STT 843 Final Project

Multivariate Analysis and Machine Learning on Survival Gene-expression Data

Lyuxun Yang

Summary

The purpose of this project is to apply various multivariate methods on the survival gene-expression data from [1], and to build better subgrouping and predictor models than the paper. For imputation, Local least squares (LLS) surpassed other 4 models by experiment NRMSE. For dimension reduction, Principal component analysis (PCA) was selected from 16 methods by overall quality scores and visualization. Finally, Gaussian mixture model (GMM) divided patients into 3 subgroups, which has more distinctive Kaplan-Meier curves than the curves of the clustering in [1]. Kernel Survival SVM surpassed other 6 models to achieve the best cross-validation concordance score (0.7398), better than the Cox model (0.6795) in [1].

1 Introduction

The study [1] developed a molecular survival predictor using the gene-expression data from patients with diffuse large-B-cell lymphoma after chemotherapy. They divided patients into 3 subgroups by hierarchical clustering, and then constructed a predictor of risk by a Cox model. The model was significant in tests and validations.

However, in [1] and its Supplementary Appendix, some questions are worthy of discussion:

1. They "excluded patients with missing values for a particular microarray element from all analyses involving that element". Since 240 patients are not too many, why not impute the dataset to reduce wasting information? Then, which imputation method should be chosen?
2. They selected 100 genes for the clustering model. The selection is directly based on the conclusion of [2]. But is that most suitable for this data? Could we select the features based on the data set?
3. Clustering gives "hard" assignments to patients. Would "soft" probabilistic models, such as Gaussian Mixture model, more reasonable?
4. For constructing the predictor, they selected genes by Wald test, availability, and variance. Would dimension reduction methods such as PCA do a better job?
5. They used simple averages of gene-expressions to construct the 5 signatures for the Cox model, while Cox model uses an exponential linear combination of them. Might nonlinear models do a better job?

My project will show my response to these questions, and finally find better solutions in both unsupervised learning (subgrouping) and supervised learning (prediction).

2 Data Description

By preprocessing, the useful parts of the dataset can be denoted as:

- $\boldsymbol{X}$: a 240×7399 gene-expression matrix, where x_{ij} is the i-th patient's j-th gene-expression value ($\log_2(CY5/CY3)$[1]).
- $\boldsymbol{y}$: a 240×1 follow-up time vector, where y_i is the i-th patient's Follow-up time (years).

- $\boldsymbol{\delta}$: a 240×1 status vector, where δ_i is the i-th patient's Status at follow-up. $\delta_i \in \{0, 1\}$ where 0 is "alive" and 1 is "dead".

I dropped other data because they are either computed from $\boldsymbol{X}$, or from external biological standards. Since I don't have relative biological backgrounds, this project will directly focus on the data itself.

All the values in $\boldsymbol{X}$ are $\log_2(CY5/CY3)$ ratios. The data range is $(-8.20, 8.37)$, median 0.00 and mean 0.01, 1st quarter -0.39 and 3nd quarter 0.40. The density plot is **Figure 1**, and QQ-plot is **Figure 2**. In Shapiro-Wilk normality test, P-value<0.01, so the data are not normal distributed, but have high kurtosis. The density plots of columns show the similar patterns. Thus, *I will not use inferences based on strict normality.*

By the correlation plot of the first 100 features (**Figure 3**), there are many strong correlations among these columns.

3 Imputation

3.1 Imputation Theories

Generally, imputation methods includes "local algorithms" and "global algorithms"[3]. I select K-nearest neighbors (KNN) and Local least squares (LLS) from "local algorithms", and Bayesian principal component analysis (BPCA) and Singular value decomposition (SVD) from "global algorithms".

"Local algorithms" only select some most similar genes to impute a missing value. Suppose the first value in microarray $\boldsymbol{g}_i$ is missing. Now select K nearest genes $\boldsymbol{g}_{s_1}, \cdots, \boldsymbol{g}_{s_K}$ by similarity measures, such as smallest L_2-norm or Pearson correlation coefficient, then we have

$$\begin{bmatrix} \boldsymbol{g}_i^T \\ \boldsymbol{g}_{s_1}^T \\ \cdots \\ \boldsymbol{g}_{s_K}^T \end{bmatrix} = \begin{bmatrix} NA & \boldsymbol{w}^T \\ b_1 & \boldsymbol{a}_1^T \\ \cdots & \cdots \\ b_K & \boldsymbol{a}_K^T \end{bmatrix}$$

NA is the missing value. In KNN [4], $\hat{NA}$ is estimated by the mean of $b_1, \cdots, b_K$. In LLS [5], it uses the Least Squares method to fit the regression $\boldsymbol{w} = \beta_1 \boldsymbol{a}_1 + \cdots + \beta_K \boldsymbol{a}_K$, then use the fitted parameters to estimate $\hat{NA} = \hat{\beta}_1 b_1 + \cdots + \hat{\beta}_K b_K$.

"Global algorithms" are more complex. I'll only briefly introduce BPCA[8] here. We've learned the "maximum variance formulation" of PCA, but PCA also has an **equivalent** "minimum-error formulation", which finds the projection vectors to minimize the sum of squared distances between each data point and its projection. The Probabilistic PCA (PPCA)[6, 7] is then formulated as $\boldsymbol{x} = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$, where $\boldsymbol{x}$ is the D-dimensional observed variable, $\boldsymbol{z} \sim N(0, \boldsymbol{I})$ is an Gaussian latent variable, $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I})$ is noise, and $\boldsymbol{W}, \boldsymbol{\mu}, \sigma^2$ are parameters. Under this framework, BPCA defines an independent Gaussian prior over each column of $\boldsymbol{W}$, so that each $\boldsymbol{w}_i$ is a Gaussian having an independent variance governed by hyperparameter α_i, so $p(\boldsymbol{W}|\boldsymbol{\alpha}) = \prod_i \left(\frac{\alpha_i}{2\pi}\right)^{D/2} \exp\left\{-\frac{1}{2}\alpha_i \boldsymbol{w_i^T} \boldsymbol{w_i}\right\}$. Now all the distributions in BPCA model have been specified, so the Maximum Likelihood can be specified, and an Expectation-Maximization (EM) algorithm can be used to efficiently estimate the model. The E step is to use old/initial parameter values to evaluate the latent variable $\boldsymbol{z}$, and the M step is to maximize the log likelihood with repect to parameters. (I don't have space for details, but they are in [6].)

