Supplementary Materials to Robust biomarker screening from gene expression data by stable machine learning-recursive feature elimination methods (Methodology)

Lingyu Li[a], Zhi-Ping Liu[a,*]

[a]*School of Control Science and Engineering, Shandong University, Jinan, Shandong 250061, China*

Abstract

Recently, identifying robust biomarkers or signatures from gene expression profiling data has attracted much attention in computational biomedicine. The successful discovery of biomarkers for complex diseases such as spontaneous preterm birth (SPTB) and high-grade serous ovarian cancer (HGSOC) will be beneficial to reduce the risk of preterm birth and ovarian cancer among women for early detection and intervention. In this paper, we propose a stable machine learning-recursive feature elimination (StabML-RFE for short) strategy for screening robust biomarkers from high-throughput gene expression data. We employ eight popular machine learning methods, namely AdaBoost (AB), Decision Tree (DT), Gradient Boosted Decision Trees (GBDT), Naive Bayes (NB), Neural Network (NNET), Random Forest (RF), Support Vector Machine (SVM) and XGBoost (XGB), to train on all feature genes of training data, apply recursive feature elimination (RFE) to remove the least important features sequentially, and obtain eight gene subsets with feature importance ranking. Then we select the top-ranking features in each ranked subset as the optimal feature subset. We establish a stability metric aggregated with classification performance on test data to assess the robustness of the eight different feature selection techniques. Finally, StabML-RFE chooses the high-frequent features in the subsets of the combination with maximum stability value as robust biomarkers. Particularly, we verify the screened biomarkers not only via internal validation, functional enrichment analysis and literature check, but also via external validation on two real-world SPTB and HGSOC datasets respectively. Obviously, the proposed StabML-RFE biomarker discovery pipeline easily serves as a model for identifying diagnostic biomarkers for other complex diseases from omics data.

Keywords: Robust biomarker discovery, Machine learning, Recursive feature elimination, Stable feature selection, Spontaneous preterm birth, High-grade serous ovarian cancer

1. Machine learning classifiers

In this work, we simply focus on the binary classification scenario. Supposing there are n samples of observations independently and identically distributed with the training dataset

$$\mathcal{D} = \{(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), \cdots, (\boldsymbol{X}_n, y_n\}, \tag{1}$$

where $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^{\mathrm{T}} \in \mathbb{R}^p$ is a p-dimensional gene expression data of the i-th sample, and $y_i \in \{0, 1\}$ is a label variable for $\boldsymbol{X}_i$ with $i = 1, 2, \cdots, n$. There are numerous popular algorithms for binary classification. Here we use the following eight machine learning methods, namely AdBoost (AB), Decision Tree (DT), Gradient Boosted Decision Trees (GBDT), Naive Bayesian (NB), Neural Network (NNET), Random Forest (RF), Support Vector Machine (SVM) and XGBoost (XGB), to learn on the discovery data.

* Corresponding author
Email address: zpliu@sdu.edu.cn (Zhi-Ping Liu)

1.1. Adboost

In the classification problem, the goal of the predictive model Adboost (AB) is to identify the class that generated a particular instance [1]. AB is inferred over a collection of labeled observations provided as tuples $(\boldsymbol{X}_i, y_i)$ containing the instance vector $\boldsymbol{X}_i$ and the true label variable y_i. Considering a classifier H that predicts the category of an instance $\boldsymbol{X}_i \in \mathbb{R}^p$ as

$$\hat{y}_H = H(\boldsymbol{X}_i), \tag{2}$$

where $\hat{y}_H \in \{-1, +1\}$. Then the error rate on the training dataset $\mathcal{D}$ is

$$\text{error}_H = \frac{1}{n} \sum_{i=1}^{n} \mathcal{J}\left(y_i \neq H\left(\boldsymbol{X}_i\right)\right), \tag{3}$$

where $\mathcal{J}\left(y_i \neq H\left(\boldsymbol{X}_i\right)\right)$ is the indicator function, which is 1 if the prediction is wrong.

AB aims to sequentially train the classifier on the modified version of the training set to produce a sequence of classifiers H_m for $m \in \{1, 2, \cdots, M\}$. The collective predictive model of this ensemble is a linear combination of the predictions of the individual classifiers in the ensemble, so the final classifier is

$$H(\boldsymbol{X}_i) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m H_m(\boldsymbol{X}_i)\right), \tag{4}$$

where α_m is the weight assigned to the classifier H_m during the training process.

