Journal of Data Science 19 (4), 634–657
October 2021
DOI: 10.6339/21-JDS1022
Data Science Reviews

Statistical Learning in Medical Research with Decision Threshold and Accuracy Evaluation

Sumaiya Z. Sande[1], Loraine Seng[1,*], Jialiang Li[1], and Ralph D'Agostino[2]

[1]*Department of Statistics and Data Science, National University of Singapore, Singapore*
[2]*Department of Mathematics and Statistics, Boston University, Massachusetts, USA*

Abstract

Machine learning methods are increasingly applied for medical data analysis to reduce human efforts and improve our understanding of disease propagation. When the data is complicated and unstructured, shallow learning methods may not be suitable or feasible. Deep learning neural networks like multilayer perceptron (MLP) and convolutional neural network (CNN), have been incorporated in medical diagnosis and prognosis for better health care practice. For a binary outcome, these learning methods directly output predicted probabilities for patient's health condition. Investigators still need to consider appropriate decision threshold to split the predicted probabilities into positive and negative regions. We review methods to select the cut-off values, including the relatively automatic methods based on optimization of the ROC curve criteria and also the utility-based methods with a net benefit curve. In particular, decision curve analysis (DCA) is now acknowledged in medical studies as a good complement to the ROC analysis for the purpose of decision making. In this paper, we provide the R code to illustrate how to perform the statistical learning methods, select decision threshold to yield the binary prediction and evaluate the accuracy of the resulting classification. This article will help medical decision makers to understand different classification methods and use them in real world scenario.

Keywords *deep learning; machine learning; net benefit; ROC; threshold*

1 Introduction

Data science has expanded quickly due to the increase in data storage capacities and exploration of computational technologies and algorithms. Medical researchers now can access large volume of data and analyze them in real time. The data mining techniques help to obtain the significant information from the patient health data and make promising predictions. In a particular clinical investigation, we may follow a three-step procedure: (i) build a risk prediction model from available data by using machine learning methods; (ii) decide a threshold value to split the predicted probability into either a positive or a negative region; (iii) apply the trained model and selected threshold for actual diagnosis and screening. We will provide a systematic review on how to conduct steps (i) and (ii) in this paper.

To perform step (i), one may consider many machine learning methods. When data are in the standard format, e.g., accessible via an Excel sheet, most shallow learning tools can be readily applied, including the familiar logistic regression, and classification trees for example. These methods are traditionally covered in the course curriculum in most graduate programs for statistics and biostatistics. On the other hand, when the data become complicated, we may

*Corresponding author. Email: loraine_seng@u.nus.edu.

Received May 10, 2021; Accepted August 18, 2021

need deep learning neural networks such as multilayer perceptron (MLP) and convolutional neural network (CNN). These more advanced learning can be used in clinical diagnostic tasks to improve the prediction results of shallow learning methods or for analyzing unstructured data such as medical imaging data. In fact medical images can come from computed tomography, magnetic resonance, ultrasound, X-ray etc. Medical image segmentation, pattern recognition and visualization has become vital for the early detection, diagnosis and treatment of many diseases. In this study we illustrate the traditional shallow learning methods and the deep learning methods on the Pima Indian diabetes dataset and the Malaria Blood smear image dataset, respectively.

To perform step (ii), one may choose the decision threshold based on two types of consideration. The first option is to directly optimize some criteria related to Receiver Operating Characteristic (ROC) curve. ROC analysis has been frequently used for the evaluation of diagnostic models (Pepe et al., 2003; Zhou et al., 2009; Erkanli et al., 2006; Krzanowski and Hand, 2009; O'Malley and Zou, 2006; Zou et al., 2012; Li et al., 2019). The so-called ROC curve is a plot of sensitivity against 1-specificity across all the possible decision thresholds from a diagnostic model. Area under the ROC curve (AUC) is a commonly reported measure to summarize overall accuracy of the diagnostic model. One may then select some "optimal" cut-off values from the constructed ROC graphs when satisfactory levels of sensitivity and specificity are achieved.

The second option for deciding a threshold is via a utility-based analysis. In fact, despite of its vast popularity, ROC curve fails to account for the clinical consequence and the practical cost-utility of the learned models (Vickers et al., 2019). The cost of a decision could mean the acquisition costs of the product, but also the costs of any additional tests or visits to health-care professional that a patient needs to make. The benefit or utility from the decision is understood as the gain in health a patient receives. It is usually difficult for medical practitioners to conduct utility analysis without external data. Decision curve analysis (DCA; Vickers and Elkin, 2006) is a formal statistical tool to overcome this problem by comparing the model with the default strategies treating all patients and treating no one irrespective of the model. Net benefit function (Vickers et al., 2012) is constructed to place net benefit and harm on the same scale for the possible range of threshold probabilities of the disease (probability at which the decision to undergo treatment or not; Zhang et al., 2018; Fitzgerald et al., 2015). DCA helps to compare default strategies with the learned model for the range of thresholds in terms of model utility. One can then make an informed decision by examining the net benefit graphs. Inference for net benefit has been established in Sande et al. (2020).

This study is to provide a systematic review of statistical learning tools for predicting patient's medical conditions and decision thresholding methods for aforementioned purposes (i) and (ii). To this end, the paper is organized as follows. Section 2 starts by introducing the data setting assumed throughout this paper. In Section 3, we provide a comprehensive review of the shallow supervised learning methods for binary classification. In Section 4, we provide a sketch of crucial components involved in training a deep learning model. In Section 5, we discuss different approaches to find the threshold probability. Then, in Section 6, we analyze two medical datasets for an illustration. We conclude with some discussions in Section 7.

2 Set Up

Consider a set of predictors $\Omega = (X_1, X_2, \ldots, X_m)$ such that $X_j \in \mathbb{R}$ and Y be the binary response variable which takes values 1 and 0 for diseased and non-diseased status respectively.

Let the prevalence of the disease be $\pi = \Pr(Y = 1)$. Suppose we have a training sample $\boldsymbol{X} = \{x_{ij}; 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$ where the covariate vector for the ith row is denoted by $\mathcal{X}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$. The columns denote the m predictor variables where the jth predictor is denoted by $\boldsymbol{X}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T$, $1 \leqslant j \leqslant m$. We also observe the corresponding response variable $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_n)^T$ for the sample. We train a model $\mathcal{M}$ based on a subset of predictors $\subseteq \{X_1, X_2, \ldots, X_m\}$ using any of the techniques to be reviewed in this section. We can then obtain the model-based predictive probability as $p = \Pr(Y = 1|\mathcal{M})$. For a sample $\{\boldsymbol{X}, \boldsymbol{Y}\}$, the predicted probability for the ith individual is given as $p_i = \Pr(Y_i = 1|\mathcal{M})$. Denote $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)^T$ as the predicted probabilities for the n individuals in the sample. If we consider a risk threshold $c \in (0, 1)$ for assessment, the binary decision rule is: the ith subject with $p_i \geqslant c$ will be assigned into class 1; otherwise in class 0.

In the next two sections we consider different statistical learning methods to obtain $\boldsymbol{p}$. We will introduce how to select threshold c in the Section 5.