When using BPCA for imputation, a variational Bayes[8] algorithm is implemented as follows: (a) initialize the missing values by averages; (b) use the above EM algorithm to estimate the parameters in BPCA; (c) use the current parameters to update the missing values; (d) use the current parameters and imputed values to update hyperparameter $\boldsymbol{\alpha}$; (e) repeat (b)-(d) until convergence.

For comparing different methods, I will do repeated experiments for each method. In each experiment, I randomly removed 10.30% of non-missing values in $\boldsymbol{X}$ (besides the existing missing

values), and use the method to impute $\boldsymbol{X}$. Normalized root mean squared error (NRMSE) is used for quality measurement, $NRMSE = \sqrt{\frac{mean[(\boldsymbol{y_{huess}} - \boldsymbol{y_{real}})^2]}{Var[\boldsymbol{y_{real}}]}}$.

3.2 Experiment Results

There are 182919 (10.30%) missing values in $\boldsymbol{X}$. Among 7399 features, 359 (4.85%) have a missing value rate grater than 40%. These "big missing" features are dropped directly. Now $\boldsymbol{X}$ becomes 240×7040.

I did totally 23 random imputation experiments for each of the methods (KNN[k=2], KNN[k=5], KNN[k=10], LLS[k=10], PPCA, BPCA, SVD). **Figure 4** is the box plot of the NRMSE for each method. Obviously, LLS has the lowest NRMSE, so it was selected to impute $\boldsymbol{X}$.

4 Dimension Reduction

4.1 Feature Selection

Feature selection was necessary according to my experiments. In my first trial, I directly used all 7040 features to train a Gaussian Mixture Model, and the corresponding Kaplan-Meier curves (**Figure 5**) showed too little difference, so it would be useless for patients. I think the main reason is that, the model learns on some genes having distinguish values between different patients, but those genes have nothing to do with the survival risk.

Therefore, a tree-based feature selection was performed. The main idea is to train a Survival Gradient Boosting Decision Tree model by using $\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\delta}$. Then, the importance scores of all features are calculated by counting the occurrence of the features in all the decision tree nodes. Finally, the features which have effects on the tree constructions are selected.

Only 380 of 7040 features were selected in this process. Now $\boldsymbol{X}$ is 240×380.

4.2 Feature Extraction

4.2.1 Feature Extraction Theories

To find the best methods for the data, my experiments include 16 common Feature Extraction methods. Here I'll introduce PCA, Kernel PCA (kPCA) and Multidimensional Scaling (MDS).

Recall standard linear PCA. Suppose $\boldsymbol{X}$ is the $n \times p$ data matrix (n samples, p features), so $\tilde{\boldsymbol{X}} = (\boldsymbol{I} - \boldsymbol{1}\boldsymbol{1}^{\boldsymbol{T}}/N)\boldsymbol{X}$ is the centered data matrix. One way of PCA is to compute the eigen-decomposition of the covariance matrix $\tilde{\boldsymbol{\Sigma}} = \frac{1}{N}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{\boldsymbol{T}}$, $\boldsymbol{P} = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_p]$, then the Principle Components are $\boldsymbol{Z} = \tilde{\boldsymbol{X}}\boldsymbol{P}$. Another *equivalent* way of PCA[11] is to compute the eigen-decomposition of the Gram matrix $\tilde{\boldsymbol{K}} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\boldsymbol{T}}$, then the Principle Components are $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}$. (Such connection can be seen as following: if $\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}}\boldsymbol{e} = \lambda\boldsymbol{e}$, then $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}}\boldsymbol{e} = \lambda\tilde{\boldsymbol{X}}\boldsymbol{e}$, so $\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$ have same eigenvalues and related eigenvectors.)

Now, extend this to Kernel PCA (kPCA)[10]. We have a kernel function $\phi(\boldsymbol{x}_i)$ to transform $\boldsymbol{x}_i$ to a high-dimensional space. The centered kernel function is $\tilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \frac{1}{N}\sum_j \phi(\boldsymbol{x}_j)$. Denote Gram matrix $\boldsymbol{K} = \{\phi(\boldsymbol{x}_i) \bullet \phi(\boldsymbol{x}_j)\}_{ij}$, $\tilde{\boldsymbol{K}} = \{\tilde{\phi}(\boldsymbol{x}_i) \bullet \tilde{\phi}(\boldsymbol{x}_j)\}_{ij}$. It can be derived[11] that

$$\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1} + \boldsymbol{1}\boldsymbol{K}\boldsymbol{1}$$

Compute the eigen-decomposition of $\tilde{\boldsymbol{K}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\boldsymbol{T}}$, then the Kernel Principle Components are also $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}$, (the element $z_{im} = \sum_{j=1}^{N} u_{jm}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)/d_m$).

Multidimensional Scaling (MDS)[11] is close to PCA. While PCA keeps the variance information, MDS keeps the distance information. Suppose we know all $d_{ij} = dist\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ where $\boldsymbol{x}_i$ is an observation vector. Then MDS is to find low dimensional vectors $\boldsymbol{z}_i$ to minimize $\sum_{ij}(d_{ij} - dist\{\boldsymbol{z}_i, \boldsymbol{z}_j\})^2$, so it retains the distance between $\boldsymbol{x}$'s as well as possible. This optimization problem can be fitted by Gradient Descent algorithm.

Some interesting connections: When using Euclidean distance, MDS gives the same result as PCA[11]. When using Geodesic distance, MDS becomes the Isometric Mapping model (Isomap). When using the ranks of the distances, we get the nonmetric MDS (nMDS).

