3 <- predict(rf3, newdata = test)


a <- as.data.frame(importance(rf2))


a1 <- a[order(a$MeanDecreaseAccuracy, decreasing = TRUE), ]
a1 <- rownames(a1)[1:10]
a2 <- a[order(a$MeanDecreaseGini, decreasing = TRUE), ]
a2 <- rownames(a2)[1:10]


rfT <- table(yhat_rf3, test$cancer)
rownames(rfT) <- c("Classified as AML", "Classified as ALL")
colnames(rfT) <- c("True AML", "True ALL")
#kable(rfT, caption = "Random Forrest Classification")
#Linear Boundary
tune_out <- tune(svm,cancer~., data = train, kernel = "linear",
  ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100)))
#summary(tune_out)
bestmod <- tune_out$best.model
#summary(bestmod)
#predict results
yhat_svm_linear <- predict(bestmod, test)


svmlT <- table(yhat_svm_linear, test$cancer)
rownames(svmlT) <- c("Classified as AML", "Classified as ALL")
colnames(svmlT) <- c("True AML", "True ALL")
#kable(svmlT, caption = "SVM Classification with Linear Kernel")


#svm with polynomial boundary
tune_out_non2 <- tune(svm,cancer~., data = train, kernel = "polynomial",
```

```
  ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100)))
#summary(tune_out_non2)
bestmod_non2 <- tune_out_non2$best.model
#summary(bestmod_non2)
#predict results
yhat_svm_non2 = predict(bestmod_non2,test)

svmpT <- table(yhat_svm_non2, test$cancer)
rownames(svmpT) <- c("Classified as AML", "Classified as ALL")
colnames(svmpT) <- c("True AML", "True ALL")
#kable(svmpT, caption = "SVM Classification with Polynomial Kernel")
enetT <- as.data.frame(enetT)
colnames(enetT) <- c("Classified", "True", "ENET")
rfT <- as.data.frame(rfT)
colnames(rfT) <- c("Classified", "True", "RF")
svmlT <- as.data.frame(svmlT)
colnames(svmlT) <- c("Classified", "True", "SVM (Linear)")
svmpT <- as.data.frame(svmpT)
colnames(svmpT) <- c("Classified", "True", "SVM (Polynomial)")

cc <- enetT %>%
  left_join(rfT, by = c("Classified", "True")) %>%
  left_join(svmlT, by = c("Classified", "True")) %>%
  left_join(svmpT, by = c("Classified", "True")) %>%
  mutate(Classified = c("AML", "ALL", "AML", "ALL"),
    True = c("AML", "AML", "ALL", "ALL"))

totalAcu <- c("Error Rate", "", "13.63%", "4.54%", "9.09%", "22.72%")
cc[nrow(cc) + 1, ] <- totalAcu

kable(cc, caption = "Overall Performance")
```

```r
tmp <- as.matrix(ddWIDE500[, -c(1, 2)])


sdTMP <- scale(tmp, center = TRUE, scale = matrixStats::colMads(tmp))
tt <- colttests(sdTMP, ddPatient$cancer)
pval <- p.adjust(tt$p.value, method = "bonferroni")
tt$p.value <- pval


tt <- tt[order(tt$p.value), ]
tt <- rownames(tt)[1:10]
# pp <- data.frame(x = 1:length(pval), pval = -log(pval, base = 10))
# ggplot(data=pp, aes(x=x, y=pval)) +
#   geom_bar(stat="identity") +
#   geom_hline(yintercept=1.30103, color = "red") +
#   labs(x = "Gene", y = expression(-log[10]~P)) +
#   ggtitle("Pseudo Manhattan Plot")
# Top 10 genes table
comb <- as.data.frame(cbind(pp, a1, a2, tt))
colnames(comb) <- c("ENET", "RF MDA", "RF MDG", "t test")


kable(comb, caption = "Top Genes")
sessionInfo()
```