3 Shallow Learning Methods

3.1 Generalized Linear Model (GLM)

GLM (Nelder and Wedderburn, 1972) include many widely used statistical models. Let $\mu = E(Y|X_1, X_2, \ldots, X_m)$ be the mean function and adopt a link function $g(\cdot)$ such that

$$g(\mu) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m,$$

where β_j's are the regression coefficients in the GLM. In this case with a binary response we have $\mu = p$. There are a few choices of link functions $g(\cdot)$ available for binary outcome. The most popular choice in medical study is the logit link. We may consider the logistic regression model by using such a link function as

$$g(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m.$$

Equivalently, logistic regression computes the predictive probability p by

$$p = \frac{\exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m)}{1 + \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m)}. \tag{1}$$

The model can be fitted by maximizing the log-likelihood function, or equivalently, minimizing the negative log-likelihood given by

$$l\left(Y_i, \beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right) = -Y_i\left(\beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right) + \log\left(1 + \exp\left(\beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right)\right).$$

After fitting the model, we obtain regression coefficient estimates $\hat{\beta} = \{\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_m\}$ and then the estimate of predictive probability $\hat{p}$ are obtained for all the subjects in the sample by substituting β_j's in (1) with $\hat{\beta}_j$'s. In `R, glm` function can be used for binary logistic regression by specifying `family = binomial(link = "logit")` in the arguments.

Another popular link function for GLM is the probit link under which we attain a probit regression model. The probit link function is given as follows

$$g(p) = \Phi^{-1}(p) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m,$$

where $\Phi(.)$ is a standard normal cumulative distribution function.

Implementing a probit model is similar to logistic regression in R, i.e. using the `glm` function but with `family = binomial(link = "probit")`.

3.2 Regularized GLM

When the predictor dimension m is very large, GLM may be unstable or infeasible. Shrinkage methods (Friedman et al., 2001) provide solution to this problem by adding penalty constraints to the model fitting objective function. The following two types of estimators are used in the literature by imposing L_2 and L_1 penalty functions, respectively:

$$\text{Ridge:} \quad \hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} l\left(Y_i, \beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right) + \frac{\lambda}{2} \sum_{j=1}^{m} \beta_j^2 \right\},$$

$$\text{LASSO:} \quad \hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} l\left(Y_i, \beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right) + \lambda \sum_{j=1}^{m} |\beta_j| \right\},$$

where $\lambda \geqslant 0$ is a regularization parameter controlling the amount of shrinkage. Ridge regression is a classical approach to reduce multicolinearity and reduce variance by shrinking the coefficients. LASSO performs variable selection due to the property of the L_1 penalty. The combination of LASSO and Ridge leads to the elastic net:

$$\hat{\beta}^{net} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} l\left(Y_i, \beta_0 + \sum_{j=1}^{m} x_{ij}\beta_j\right) + \lambda\left[0.5(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1\right],$$

where $0 \leqslant \alpha \leqslant 1$ is a weight parameter. We note that $\alpha = 0$ corresponds to ridge regression and $\alpha = 1$ corresponds to LASSO.

In `R`, regularized GLM is implemented by the package `glmnet` (Friedman et al., 2010). The elastic net penalty is controlled by the option `alpha`, and bridges between LASSO (`alpha = 1`, the default) and ridge regression (`alpha = 0`). The tuning parameter λ controls the overall strength of the penalty and is usually selected by a cross-validation (CV) method. After obtaining the regression coefficient $\hat{\beta}$, we can compute the class probabilities $\hat{p}$ for classification.

3.3 Generalized Additive Model (GAM)

Generalized additive model (GAM) was originally introduced by Trevor Hastie and Robert Tibshirani (Hastie and Tibshirani, 1986) as a nonlinear extension of GLM. In many data applications, GAM has proven to be a powerful statistical learning method. Following a similar structure to GLM, GAM (Hastie, 2017) can be given by

$$g(\mu) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_m(X_m),$$

where $f_1(.), f_2(.), \ldots, f_m(.)$ are unspecified smooth non-parametric functions. GLM can be understood as a special case of GAM when each f function is linear. Clearly, the main advantage

of GAM over GLM is that the covariate effect is not necessary to be linear. Typically, for binary response, we have $\mu = p$ and we may still use the logit link function $g(p) = \log(p/(1-p))$.

The building blocks of GAM is to estimate the functional effects for individual covariates. These functions are usually assumed to be smooth with derivatives. The most commonly adopted smoothers in the literature are running line smoother, locally estimated scatterplot smoother (loess), Regression splines and Smoothing splines. We offer more detailed review in the supplementary file.

In `R`, Generalized additive models are fitted by using `gam` package (Hastie, 2020a) or `mgcv` (Wood, 2003) package. `gam` package uses AIC criterion for the model selection while `mgcv` package uses any of the GCV/UBRE/REML/AIC criterion available in option `method` of the function `gam()`.

3.4 Support Vector Machine (SVM)

Support vector machine (SVM; Cortes and Vapnik, 1995) has been successful for many complicated classification tasks. An SVM solution constructs a decision boundary that explicitly separates the data into different classes. This boundary is referred as 'separating hyperplane', living in the multi-dimensional covariate space. Support vectors are data points closer to the hyperplane which impact the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.

Conventionally, the outcome is coded as -1 and 1, which is slightly different from the 0-1 coding for all other sections of this paper. To avoid confusion with what was defined at the beginning of Section 2, we denote the outcome in this subsection as $Y_i^* \in \{-1, 1\}$ and $\boldsymbol{Y}^* = (Y_1^*, Y_2^*, \ldots, Y_n^*)^T$. The feature data for ith subject is transformed from his/her original covariate $\mathcal{X}_i$. Let the hyperplane be given as

$$\boldsymbol{W}\psi(\boldsymbol{X}) + B = 0,$$

where B is a bias term, $\boldsymbol{W}$ is weight vector and $\psi(.)$ is a fixed transformation function such that $\psi : X \to Z$ where X is an m-dimensional input space and Z is a p-dimensional feature space. The objective is to classify the data points in such a way that the distance between the two classes is as wide as possible.

Instead of maximizing the margin, we can equivalently minimize the euclidean norm of the weight vector $\boldsymbol{W}$ with a constraint that the SVM predicted value and the actual response value share the same sign:

$$\min_{W,B} \frac{1}{2}\|\boldsymbol{W}\|_2,$$
$$\text{subject to} \quad Y_i^*\big(\boldsymbol{W}\psi(\mathcal{X}_i) + B\big) \geqslant 1, \quad i = 1, \ldots, n.$$

We solve the above optimization problem by the Lagrangian method which is given by

$$L(\boldsymbol{W}, B, \alpha) = \frac{1}{2}\boldsymbol{W}^T\boldsymbol{W} - \sum_{i=1}^{n} \alpha_i\big[Y_i^*\big(\boldsymbol{W}\psi(\mathcal{X}_i) + B\big) - 1\big].$$