1.2. Decision tree

For the given the training dataset $\mathcal{D}$ with vector $\boldsymbol{X}_i \in \mathbb{R}^p$ and label vector y, a decision tree (DT) recursively partitions the feature space such that the samples with the same label values are grouped together [2]. Let the data at node m be represented by Q_m with N_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m partition the data into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets,

$$Q_m^{left}(\theta) = \left\{(\boldsymbol{X}_i, y) \mid x_{ij} <= t_m\right\}, \quad Q_m^{right}(\theta) = Q_m \backslash Q_m^{left}(\theta). \tag{5}$$

Let the label of the classification outcome equal to $0, 1, \cdots, k-1$, for node m, let $p_{mk} = \frac{1}{N_m} \sum_{y \in Q_m} I(y = k)$ be the proportion of class k observations in node. Useing the Gini function as the measure of impurity, it holds

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}). \tag{6}$$

Then the quality of a candidate split of node m is then computed using the impurity function (6),

$$G\left(Q_m, \theta\right) = \frac{N_m^{left}}{N_m} H\left(Q_m^{left}(\theta)\right) + \frac{N_m^{right}}{N_m} H\left(Q_m^{right}(\theta)\right), \tag{7}$$

Finally, selecting the parameters that minimises the impurity

$$\theta^* = \arg\min_{\theta} G(Q_m, \theta), \tag{8}$$

with recursing for subsets $Q_m^{left}(\theta^*)$ and $Q_m^{right}(\theta^*)$ until the maximum allowable depth is reached, where $N_m < \min_{samples}$ or $N_m = 1$.

1.3. Gradient boosted decision trees

The purpose of gradient boosted decision trees (GBDT) is to find a model F to fit the data such that the predicted value $\hat{y}_i$ for the j-th training example $\boldsymbol{X}_i$ is approximately equal to the j-th target value y_i or equivalently $\hat{y}_i = F(\boldsymbol{X}_i) \sim y_i$ for $\forall i = 1, 2, \cdots, n$ [3]. In binomial classification, let the loss function be $\mathscr{L}(y, \hat{y})$, GBDT will minimize the loss function that reflects how badly the model $\hat{y}$ currently fits the data y.

GBDT iteratively updates the $\hat{y}$ in the direction of the negative gradient of the loss function as follows,

$$\hat{y}_i \to \hat{y}_i - \eta \frac{\partial \mathscr{L}}{\partial \hat{y}_i}, \tag{9}$$

where η is the learning rate. Equivalently, GBDT updates F_{m+1} by adding another "delta model" f_{m+1},

$$\hat{y}_i = F_m(\boldsymbol{X}_i) + f_{m+1}(\boldsymbol{X}_i) \quad \forall i \in 1, \cdots, n, \tag{10}$$

where $f_{m+1}(\boldsymbol{X}_i) = -\eta \frac{\partial \mathscr{L}}{\partial \hat{y}_i}$, which is fitted using a decision tree and each tree will have its own loss $\mathscr{L}^{f_{m+1}}$ separate to the global loss function $\mathscr{L}$.

1.4. Naive Bayes

Given data $\boldsymbol{X}_i$ of p features, naive Bayes (NB) predicts the class C_k for $\boldsymbol{X}_i$ according to the probability

$$p(C_k|\boldsymbol{X}_i) = p(C_k|x_{i1}, x_{i2}, \cdots, x_{ip}), \quad \forall k \in 1, \cdots, K, \tag{11}$$

where K is all possible outcomes. Using the Bayes Theorem [4], Equation (11) can be factored as

$$p(C_k|\boldsymbol{X}_i) = \frac{p(\boldsymbol{X}_i|C_k)p(C_k)}{p(\boldsymbol{X}_i)} = \frac{p(x_{i1}, x_{i2}, \cdots, x_{ip}|C_k)p(C_k)}{p(x_{i1}, x_{i2}, \cdots, x_{ip})}. \tag{12}$$

Combining the Chain Rule, $p(x_{i1}, x_{i2}, \cdots, x_{ip}|C_k)$ in Equation (12) can be further decomposed as

$$p(x_{i1}, x_{i2}, \cdots, x_{ip}|C_k) = p(x_{i1}|x_{i2}, \cdots, x_{ip}, C_k)p(x_{i2}|x_{i3}, \cdots, x_{ip}, C_k) \cdots p(x_{i,p-1}|x_{ip}, C_k)p(x_{ip}, C_k). \tag{13}$$