7 quality indexes were used to compare the similarity of the original features and the extracted features. Q_local, Q_global, and mean_R_nx are rank based criteria[12], and others are correlation based criteria[13].

4.2.2 Feature Extraction Experiment Results

In the experiments, 16 common feature extraction methods were performed on $\boldsymbol{X}$ to reduce the dimension to 240×2 (2 is only for comparison, not for use at end). The 7 quality indexes were calculated for each method. The results are in **Figure 6**. The higher quality value the better. 3 methods are obviously worse than others, but the difference among the top methods are trivial.

To compare these methods, I averaged the normalized quality values for each method, but the difference is still trivial. Then, I checked the 2D plot of the extracted data, and found that PCA gave the best pattern (See **Figure 8**, don't care the colors for now). In addition, PCA is easier for interpretation and k selection. Therefore, PCA was selected as the Feature Extraction method.

By PCA, **Figure 7** shows the Scree plot. The **Table 1** shows the number of principle components to be chosen corresponding to different proportions of variance. It shows PCA's high efficiency of extracting variance. These k selections will all be considered in the machine learning model cross-validation parts.

5 Survival Machine Learning

5.1 Theories

5.1.1 Gaussian Mixture Model

Gaussian Mixture Model(GMM)[6] suppose the observation $\boldsymbol{x}_i$ comes from the mixture of K Gaussians:

$$p(\boldsymbol{x}_i) = \sum_{k=1}^{K} \pi_k \phi(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\phi(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the density function of $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, π_k are mixing coefficients with summation 1, K is the hyperparameter. The log likelihood function is

$$\ln p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \phi(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

It is hard to be directly maximized, so the EM iterative algorithm is used: (a) initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$; (b) (E step) denote $z_i \sim Multinomial(\boldsymbol{\pi})$ as the latent variable where $z_i = k$ means the i-th sample is from the k-th Gaussian. For all i, k, calculate $w_{ik} := p(z_i = k|\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\pi}) = \frac{\pi_k \phi(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \phi(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$; (c) (M step) update $\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_i w_{ik} \boldsymbol{x}_i$, $\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_i w_{ik} (\boldsymbol{x}_i - \mu_k^{new})(\boldsymbol{x}_i - \mu_k^{new})^T$, $\pi_k^{new} = N_k/N$; (d) repeat (b) - (c) until convergence.

5.1.2 Survival SVM

Support vector machine (SVM)[11] is a model to find the hyperplanes to separate observations into categories, which are divided by a hard or soft gap that is as wide as possible. Define a hyperplane by $\{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{w} + b = 0\}$ where $\boldsymbol{w}, b$ are parameters, then the optimization problem is equivalent to $\{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} \xi_i$, subject to $y_i(\boldsymbol{x}^T \boldsymbol{w} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for any $i\}$.

Now in Survival SVM[14], we only have the follow-up time y_i and status δ_i (dead=1). What we can be sure is only that patient i died after j if $(i, j) \in \mathcal{P} = \{(i, j) \mid y_i > y_j \wedge \delta_j = 1\}_{i,j=1,\ldots,n}$.

We want to separate all these patient pairs (i, j), so the Survival SVM model is:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} \xi_{ij} \quad \text{s.t.} \quad \mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_j \geq 1 - \xi_{ij}, \quad \xi_{ij} \geq 0, \quad \forall (i,j) : y_i > y_j \wedge \delta_j = 1$$

Some modifications can be made: Fast Survival SVM[15] is ranking-based, Kernel Survival SVM[16] replaces $\boldsymbol{x}_i$ by its kernel function, etc.

5.2 Final Results

5.2.1 Subgroups of Patients

I used the 380 features from feature selection to fit the GMM model. I tried to set different numbers of components in the model, and found $k = 3$ fitted the best, which is the same k in [1]. The subgroups are shown in **Figure 8**, where different colors represent different groups. To visualize the probability for each patient in each group, I plotted the Gaussian distributions based on 2D PCA data in **Figure 9**. Then I estimated the Kaplan-Meier curves of subgroups in **Figure 11**.

In contrast, the subgroups in paper [1] is visualized in **Figure 10**, and the Kaplan-Meier curves for these subgroups are in **Figure 12**. Obviously, GMM subgroup curves (**Figure 11**) are more distinctive. That means GMM did a better job on separating patients with different survival risks.

In addition, the GMM can output the probabilities for a new patient in the subgroups (as **Figure 9**). This should be more useful for diagnosis, than the "hard" clustering.

5.2.2 Cross-validation and Final Predictor

7 models (2 linear models, 2 ensemble models, 3 SVM models) are compared in this part. For each model, I fitted it on 8 data sets, including all choices in **Table 1** plus the original dataset before PCA (380 features). For each model with each data set, a 3-fold cross-validation was performed under different combinations of hyper-parameters for the specific model, and the combination with the highest cross-validation concordance score[17] was recorded. So, totally 56 results were recorded, in **Figure 13** and **Table 2**.

In **Figure 13**, the scores of models on the Origin ($\boldsymbol{X}$ before PCA) are usually the highest. That is because it doesn't loss any information in the data. However, the models with variance proportion of 70% to 80% sometimes performs better than models with 90% to 95%. This means sometimes a lower dimension PCA can extract the useful information so that the low-dimensional data can be rather easier for models to learn.

Kernel Survival SVM with the Origin dataset has the biggest cross-validation score, 0.7398. The hyper-parameters tuned by the cross-validation are: $\alpha = 0.5$, kernel = RBF (Gaussian kernel), Rank Ratio = 1. Thus, that is my final predictor.

Table 3 shows a comparison between this model and the final model in paper [1]. It shows that the Kernel Survival SVM model performs better for survival prediction for this dataset, according to both test score and cross-validation score.