The solution $\boldsymbol{W}^*$ is given by

$$\boldsymbol{W}^* = \sum_{i=1}^{n} \alpha_i^* Y_i^* \psi(\mathcal{X}_i),$$

where α_i^* are the optimal Lagrange multipliers. We define a real valued kernel function $K(\mathcal{X}_i, \mathcal{X}_j)$ $K : X \times X \to \mathbb{R}$ with the property $K(\mathcal{X}_i, \mathcal{X}_j) = \psi(\mathcal{X}_i)^T \psi(\mathcal{X}_j)$. Hence, we do not need the explicit coordinates in the feature space Z or the transformation function $\psi(.)$. The kernel function directly calculates the value of the dot product of the transformed data points in the feature space. The following are some common kernel functions used in SVM.

$$\begin{aligned}
\text{Linear Kernel:} \quad K(\mathcal{X}_i, \mathcal{X}_j) &= \sum_{k=1}^{m} x_{ik} x_{jk}. \\
\text{Radial/Gaussian Kernel:} \quad K(\mathcal{X}_i, \mathcal{X}_j) &= \exp\left[-\gamma \sum_{k=1}^{m} (x_{ik} - x_{jk})^2\right]. \\
\text{Polynomial Kernel:} \quad K(\mathcal{X}_i, \mathcal{X}_j) &= \left(b_0 + \gamma \sum_{k=1}^{m} x_{ik} x_{jk}\right)^d. \\
\text{Sigmoid Kernel:} \quad K(\mathcal{X}_i, \mathcal{X}_j) &= \tanh\left(b_0 + \gamma \sum_{k=1}^{m} x_{ik} x_{jk}\right).
\end{aligned}$$

SVM can be implemented in `R` using several packages. For example, one can use the `e1071` package (Meyer et al., 2019) where the `svm()` function provides a rigid interface to `libsvm` (Chang and Lin, 2011) along with visualization and parameter tuning methods. Package `kernlab` (Karatzoglou et al., 2004) features a variety of kernel-based methods and includes a SVM method based on the optimizers used in `libsvm` and `bsvm` (Hsu and Lin, 2002). It aims to provide a flexible and extensible SVM implementation. Package `klaR` (Weihs et al., 2005) includes an interface to `SVMlight`, a popular SVM implementation that additionally offers classification tools such as Regularized Discriminant Analysis. Another package `svmpath` (Hastie, 2020b) provides an algorithm that fits the entire path of the SVM solution. We have used the function `svm()` from the package `e1071` for the numerical works in this paper.

3.5 Decision Tree

As the name suggests, decision tree (Safavian and Landgrebe, 1991) is a tree-like algorithm to express the decision visually explicit. Decision tree can be used for both regression and classification. In classification trees, class label is assigned to the leaf while the conditions on features are represented by branches. This algorithm works upside-down by choosing a variable that splits the data with some decision rule. There are various tree algorithms developed such as ID3 (Iterative Dichotomiser 3), C4.5 (Salzberg, 1994), CART (classification and regression tree) (Breiman, 2017), CHAID (Chi-square automatic interaction detection) (Kass, 1980) and conditional inference tree (Hothorn et al., 2006). To avoid overfitting, a common strategy is to grow the tree until each node contains a small number of instances then use pruning to remove nodes that do not provide additional information. There are different prunning methods available such as cost complexity prunning (Breiman et al., 1984), minimum error prunning (Niblett and Bratko, 1987), pessimistic pruning (Quinlan, 1993), error based pruning (Quinlan, 1993) and minimum description length pruning (Mehta et al., 1995).

There are two types of splitting criteria – Univariate and multivariate. Univariate trees test only one attribute at a time while multivariate decision trees test more than one attributes at a node. A large portion of the multivariate splitting criteria is dependent on the linear combination of input predictors. It chooses not the best attribute but the best linear combination

of that attribute. This is usually done by greedy search, direct customizing, or direct discriminate dissection.

Different methods have been proposed to improve the accuracy of the posterior probabilities, including smoothing methods, specialized trees, combined methods (decision trees combined with other methods such as Naive Bayes), fuzzy methods and ensemble methods (Bagging and Boosting). Some of these make drastic changes in the fundamental properties of trees and compute probabilities by modifying the tree itself. Distance based probability estimates (Alvarez et al., 2007) have also been provided.

To implement classification tree in `R`, we may use `rpart` package (Therneau and Atkinson, 2019) and we can obtain posterior probabilities by using `predict()` function. This package implements univariate splitting. There are other `R` packages which implement the multivariate splitting, such as `mvpart` (Breiman et al., 1984), `optpart` (Roberts, 2020), `partDSA` (Molinaro et al., 2010) and `party` (Hothorn et al., 2006). We can perform the complexity pruning used with the `prune()` function in the same package by choosing an optimal complexity parameter (*cp*).

3.6 Random Forest

To improve the performance of decision trees, ensemble techniques are used where results from multiple classifiers are aggregated. Two well known methods are Bagging (Breiman, 1996) and Boosting (Schapire et al., 1998). In bagging multiple independent trees are constructed on the basis of bootstrap data points. Such bagging algorithms aim to reduce the complexity of tree models that overfit the training data. In contrast, boosting is an approach to increase the complexity of weak models that suffer from high bias, that is, models that underfit the training data.

Random Forest (RF) is actually a bagging method which comprises of large number of weak trees trained from different resampled data (Breiman, 2001). Every individual tree selects splitting variables (features) differently by randomization and hence reduces the chance of correlation among trees. This phenomenon is called 'feature bagging'. Each tree predicts the class and the class with maximum votes gives the final prediction.

To implement RF in `R`, we may use `randomForest()` function from `randomForest` package (Liaw and Wiener, 2002) and obtain posterior probabilities by `predict()` function.

3.7 Gradient Boosting Machine (GBM)

Gradient Boosting Machine (GBM; Friedman, 2001) is another ensemble learning approach. Boosting is a process that uses a set of machine learning algorithms to combine weak learners such as trees to form strong learner in order to increase the prediction accuracy of the model. In fact, bagging algorithm constructs models in parallel. Consequently, one model is not more superior than another and the final prediction of the bagged model is based on a majority vote of the individual predictions of the ensemble members. Boosting, on the other hand, uses a sequential approach i.e. each model tries to correct the mistakes made by the previous one. There are several variants of boosting: Adaptive boosting (AdaBoost), GBM and Extreme Gradient boosting (XGBoost). In AdaBoost, after evaluating the first weak learner (decision tree) with equal weights, we attach higher weights to misclassified subjects in the successive iterations until all the data points are correctly classified. In GBM, we do not put higher weights on misclassifications but choose to optimise the loss function. While XGBoost, an advanced version

of GBM, is designed to focus on computational speed and model efficiency. Gradient boosting also allows one to optimise a user specified cost function, instead of a loss function that does not essentially correspond with real world applications.

Owing to its better performance over other boosting methods, we only implement extreme gradient boosting algorithm. To implement Extreme Gradient boosting in `R`, we may use `xgboost` package (Chen et al., 2021). We prepare training and test data by creating `xgb.DMatrix()` objects with the feature and label data. The model is fitted on the training set by using `xgb.train()` function.