Specifically, for the given the class C_k, the NB model assumes that feature x_{il} is independent of feature x_{ij} for $l \neq j$, it is to say $p(x_{ij}|x_{i,j+1}, \cdots, x_{ip}|C_k) = p(x_{ij}|C_k)$, so it holds

$$p(x_{i1}, \cdots, x_{ip}|C_k) = \prod_{j=1}^{p} p(x_{ij}|C_k). \tag{14}$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k|x_{i1}, x_{i2}, \cdots, x_{ip}) &\propto p(C_k, x_{i1}, x_{i2}, \cdots, x_{ip}) \\ &\propto p(C_k)p(x_{i1}, x_{i2}, \cdots, x_{ip}|C_k) \\ &\propto p(C_k)p(x_{i1}|C_k)p(x_{i2}|C_k) \cdots p(x_{ip}|C_k) \\ &\propto p(C_k) \prod_{j=1}^{p} p(x_{ij}|C_k). \end{aligned} \tag{15}$$

where $\propto$ denotes proportionality.

For the classification problem, the NB model gives the probability of a data point $\boldsymbol{X}_i$ belonging to class C_k as proportional to a simple product of $n + 1$ factors (the class prior $p(C_k)$) plus n conditional feature probabilities $p(\boldsymbol{X}_i|C_k)$ [4]. The most likely class assignment for a data point $\boldsymbol{X}_i$ can be found by calculating $p(C_k) \prod_{j=1}^{p} p(x_{ij}|C_k)$ for $k = 1, 2, \cdots, K$ and assigning $\boldsymbol{X}_i$ the class C_k for which this value is largest. In mathematical notation, it is defined

$$\hat{C} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k), \tag{16}$$

where $\hat{C}$ is the estimated class for $\boldsymbol{X}_i$ given its features $x_{i1}, x_{i2}, \cdots x_{ip}$.

1.5. Neural network

For neural network (NN) model, the formal neuron has p inputs $x_{i1}, x_{i2}, \cdots, x_{ip}$ that model the signals coming from dendrites [5]. The inputs are labeled with the corresponding synaptic weights $w_1, w_2, \cdots, w_p$ that measure their permeabilities, where some of these synaptic weights may be negative. Then, the weighted sum of input values is used to represent the excitation level of the neuron,

$$\xi_i = \sum_{j=1}^{p} w_j x_{ij}. \tag{17}$$

After reaching the threshold h, the value of excitation level ξ_i induces the output y_i of the neuron, where the nonlinear grow of output $y = \sigma(x)$ is determined by the activation function σ. The simplest type of activation function is the hard limiter as follows,

$$\sigma(\xi_i) = \begin{cases} 1, & \text{if } \xi_i \geq h, \\ 0, & \text{if } \xi_i < h. \end{cases} \tag{18}$$

By a formal manipulation, the function σ can have zero thresholds while the actual threshold with the opposite sign is seen as a further weight, bias $w_0 = -h$ of additional formal input $x_{i0} = 1$ with constant unit value. Then the mathematical formulation of neuron function is as follows,

$$y_i = \sigma(\xi_i) = \begin{cases} 1, & \text{if } \xi_i \geq 0, \\ 0, & \text{if } \xi_i < 0, \end{cases} \tag{19}$$

where $\xi_i = \sum_{j=0}^{p} w_j x_{ij}$.

1.6. Random forest

For the binary classification problem, random forest (RF) constructs many individual decision trees at training and pools the predictions from all trees to make the final prediction of the classes [6, 7]. For each decision tree, assuming only two child nodes, it holds

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}, \tag{20}$$

where ni_j is the importance of node j, w_j is the weighted number of samples reaching node j, C_j is the impurity value of node j, $left(j)$ is child node from left split on node j and $right(j)$ is child node from right split on node j.

RF calculates the importance of a node using Gini importance, where the importance for each feature on a decision tree is calculated as

$$fi_i = \frac{\sum_{j:\text{ node } j \text{ splits on feature } i} ni_j}{\sum_{k\in \text{ all nodes}} ni_k}, \tag{21}$$

where fi_i and fi_j are the importance of feature i and j, respectively. Dividing by the sum of all feature importance values, Equation (21) can be normalized as follows,

$$\text{norm } fi_i = \frac{fi_i}{\sum_{j\in \text{ all features}} fi_j}, \tag{22}$$

where $\text{norm } fi_i \in [0, 1]$. Then the final feature importance of the RF method is the average over all the trees calculating as follows,

$$RFfi_i = \frac{\sum_{j\in \text{ all trees}} \text{norm } fi_{ij}}{T}, \tag{23}$$

where $RFfi_i$ represents the importance of feature i calculated from all trees in the RF model, $\text{norm } fi_{ij}$ represents the normalized feature importance for i in tree j and T is the total number of trees.