6 Further Discussion

1. In theory, the multivariate models have many connections (besides PCA-MDS-nMDS-Isomap in **Section 4.2.1**, also like PCA-ICA-FA, etc.), but for this dataset, I haven't found a way to clearly visualize these connections in experiments.
2. In the feature extraction comparing experiments, I only used the default hyperparameters. If given more time to tune the hyperparameters, better results might be achieved.
3. For dividing patient subgroups, other good unsupervised learning, like k-means and spectral clustering, could be tried. A comparison analysis could benefit.

7 Appendix

7.1 Tables

Table 1: The relationship between proportion of variance and the number of principle components

Variance Proportion	**Number of PCs**
0.3	4
0.5	13
0.7	34
0.8	54
0.9	91
0.95	124
0.99	184

Table 2: Best Cross-validation results for each model under each data set

Variance Proportion	**0.3**	**0.5**	**0.7**	**0.8**	**0.9**	**0.95**	**0.99**	**1.0**	**Origin**
KernelSurvivalSVM	0.6472	0.6178	0.6423	0.6831	0.7089	0.7086	0.7248	0.7311	0.7398
CoxPH	0.6511	0.6745	0.7192	0.7124	0.6562	0.6902	0.7236	0.7382	0.7382
SurvivalSVM	0.6509	0.6866	0.7113	0.7256	0.6187	0.6496	0.7048	0.7334	0.7292
MinlipSurvival	0.5504	0.6159	0.6692	0.7030	0.7120	0.7199	0.7307	0.7321	0.7289
GradientBoosting	0.5858	0.6139	0.6437	0.6189	0.6142	0.5824	0.5664	0.5664	0.6888
ComponentwiseGB	0.6557	0.6604	0.6818	0.6802	0.6533	0.6504	0.6557	0.6523	0.6776
IPCRidge	0.3984	0.3425	0.3221	0.3625	0.4569	0.4085	0.3478	0.3497	0.3497

Table 3: The Concordance Score comparison between KSVM and Cox model in paper [1]

Model	**Training Score**	**Test Score**	**CV Score**
KSVM	0.9975	0.7248	0.7398
Cox in [1]	0.7074	0.6632	0.6795