3.8 K-Nearest Neighbor (KNN)

The K-Nearest Neighbour (KNN; Keller et al., 1985) has a long history in statistics and was originally proposed as a local nonparametric learning algorithm. In the classification setting, the KNN algorithm essentially boils down to forming a majority vote between the K most similar instances to a given unseen observation. The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks and SVM. Despite its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications such as economic forecasting, data compression and genetics. The key parameter 'K' specifies the number of nearest neighbours used for the approximation. Practitioners often prefer to setting an odd number to avoid ties in the voting scores. When defining neighbors, KNN algorithm can adopt all kinds of distance metrics such as Euclidean distance, Hamming distance, and correlation, among others.

KNN is usually implemented in R using `knn()` function from `class` package (Venables and Ripley, 2002). This function outputs the maximum predicted probabilities only. Another package `caret` (Kuhn, 2020) in `R` provides more flexible implementation of KNN model which also gives the training and predicting setup as the other models in this tutorial.

3.9 Naive Bayes Classifier

Naive Bayes (Chen et al., 2009) is a well-known Bayesian classification method. Assume that the predictors are conditionally independent in each class, we have

$$\Pr(Y|X_1,\ldots,X_m) \propto \Pr(Y)\prod_{j=1}^{m}\Pr(X_j|Y).$$

Through the computation of individual terms in the product, we obtain the posterior probabilities as the predicted probabilities. Different versions of naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $\Pr(X_j|Y)$. This probability distribution can be Gaussian, multinomial or Bernoulli.

To implement Naive Bayes in `R`, we will use the package `e1071` (Meyer et al., 2021) in this paper. We can use `naiveBayes()` function from the package to fit the model and then obtain the predictive probabilities by the `predict()` function. One may use other `R` packages such as `naivebayes` (Majka, 2019) which allows more distribution options.

3.10 Linear Discrimination Analysis (LDA)

Linear Discriminant Analysis (LDA; Mika et al., 1999) is one of the oldest approaches for classification and could be more stable than logistic regression for normally distributed data. LDA

assumes the density functions of the two classes are multivariate normal distribution and directly uses the density ratio to form classifiers. LDA further assumes a condition called 'homoscedasticity' where the covariance matrices for the two classes are equal. If we relax this condition and allow $\mathbf{\Sigma}_0 \neq \mathbf{\Sigma}_1$, we may derive the quadratic discrimination analysis (QDA).

LDA can be implemented in `R` by using `MASS` package (Venables and Ripley, 2002). `lda()` function in the package is used to train the model and `predict()` is used for the predictions.

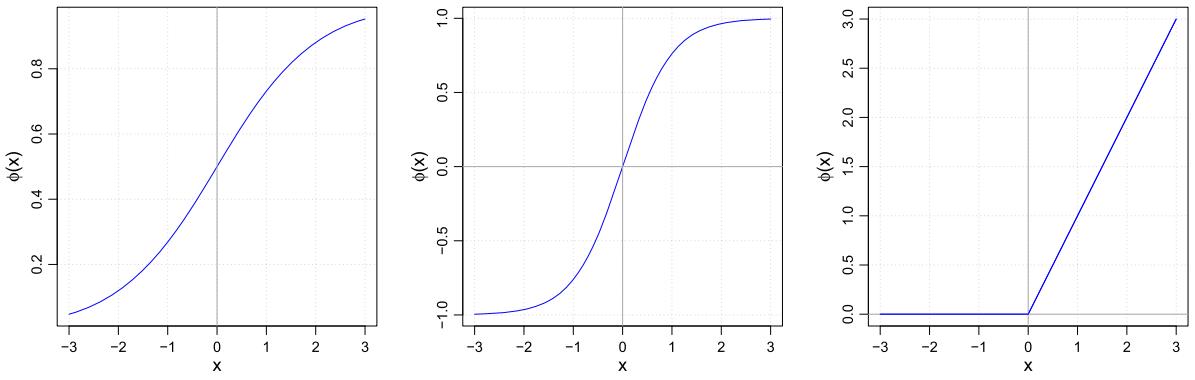
4 Deep Learning (DL)

An Artificial Neural Network (ANN; Schalkoff, 1997) is a computational model that is inspired by the way neural networks in the human brain process information. ANN have generated a lot of excitement in Machine Learning research and industry, thanks to many breakthrough results in speech recognition, text processing and computer vision. A neuron (also referred as a node) is a basic unit of computation in neural networks. It receives inputs from the other neurons or the source and produces output. Every input is assigned the weight (w) and bias (b) and then is transformed by an activation function ($\phi(.)$) which is non-linear, differentiable and monotonous. There are several activation functions used in practice. Some commonly used activation functions are:

$$\begin{aligned} \text{Sigmoid function:} \quad \phi(x) &= \frac{1}{(1+e^{-x})}. \\ \text{Tanh function:} \quad \phi(x) &= 2\,\text{Sigmoid}(2x) - 1. \\ \text{Rectified Linear Unit (ReLU) function:} \quad \phi(x) &= \max(0, x). \end{aligned}$$

The three activation functions are displayed in Figure 1. If we do not apply an activation function, the output signal would simply be a linear function. A linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data.

ANN is a feed-forward neural network in which information moves in only one direction (forward) as input → hidden layer → output and there are no cycles and loops in the network. As an representative case of ANN, multilayer perceptron (MLP; Gardner and Dorling, 1998)



(a) Sigmoid function (b) Tanh function (c) ReLU function

Figure 1: Activation functions.

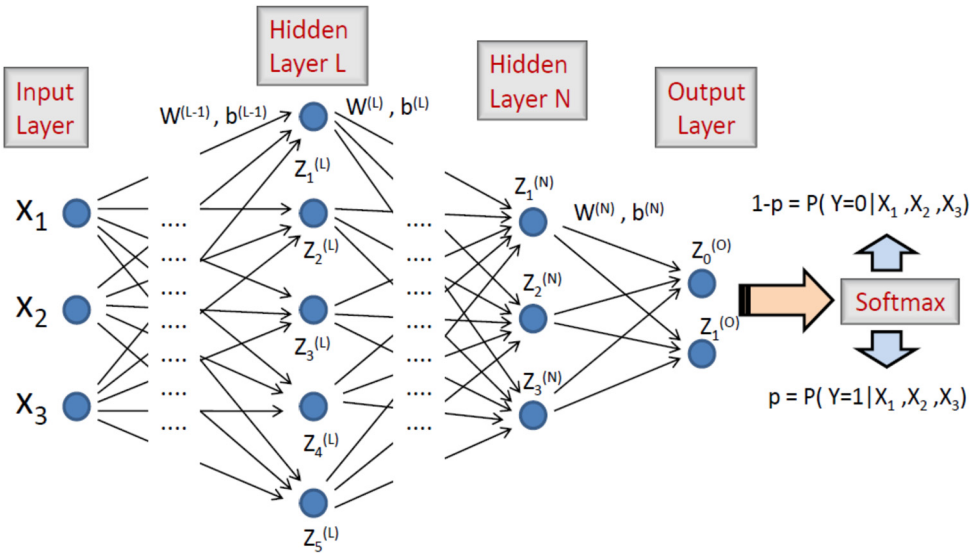


Figure 2: Neural Network Architecture.

consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Figure 2 gives a graphical illustration of MLP where we consider three hypothetical input features passing through N hidden layers.