1.7. Support vector machine

Support vector machine (SVM) is a classifier defined in the feature space to maximize the interval between two classes [8, 9, 10], which is widely used in the fields of pattern recognition, data mining and machine learning. When the data is linearly separable, SVM finds the optimal classification hyperplane $\boldsymbol{\omega}^{\mathrm{T}} \cdot \boldsymbol{X}_i + b = 0$ and separates the samples of the two classes completely. The optimization problem is,

$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } y_i\left(\boldsymbol{\omega}^{\mathrm{T}} \cdot \boldsymbol{X}_i + b\right) \geq 1 - \xi_i, \quad i = 1, 2, \cdots, n, \\ &\qquad \xi_i \geq 0, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{24}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_p)^{\mathrm{T}}$ represents the normal vector of the hyperplane (the weight vector of the feature), ξ_i $(i = 1, 2, \cdots, n)$ are the slack variables, b is the bias, C is the penalty parameter used to balance between minimizing the error and maximizing the interval.

Equation (24) is called the *soft margin SVM* and can be transformed into the following dual problem,

$$\begin{aligned} &\min \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\left(\boldsymbol{X}_i^T\boldsymbol{X}_j\right) - \sum_{i=1}^{n}\alpha_i \\ &\text{s.t. } \sum_{i=1}^{n}\alpha_i y_i = 0, \quad i = 1, 2, \ldots, n, \\ &\qquad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots n, \end{aligned} \tag{25}$$

where $\alpha_i \geq 0$ $(i = 1, 2, \ldots, n)$ are the *Lagrange* multipliers. As one of the classic quadratic programming problems, applying the technique of convex optimisation to solve Problem (25), it derives

$$\boldsymbol{\omega} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{X}_i. \tag{26}$$

while b can be simply calculated from the first equality constraint formula in Equation (24) with $\alpha_i > 0$ $(i = 1, 2, \ldots, n)$.

1.8. XGBoost

XGBoost (XGB) model is a scalable machine learning system for tree boosting [11]. It uses K additive trees to create the ensemble model,

$$\hat{y} = \hat{f}(x) = \sum_{i=0}^{K}\hat{f}_i(x), \quad \hat{f}_i(x) \in F, \tag{27}$$

and $F = f(x) = w_{q(x)}$ with $q : \mathbb{R}^m \to T$, $w \in \mathbb{R}^T$, in which q represents the structure of the tree that maps an input to the corresponding leaf index at which it ends up being, T is the number of leaves in the tree, and w_i represents the score on i-th leaf contained in its regression tree.

For learning the set of functions used in model (27), XGB model minimize the following regularized objective,

$$\mathscr{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right), \tag{28}$$

with $\Omega\left(f_k\right) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, where l is a differentiable convex loss function that measures the difference between the prediction y_i and the target y_i, Ω is a regularized term for number of leaf nodes used to penalizes the complexity of the regression tree functions so as to smooth the final learnt weights to avoid over-fitting.

For the t-th iteration, it needs to add f_t to minimize the following objective function,

$$\mathscr{L}^{(t)} = \sum_i l\left(\hat{y}_i^{t-1} + f_t\left(x_i\right), y_i\right) + \sum_t \Omega\left(f_t\right), \tag{29}$$

Using Taylor series expansion and ignoring higher order terms, the second-order approximation of objective function (29) can be written as

$$\mathscr{L}^{(t)} = \sum_i^n \left[l\left(\hat{y}_i^{t-1}, y_i\right) + \partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_t\left(x_i\right) + \frac{1}{2}\partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_i^2\left(x_i\right)\right] + \Omega\left(f_t\right), \tag{30}$$

The approximation (30) helps in getting a closed form optimal solution for w. Removing the constant terms to simplify the objective function, it holds

$$\mathscr{L}^{(t)} = \sum_i^n \left[\partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_t\left(x_i\right) + \frac{1}{2}\partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_i^2\left(x_i\right)\right] + \Omega\left(f_t\right). \tag{31}$$

Let $I_j = \{i|q(x_i) = j\}$ be the set of all the input data points that ended up in j-th leaf node, then Equation (31) can be written as

$$\begin{aligned}
\mathscr{L}^{(t)} &= \sum_i^n \left[\partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_t\left(x_i\right) + \frac{1}{2}\partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) f_i^2\left(x_i\right)\right] + \gamma T + \frac{1}{2}\lambda\|w\|^2 \\
&= \sum_{j=1}^T \left[\left(\sum_{i\in I_j} \partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right)\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} \partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) + \lambda\right) w_j^2\right] + \gamma T.
\end{aligned} \tag{32}$$