7.2 Figures

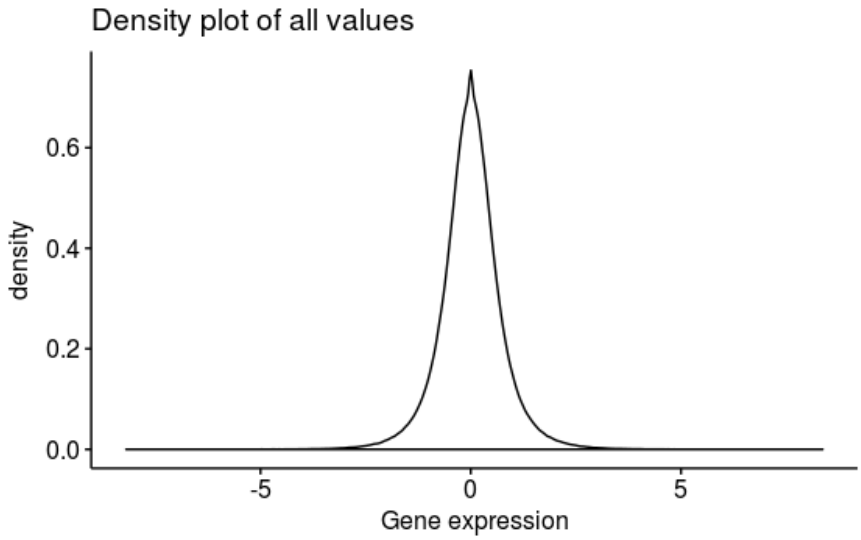

Figure 1: Density plot of all values in matrix $\boldsymbol{X}$

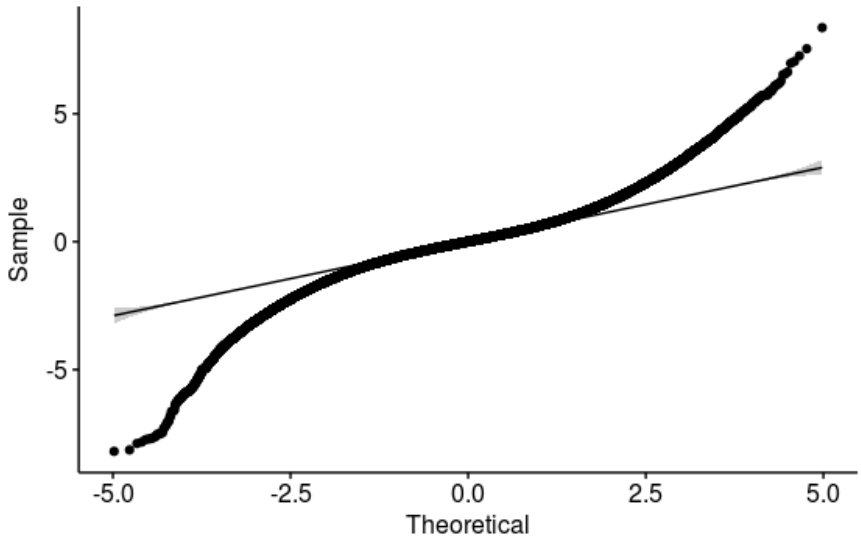

Figure 2: QQplot of all variables in matrix $\boldsymbol{X}$

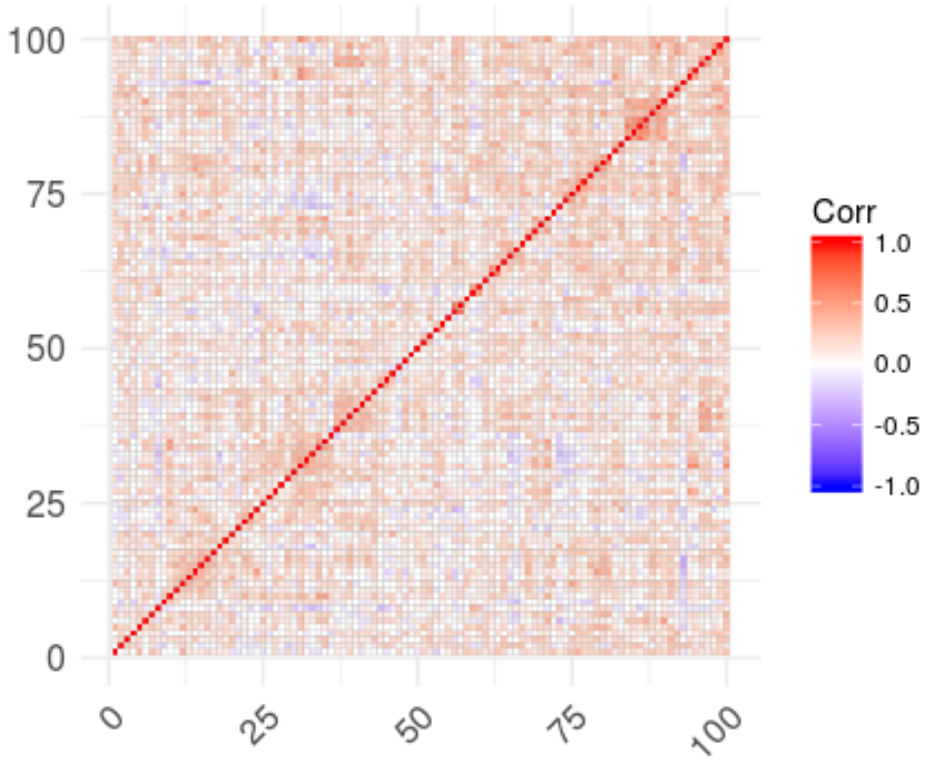

Figure 3: Correlation plot of the first 100 features

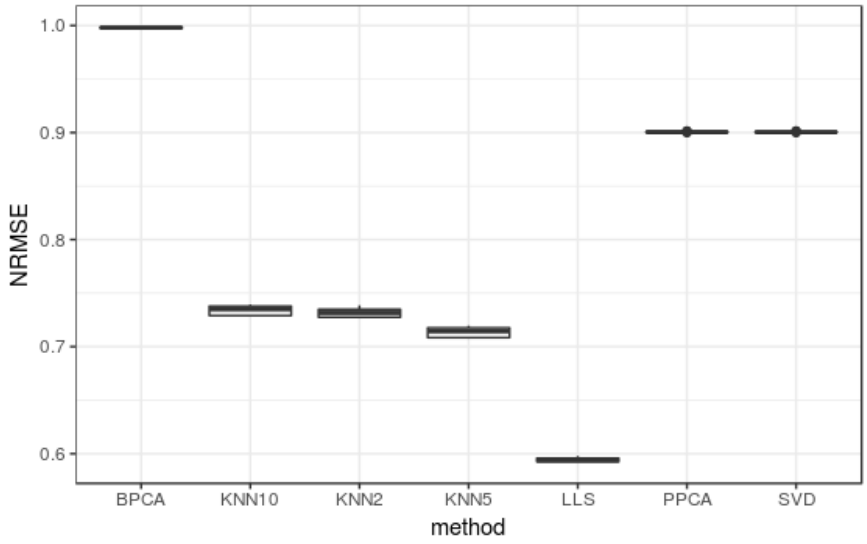

Figure 4: The box plot of NRMSE of each imputation method

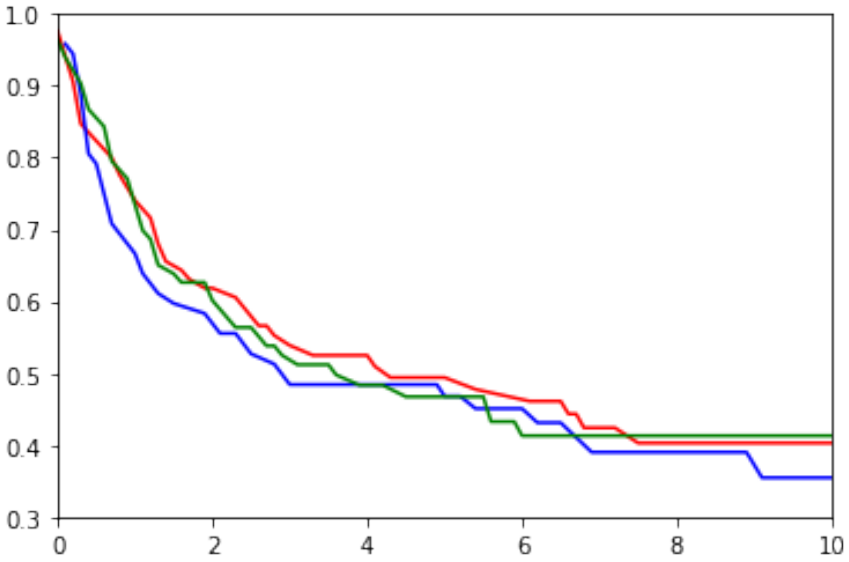

Figure 5: kaplan-meier lines of subgroups by data **without feature selection**. (to show the bad results if not select features)