We may consider a recursive formula to describe the DL process. In particular, the ith neuron in the L^{th} layer is given by

$$Z_i^{(L)} = \phi\left(\sum_j w_{j,i}^{(L-1)} Z_j^{(L-1)} + b_i^{(L-1)}\right),$$

where $Z_j^{(L-1)}$ is the jth neuron in the $(L-1)^{th}$ hidden layer. $w_{j,i}^{(L-1)}$ and $b_i^{(L-1)}$ are the weight and bias associated with the neuron $Z_j^{(L-1)}$ and $Z_i^{(L)}$ respectively.

In multiclass classification problems, we often use a softmax function as the activation function in the output layer to ensure that the outputs are probabilities which add up to 1. The softmax function is similar to the Sigmoid function in binary classification problem.

A cost function provides the error between the predicted outcome and the true outcome. Let $\hat{Y}_i$ be the prediction for Y_i, the outcome value for the ith subject. For binary outcome, there are different cost functions used in practice such as

$$\begin{aligned}
\text{Cross-Entropy Loss:} \quad C(Y_i, \hat{Y}_i) &= -\big[Y_i \log(\hat{Y}_i) + (1 - Y_i) \log(1 - \hat{Y}_i)\big]. \\
\text{Hinge Loss:} \quad C(Y_i, \hat{Y}_i) &= \max(0, 1 - \hat{Y}_i Y_i). \\
\text{Squared Hinge Loss:} \quad C(Y_i, \hat{Y}_i) &= \max(0, 1 - \hat{Y}_i Y_i)^2.
\end{aligned}$$

Cross-Entropy is widely used for logistic regression or GAM type of learning tasks. Hinge loss or Squared Hinge cost functions are primarily designed for SVM type of learning tasks. For Cross-Entropy loss the outcome should be coded as $\{0, 1\}$ while for hinge and squared hinge loss the outcome should be coded as $\{-1, 1\}$.

A key computation step for DL is to repeatedly optimize the cost function at the current layer to obtain the updated weight coefficients. Cost function is usually a non-convex function for

these parameters with multiple local minima and we usually have to adopt the backpropagation method to optimize it.

Any ANN is learned by an algorithm called backpropagation (backward propagation of errors) which requires the use of optimizing algorithm. Initially all the weights and biases are randomly assigned. For every input in the training dataset, the ANN may be activated to yield the observed output. This output is compared with the desired output with a cost function, and the error is propagated back to the previous layer. Optimizers calculate the gradient of the cost function with respect to all the parameters (weights and biases) to minimize the cost function. Once the cost function is minimized, we have a 'learned' neural network algorithm which, we consider is ready to work with 'new' inputs. We can optimize the cost function with one of the following optimisers:

- **Gradient Descent:** Gradient Descent is the most important technique and the foundation of how we train and optimize ANN (Bengio, 2012; Andrychowicz et al., 2016). The gradient can be easily derived using the chain rule for differentiation. Updation of weights or bias in neural network takes place as

$$\theta_t = \theta_{t-1} - \eta \bigtriangledown C(\theta_{t-1}),$$

  where θ_t is the updated weight (or bias parameter), θ_{t-1} is the previous weight (or bias parameter) while $\bigtriangledown C(\theta_{t-1})$ is a gradient of a cost function and η is a learning rate.
- **Stochastic Gradient Descent:** Stochastic Gradient Descent (SGD) on the other hand performs a parameter update for each training record (Bottou, 1991, 2010). Due to these frequent updates, it has high variance and causes the cost function to fluctuate with different intensities. This helps to discover new and possibly better local minima but sometimes it could prohibit the convergence to the exact minimum.
- **Mini batch gradient Descent:** To complement SGD and Gradient Descent, Mini Batch Gradient Descent (Khirirat et al., 2017; Ruder, 2016) is often used as it chooses the best of both techniques and performs a parameter update by dividing the entire training sample into mini-batches with the provided batch size. It reduces the variance in the parameter updates, which can ultimately lead us to a much better and stable convergence.
- **AdaGrad:** This optimiser uses a different learning rate for every parameter θ at a particular step based on the past gradients which are already computed for that parameter (Duchi et al., 2011; Dean et al., 2012). Its main weakness is that its learning rate η is always decreasing and decaying. The learning rate shrinks and eventually becomes so small, that the model just stops learning entirely and stops acquiring new additional information.
- **Adadelta:** This is an extension of AdaGrad which tends to correct the decaying learning rate problem (Zeiler, 2012; Schaul et al., 2013). Instead of accumulating all previous squared gradients, Adadelta limits the window of accumulated past gradients to some fixed size.
- **Adam:** Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter (Kingma and Ba, 2014; Reddi et al., 2019). In addition to storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients, similar to momentum. It is considered as the most efficient optimization algorithm so far.

Any optimization or learning algorithm has a number of hyperparameters to monitor the DL process. Two important hyperparameters are the batch size and number of epochs. Both are integer values. We need them when the data volume is too big and we cannot pass all the data to the computer at once. The batch size is a hyperparameter that defines the number

of samples of training data to work through before updating the internal model parameters. A training dataset can be divided into one or more batches. When all training samples are used to create one batch, the learning is usually called batch gradient descent. When the batch is the size of one sample, the learning is the stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning is the mini-batch gradient descent. The number of epochs is the number of complete passes through the training dataset. As the number of epochs increases, the weights are changed more frequently in the ANN and the learning architecture develops from underfitting, to optimal, and then to overfitting phases.

The learning rate is a configurable hyperparameter used in the training of neural networks, and has a small positive value, often in the range between 0.0 and 1.0. It controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights during each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a sub-optimal solution, whereas a learning rate that is too small can cause the process to slow down and get stuck. The challenge of training DL neural networks involves carefully selecting the learning rate or equivalently, the number of epochs.

4.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is also a type of feed forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. The inputted 2D colored image has dimension in the form of height × width × depth, where height × width are the dimensions of the 2D image while depth corresponds to the number of channels in the image, for instance, in colored 2D image, the three channels are Red (R), Green (G), Blue (B) or simply called RGB.

CNN compares the image in a piecewise manner. The pieces are usually also called features. By finding rough feature matches, CNN gives better performance than whole image matching frameworks. CNN incorporates mainly three layers: Convolution + ReLU layer, Pooling layer and Fully Connected layer. Figure 3 illustrates the computation steps involved in a typical CNN framework.

Fully connected feed-forward neural network would need large number of neurons which increases the number of parameters in computation. Convolution provides solution to this problem by reducing the number of neurons in subsequent layers. The functional element involved in carrying out convolution operation is called filter or kernel. A kernel should have the same depth as the input image and moves through the image with certain stride value to generate the dot product between the kernel and the image portion that it strides on. For example, in Figure 3, after mapping the 3×3 kernel on the highlighted patch of the figure gives the value in the upper right corner $0.69 = 1 \times 0.79 + 0 \times 0.55 - 1 \times 0.75 + 1 \times 0.30 + 0 \times 0.81 - 1 \times 0.42 + 1 \times 0.99 + 0 \times 0.56 - 1 \times 0.77$. After that we use an activation function such as ReLU to achieve the output at this layer. The convolution layer is used to extract the high level features as edges, lines, and color gradient orientation etc.