For a fixed tree structure, the optimal weight w_j^* of the leaf j can be computed by differentiating the Equation (32) with respect to w and equating to 0,

$$w_j^* = -\frac{\sum_{i\in I_j} \partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right)}{\sum_{i\in I_j} \partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) + \lambda}, \tag{33}$$

Given any tree structure, Equation (33) can find what the optimal leaf node weights should be. Next, it needs to find the optimal tree structure that minimizes the loss. By replacing w_j^* in Equation (33), the corresponding optimal value for given tree structure q can be calculated as follows,

$$\mathscr{L}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^T \frac{\left[\sum_{i\in I_j} \partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right)\right]^2}{\sum_{i\in I_j} \partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right) + \lambda} + \gamma T. \tag{34}$$

Considering that it is impossible to enumerate all the possible tree structures, the XGB model uses weighted quantile sketch to decide the candidate split points so as to calculus the total loss after split minus total loss before the split. Assume that I_L and I_R are the instance sets of left and right nodes after the split, let $I = I_L \cup I_R$, then the loss reduction after the split is given by

$$\begin{aligned}
\mathscr{L}_{\text{split}} &= \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma(T + 1 - T) \\
&= \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma.
\end{aligned} \tag{35}$$

where $g_i = \partial_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right)$ and $h_i = \partial^2_{\hat{y}^{t-1}} l\left(\hat{y}_i^{t-1}, y_i\right)$.

2. Choice of parameters

In this section, we introduce all chosen parameters used in all eight methods. Table 1 clearly illustrates the Classifier functions, Parameters setting and Packages. Whats more, all the codes can be found at `https://github.com/zpliulab/StabML-RFE`.

Table 1: The choice of the parameters used in eight ML-RFE methods.

Methods	Classifier functions	Parameters setting	Packages
AB-RFE	AdaBoostRegressor	random_state=2022, n_features_to_select = 1	sklearn.ensemble in Python
DT-RFE	DecisionTreeClassifier	random_state=2022, n_features_to_select = 1	sklearn.tree in Python
GBDT-RFE	GradientBoostingClassifier	random_state=2022, n_features_to_select = 1	sklearn.ensemble in Python
NB-RFE	BernoulliNB	n_features_to_select = 1	sklearn.naive_bayes in Python
NNET-RFE	rfe	method = "nnet", tuneGrid = expand.grid(size = c(8),decay = c(0.1)),maxit = 30, MaxNWts = 100000	caret in R
RF-RFE	RandomForestClassifier	random_state=2022, n_features_to_select = 1	sklearn.ensemble in Python
SVM-RFE	svm, kernel='linear'	random_state=2022, n_features_to_select = 1	sklearn in Python
XGB-RFE	XGBRegressor	random_state=2022, n_features_to_select = 1	xgboost in Python

References

[1] Y. Freund, R. E. Schapire, et al., Experiments with a new boosting algorithm, in: Procof International Conference on Machine Learning, volume 96, Citeseer, 1996, pp. 148–156.

[2] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[3] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of Statistics (2001) 1189–1232.

[4] P. Domingos, M. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Machine Learning 29 (1997) 103–130.

[5] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: Neural Networks for Perception, Elsevier, 1992, pp. 65–93.

[6] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

[7] J. Li, Z. Chen, L. Tian, C. Zhou, M. Y. He, Y. Gao, S. Wang, F. Zhou, S. Shi, X. Feng, et al., Lncrna profile study reveals a three-lncrna signature associated with the survival of patients with oesophageal squamous cell carcinoma, Gut 63 (2014) 1700–1710.

[8] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

[9] C.-Y. Sun, T.-F. Su, N. Li, B. Zhou, E.-S. Guo, Z.-Y. Yang, J. Liao, D. Ding, Q. Xu, H. Lu, et al., A chemotherapy response classifier based on support vector machines for high-grade serous ovarian carcinoma, Oncotarget 7 (2016) 3245–3254.

[10] Z.-H. Zhu, B.-Y. Sun, Y. Ma, J.-Y. Shao, H. Long, X. Zhang, J.-H. Fu, L.-J. Zhang, X.-D. Su, Q.-L. Wu, et al., Three immunomarker support vector machines–based prognostic classifiers for stage ib non–small-cell lung cancer, Journal of Clinical Oncology 27 (2009) 1091–1099.

[11] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.