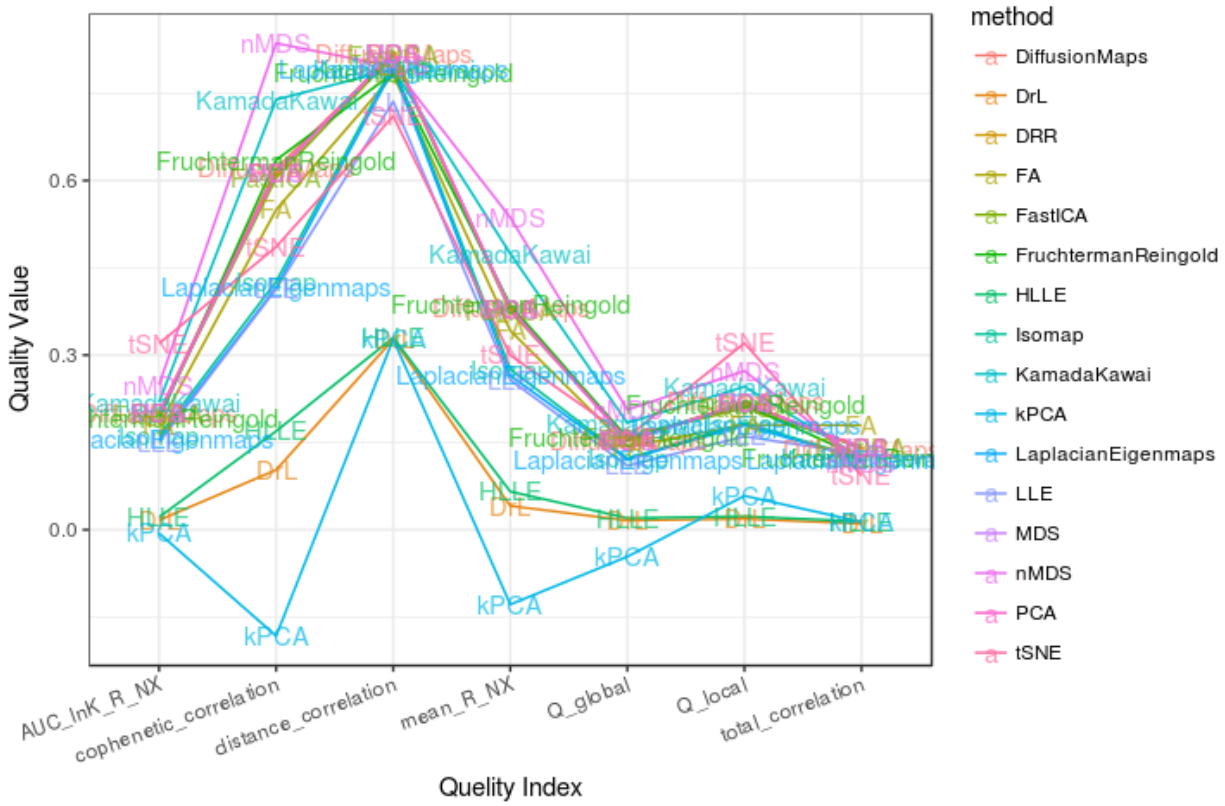

Figure 6: Qualities of different feature extraction methods

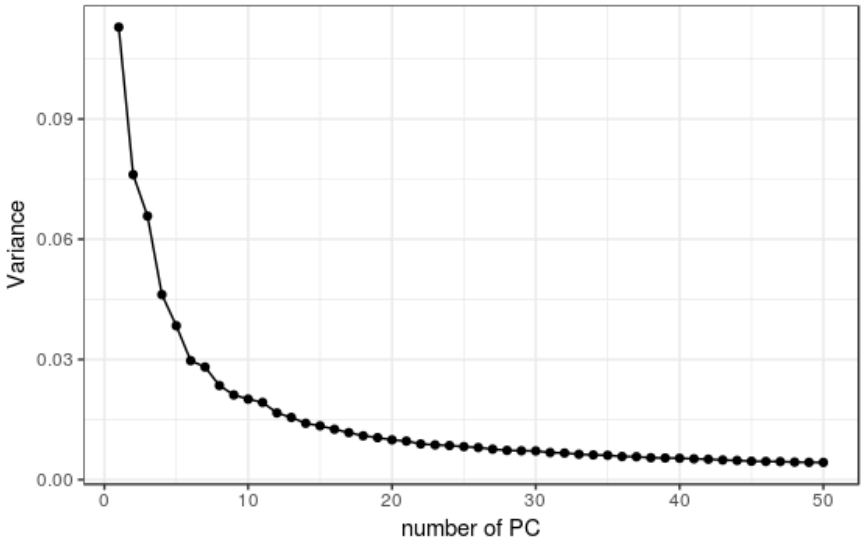

Figure 7: Scree Plot of PCA on X

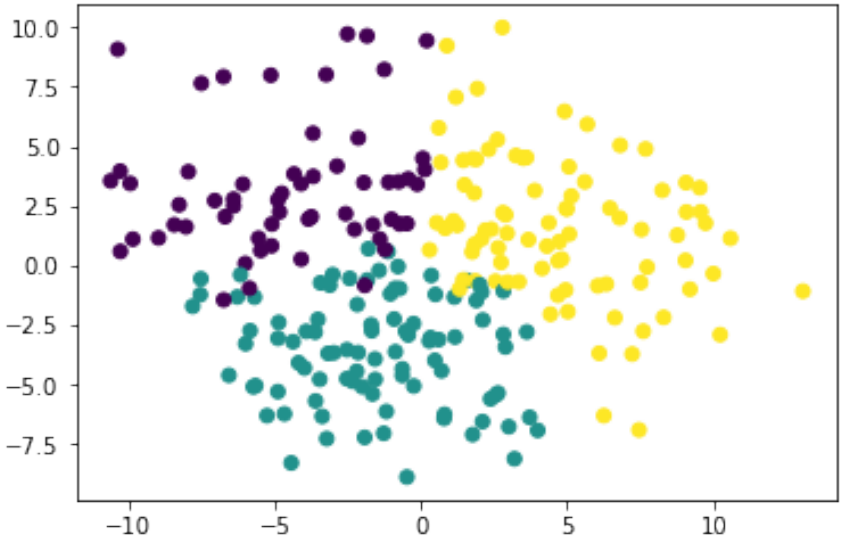

Figure 8: The subgroups by Gaussian Mixture Model on **all features**, visualized by 2D PCA. Different colors represent different groups