Pooling layer reduces the dimension of the convoluted feature. There are two types of pooling functions. One is MAX pooling which returns the maximum value from the slice of the convoluted feature and other is AVERAGE which returns average value of the slice of the convoluted feature. Pooling also helps in noise reduction and extracts dominant features which are rotational and positional invariant from the input image.

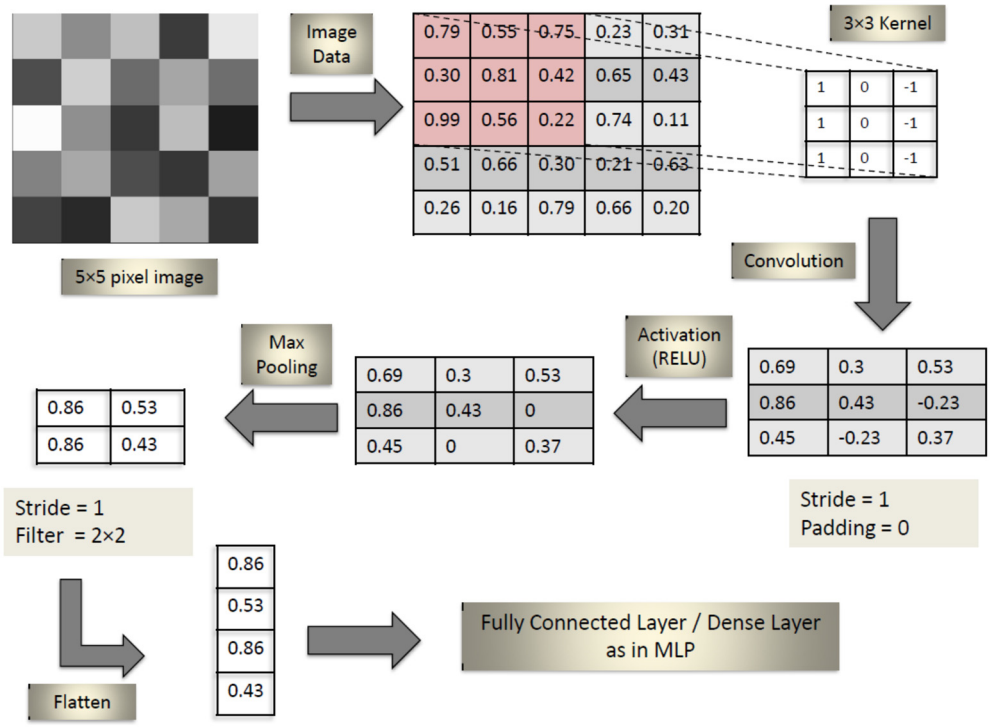


Figure 3: CNN Architecture.

Now the input image has been converted into a suitable form for MLP which is a fully connected layer for CNN, we can transform the output from the last pooling layer into a column vector (flattening) which is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax as the final output.

In `R`, we may implement CNN by using '`keras`' for `R` interface (Chollet et al., 2017). The code to implement CNN in `R`, is given in the Appendix.

5 Medical Decision Threshold

We now proceed to discussing the central question on the determination of decision threshold. After attaining the outputted predicted probability (cf. Figure 4) from any of the aforementioned classification methods, one may then choose a threshold value $c \in (0, 1)$ for medical diagnosis. If the predicted value is above c, we declare a positive condition for the subject; otherwise, the condition is declared to be negative. We may denote the outcome of the decision as a binary test result $T(c) = \mathbb{1}\{p \geqslant c\}$. At a given threshold, c, the test sensitivity $se(c)$ and the test specificity $sp(c)$ are

$$se(c) = \Pr\big(T(c) = 1|Y = 1\big),$$
$$sp(c) = \Pr\big(T(c) = 0|Y = 0\big).$$

(In the case of SVM, $se(c) = \Pr(T(c) = 1|Y^* = 1)$ and $sp(c) = \Pr(T(c) = 0|Y^* = -1)$.) We consider two types of statistical approaches, based on ROC analysis and decision curve analysis, respectively, to determine the c value in practice.

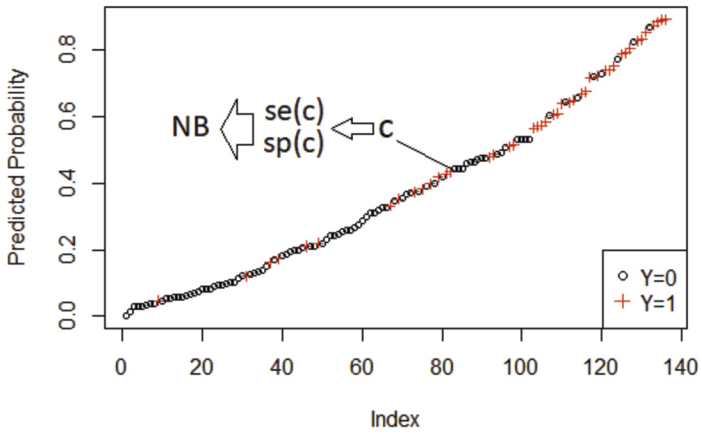


Figure 4: Sorted predicted probabilities for Pima Indian diabetes data obtained from a random forest classifier.

5.1 ROC Analysis

Receiver operating characteristic (ROC) analysis is a common tool to investigate accuracy of a prediction model (Zhou et al., 2009; Zou et al., 2012; Pepe et al., 2003; Krzanowski and Hand, 2009). An ROC curve is a plot of sensitivity ($se(c)$) vs 1-specificity ($sp(c)$) across different threshold probabilities c. Area Under ROC curve (AUC) is usually reported as an overall accuracy measure. Classification model with AUC equal to 0.5 does not have the discrimination ability while a model with AUC equal to 1 perfectly classifies all the subjects. In addition to AUC, there are many other alternative summary measures associated with ROC curve such as partial AUC, weighted AUC and sensitivity at fixed specificity (Pepe et al., 2003; Li and Zhou, 2009; Li and Fine, 2010; Yu et al., 2017; Li et al., 2019). Based on ROC curve, we can consider three criteria to determine the threshold. The following three methods may each lead to an optimal $\hat{c}$.

- **Maximizing Youden Index:** The Youden index is a simple summary measure for evaluating the classifier (Youden, 1950; Nakas et al., 2010, 2012) given by

$$J(c) = se(c) + sp(c) - 1.$$

  We denote $\hat{c}_J$ to be the optimal threshold obtained by maximizing $J(c)$. Mathematically, $\hat{c}_J$ is equivalent to the point on the ROC curve with the largest vertical distance to the diagonal from the lower left corner to the upper right corner.
- **Closest to (0,1) criterion:** Another idea is to maximize sensitivity and specificity to the extent that both are closest to 100%. Hence, the point closest to the upper left corner point $(0, 1)$ in an ROC plot corresponds to an optimal threshold (Pepe et al., 2003; Perkins and Schisterman, 2006). The Euclidean distance from the point (0,1) is given by

$$D(c) = \sqrt{\big(1 - se(c)\big)^2 + \big(1 - sp(c)\big)^2}.$$

  By minimizing $D(c)$ we obtain $(\hat{c}_D)$.
- **Equal se and sp criterion:** The third principle to determine a best threshold from an ROC curve is to find the point where sensitivity and specificity are equivalent (Sanchez, 2016). We denote this threshold by $(\hat{c}_{TF})$ obtained by solving the following equation

$$se(c) = sp(c).$$

The intersection between the ROC curve and the descending diagonal in the unit square gives $\hat{c}_{TF}$. Recall that $se(c)$ is a decreasing function of c while $sp(c)$ is an increasing function of c. This choice of threshold thus achieves a balance between true positive and false positive fractions.

5.2 Decision Curve Analysis (DCA)

Vickers and Elkin (2006) proposed decision curve analysis as a simple approach to quantify the clinical utility of a risk prediction model. For a practical decision analysis, harms and benefits need to be quantified, leading to an optimal threshold probability. It may however often be difficult to define this threshold without sufficient information (Steyerberg et al., 2010). DCA addresses the concerns between those solely interested in the accuracy and those solely interested in the utility of a prediction model (Allyn et al., 2017; Talluri and Shete, 2016). Some studies found the connection between ROC and DCA (Rousson and Zumbrunn, 2011; Baker and Kramer, 2007). The key building block of DCA is the incorporation of the utility function. We may define a net benefit function of threshold probability as follows:

$$\phi(c) = se(c)\pi - \big(1 - sp(c)\big)(1-\pi)\frac{c}{1-c},$$

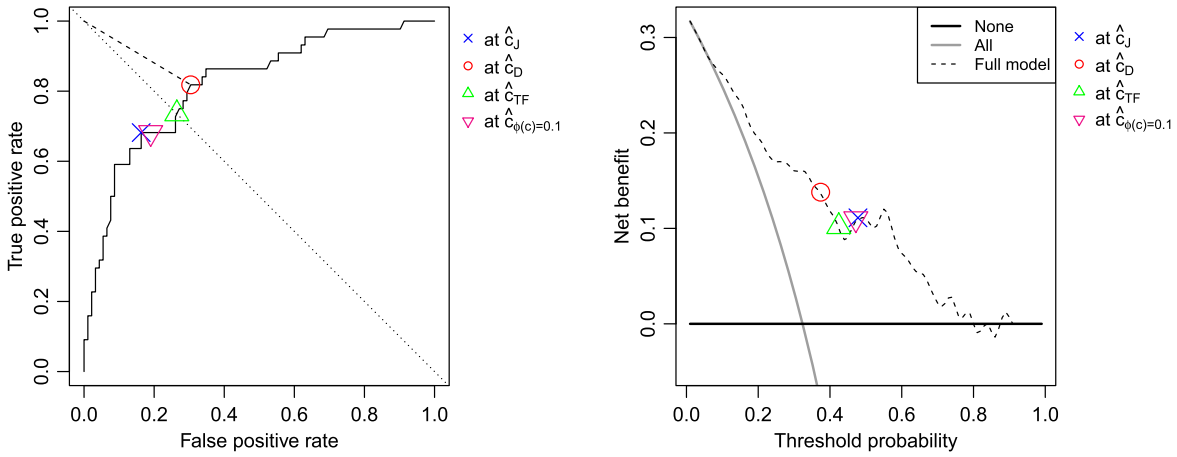
where $se(c)$ and $sp(c)$ are sensitivity and specificity at threshold probability c respectively, π denotes the disease prevalence in the population. Net benefit considers the consequences of the decisions based on a threshold. The construction of decision curve incorporates default strategies "treat none" and "treat all" in the analysis and the model is compared against such references. This gives the significance of a model over default strategies and we can decide whether to consider the model for analysis or not. The net benefit for "treat none" is zero while that for "treat all" strategy ($\phi_1(c)$) is

$$\phi_1(c) = \pi - (1-\pi)\frac{c}{1-c}.$$

DCA is decision analytic method and incorporates the physician or patient's preferences (Fitzgerald et al., 2015; Van Calster et al., 2013; Sande et al., 2020). We usually inspect a decision curve and look for which strategy leads to the greatest net benefit (i.e., the curve above others). In `R`, we can draw decision curve using the package `dca` (Sjoberg, 2021) after obtaining predicted probabilities from the classifier. We note that this package is not available at the `CRAN` site yet and has to be downloaded from `Github` using the `remotes` package (Hester et al., 2021). Note that the updated package is `dcurves` and the new function `dca()` works differently.

Based on the plotted decision curve, one can decide an appropriate threshold corresponding to a desired net benefit value. For example, if we intend to achieve $\phi(c) = 0.1$, the corresponding $\hat{c}_{\phi(c)=.1}$ is found to be 0.472 using the PIMA data (cf. Figure 5(b)). In practice the risk threshold c can be selected by the policy makers and then used to compute the net benefit at that threshold assuming that the risk threshold accurately summarizes the costs and benefits of intervention (Kerr et al., 2016; Van Calster et al., 2018).

We illustrate all decision thresholds in an ROC curve in Figure 5(a) where $\hat{c}_J = 0.478$, $\hat{c}_D = 0.374$, $\hat{c}_{TF} = 0.424$ and $\hat{c}_{\phi(c)=.1} = 0.472$. We can see that thresholds obtained by different methods are quite close to each other with similar sensitivity values but slightly different specificity values in this example. We also mark all four thresholds in the same decision curve in Figure 5(b). The net benefit values for all the thresholds are close to 10% in this case. The threshold $\hat{c}_D$ appears to yield a larger net benefit 0.138 than other thresholds.



(a) ROC curve (b) Decision curve

Figure 5: Different decision thresholds plotted in ROC and Decision curves by random forest model for Pima Indian Diabetes Dataset.

6 Case Studies

6.1 Pima Indian Diabetes Dataset

This dataset (Smith et al., 1988) is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. All patients are females at least 21 years old of Pima Indian heritage. The purpose of this study is to diagnostically predict whether or not a patient has diabetes, based on her demographic and clinical measurements. This is a public dataset available in the Kaggle datasets repository (https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv). The data include diagnosis results of 768 women with 9 variables summarized in Table 1.

Table 1: Information of Variables in the Pima Indian Diabetes dataset (more information for the calculation of Diabetes Pedigree Function can be found in Smith et al., 1988).

Variable	Summary
	Mean ± SD
Pregnancies	3.84 ± 3.36
Diabetes Pedigree Function	0.47 ± 0.33
Age (*years*)	33.24 ± 11.76
Glucose (*mg/dl*)	121.51 ± 30.55
Diastolic Blood Pressure (*mmHg*)	72.43 ± 12.30
Triceps skin fold thickness (*mm*)	28.67 ± 10.34
Insulin (*mu U/ml*)	149.68 ± 108.56
BMI (kg/m^2)	32.44 ± 6.91
	Proportion
Diabetes	Yes: 34.9%

Table 2: Training and test set (with 7:3 proportion) AUC for all the classification methods along with 95% Confidence Interval.