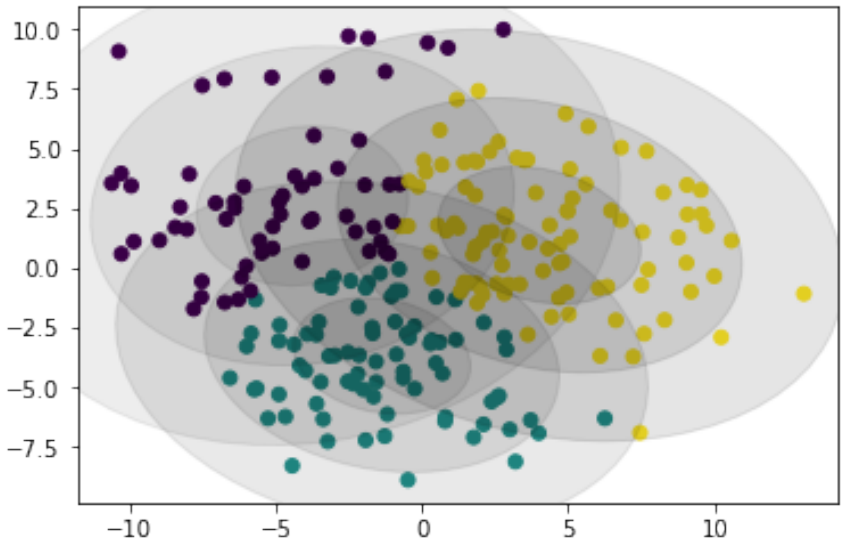

Figure 9: The subgroups by Gaussian Mixture Model on **only first 2 PCA features**. Ellipses show the 3 Gaussian distributions. Different colors represent different groups

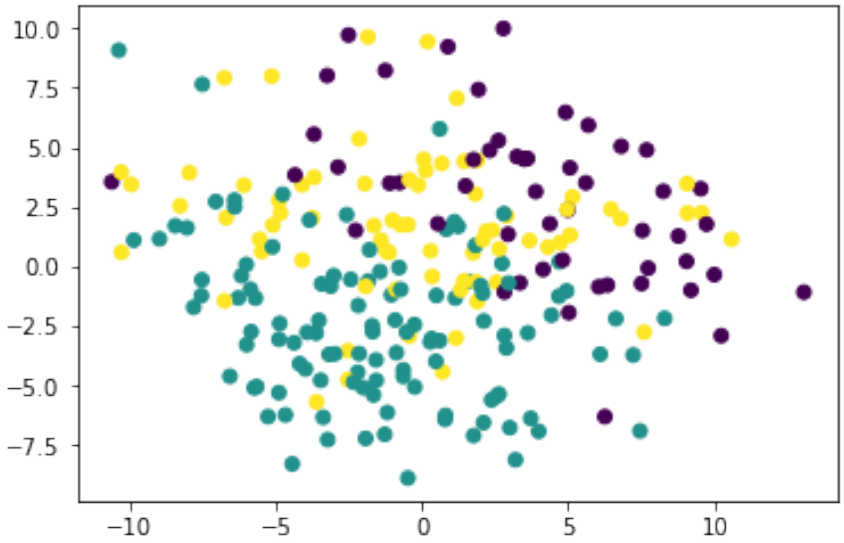

Figure 10: The subgroups from the **paper [1]**, visualized by 2D PCA. Different colors represent different groups

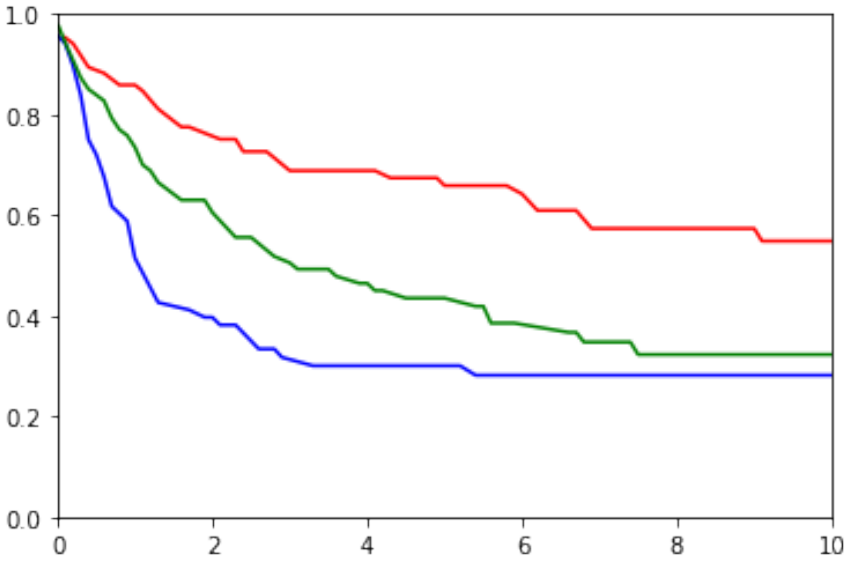

Figure 11: The Kaplan-Meier curves of 3 subgroups by Gaussian Mixture Model

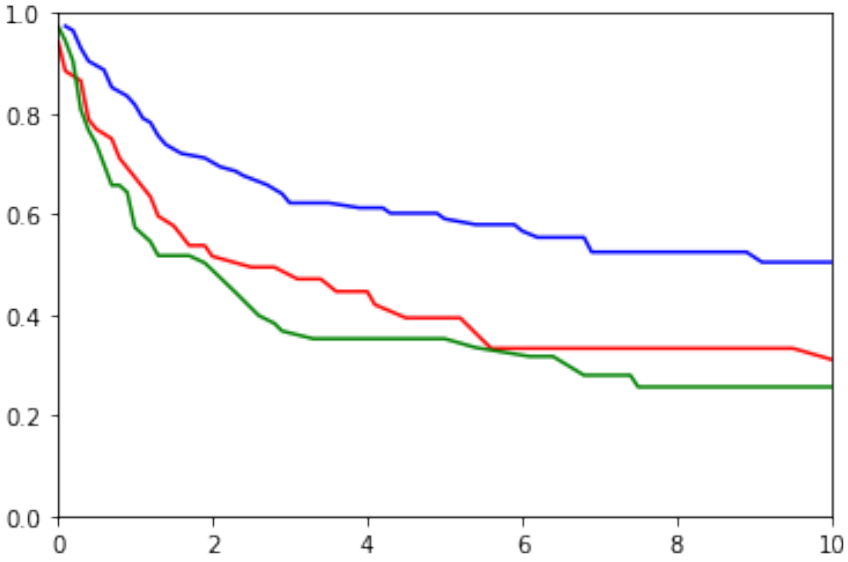

Figure 12: The Kaplan-Meier curves of the 3 subgroups in the paper [1]

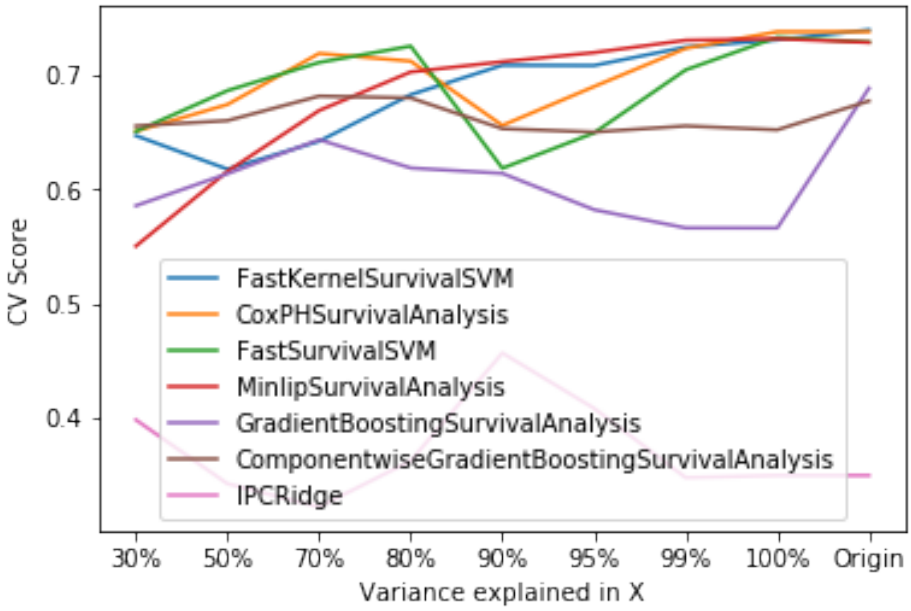

Figure 13: The Cross-validation Concordance scores on different model and different X data sets

7.3 Code Description

I used **R** and **Python** in this project. All **Python** codes and results were exported to HTML pages, so it is easy for reading (no need to install **Python**).

After deadline, all project files will be uploaded to `https://github.com/yanglyuxun/multivariate_gene_analysis`.

7.3.1 Code Files

- **1_Imputation.R**: R codes for imputation experiments.
- **2_FeatureSelection.ipynb** (**.html**): Python codes for GBDT based feature selection.
- **3_Dimensionality_reduction.R**: R codes for the feature extraction experiments.
- **3.1_FactorAnalysis.ipynb** (**.html**): Python codes for factor analysis. It is too slow in R, so utilize the Python parallel computation.
- **3.2_GaussianMixture.ipynb** (**.html**): Python codes for GMM.
- **4_Survival_Machine_Learning.ipynb** (**.html**): Python codes for other machine learning parts.

7.3.2 Environments

- Linux Mint 18.2 Cinnamon 64-bit (Kernel 4.10.0-35-generic)
- R 3.4.4 x86_64-pc-linux-gnu (64-bit)
- Python 3.6.4 (64-bit)

References

[1] Rosenwald, Andreas, et al. "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma." *New England Journal of Medicine* 346.25 (2002): 1937-1947.

[2] Alizadeh, Ash A., et al. "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling." *Nature* 403.6769 (2000): 503.

[3] Chiu, Chia-Chun, et al. "Missing value imputation for microarray data: a comprehensive comparison study and a web tool." *BMC systems biology* 7.6 (2013): S12.

[4] Troyanskaya, Olga, et al. "Missing value estimation methods for DNA microarrays." *Bioinformatics* 17.6 (2001): 520-525.

[5] Kim, Hyunsoo, Gene H. Golub, and Haesun Park. "Missing value estimation for DNA microarray gene expression data: local least squares imputation." *Bioinformatics* 21.2 (2004): 187-198.

[6] Christopher, M. Bishop. PATTERN RECOGNITION AND MACHINE LEARNING. Springer-Verlag New York, 2016. Chapter 9,12.

[7] Tipping, Michael E., and Christopher M. Bishop. "Probabilistic principal component analysis." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999): 611-622.

[8] Oba, Shigeyuki, et al. "A Bayesian missing value estimation method for gene expression profile data." *Bioinformatics* 19.16 (2003): 2088-2096.

[9] Tuikkala, Johannes, et al. "Missing value imputation improves clustering and interpretation of gene expression microarray data." *BMC bioinformatics* 9.1 (2008): 202.

[10] Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis." *International Conference on Artificial Neural Networks.* Springer, Berlin, Heidelberg, 1997.

[11] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. New York: *Springer series in statistics*, 2001. Chapter 12, 14.

[12] Lueks, Wouter, et al. "How to evaluate dimensionality reduction?-improving the co-ranking matrix." *arXiv preprint*:1110.3917 (2011).

[13] Székely, Gábor J., Maria L. Rizzo, and Nail K. Bakirov. "Measuring and testing dependence by correlation of distances." *The annals of statistics* (2007): 2769-2794.

[14] Van Belle, Vanya, et al. "Support vector machines for survival analysis." *Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare.* 2007.

[15] Pölsterl, Sebastian, Nassir Navab, and Amin Katouzian. "Fast training of support vector machines for survival analysis." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, Cham, 2015.

[16] Pölsterl, Sebastian, Nassir Navab, and Amin Katouzian. "An Efficient Training Algorithm for Kernel Survival Support Vector Machines." *arXiv preprint* arXiv:1611.07054 (2016).

[17] Harrell, Frank E., Kerry L. Lee, and Daniel B. Mark. "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in medicine* 15.4 (1996): 361-387.