Methods	Training (70%)		Test (30%)	
	AUC	Confidence Interval	AUC	Confidence Interval
GLM-Logit	0.8472	(0.8145, 0.8798)	0.8430	(0.7920, 0.8941)
GLMNET-LASSO	0.8453	(0.8126, 0.8781)	0.8425	(0.7914, 0.8935)
GAM	0.8453	(0.8122, 0.8784)	0.8430	(0.7917, 0.8944)
LDA	0.8468	(0.8141, 0.8795)	0.8431	(0.7920, 0.8942)
Naive Bayes	0.8209	(0.7858, 0.8560)	0.8210	(0.7670, 0.8750)
XGBoost	1.0000	(1.0000, 1.0000)	0.8180	(0.7622, 0.8738)
Decision Tree	0.8679	(0.8353, 0.9005)	0.7767	(0.7138, 0.8397)
SVM	0.7084	(0.6815, 0.7353)	0.6701	(0.6153, 0.7249)
RF	1.0000	(1.0000, 1.0000)	0.8391	(0.7872, 0.8910)
KNN	0.8617	(0.8310, 0.8924)	0.8250	(0.7709, 0.8791)
MLP	0.8262	(0.7914, 0.8611)	0.8195	(0.7648, 0.8742)

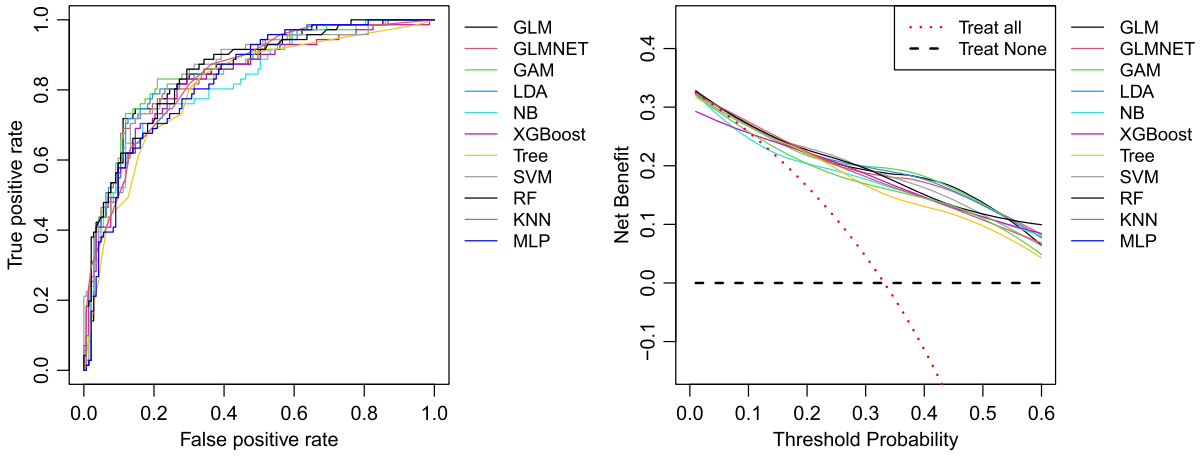
The full data was divided into training and test data with the ratio of 7:3. We trained all the classification models described previously on the training set and then obtain the predicted probabilities for the training and test sets separately. The classification was repeated for different randomly partitioned training and test sets for 100 times and then average AUC values for training and test sets were calculated. We reported the results in Table 2. For this data set XGBoost and random forest (RF) achieves 100% accuracy to predict the diabetes status for the training sample. However for the test set XGBoost is not as good as LDA.

We plotted ROC and decision curves for the test data as shown in Figures 6(a) and 6(b) respectively for a single sample set for the purpose of illustration. Specifically, the ROC plot is obtained using the `ROCR` package (Sing et al., 2005). Next, we produced decision curves for all the classification methods using the `dca` package (Sjoberg, 2021). In general all methods yield similar but distinct ROC curves and decision curves. One may select a sensible threshold based on the methods introduced in the preceding section for all these methods.

We note that decision curves offer more valuable information on utility on top of the usual accuracy assessment provided in ROC curves. If the risk threshold preferred by physician or patient lies within a range where a certain classifier has higher net benefit compared to others, then the treatment decision should be based on the predicted risks from this classifier.

6.2 Malaria Parasite Detection in Thin-blood Smear Images

While the Pima Indian diabetes data allow shallow learning, we next focus on a case study with deep learning. Malaria is a mosquito-borne infectious disease caused by the Plasmodium parasites transmitted through the bite of female Anopheles mosquito. Physicians commonly examine thick and thin blood smears to diagnose disease and compute parasitemia. Their accuracy depends on blood smear quality and expertise in classifying and counting parasitized and uninfected cells. Such examinations could be arduous for large-scale diagnoses resulting in poor quality. Convolutional Neural Networks (CNN) promise highly scalable and superior results with end-to-end feature extraction and classification. Automating malaria screening using deep learning



(a) ROC curves (b) Decision curves

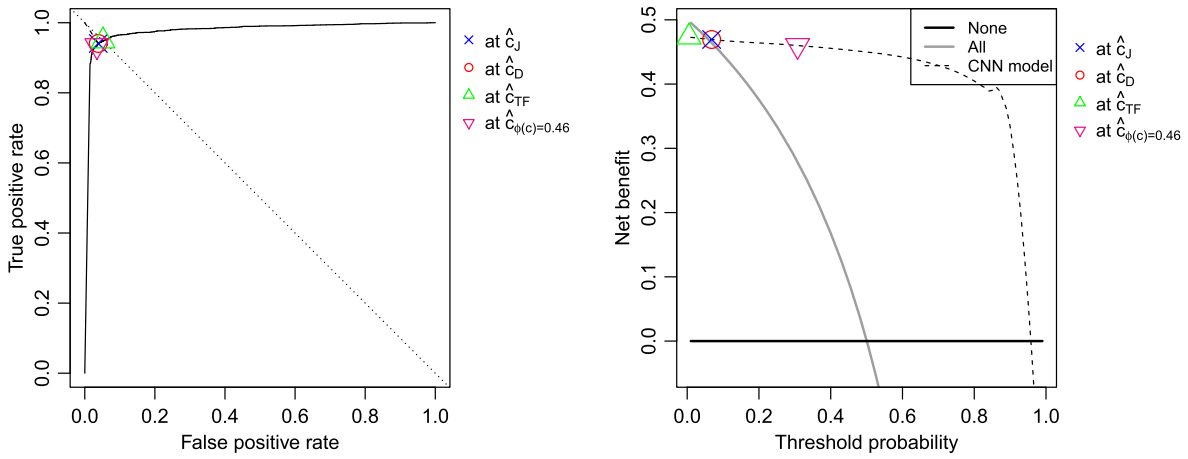
Figure 6: Test Set ROC and Decision curves for Pima Indian Diabetes Dataset.

(a) Parasitized (b) Uninfected

Figure 7: Sample thin blood smear slide images of parasitized and uninfected cells.

methods could serve as an effective diagnostic aid. The segmented cells from the thin blood smear slide images for the parasitized and uninfected classes are open-source and made available at (https://ceb.nlm.nih.gov/repositories/malaria-datasets/). The data contains a total of 27,558 cell images with equal instances of parasitized and uninfected cells. Sample images of parasitized ($Y = 1$) and uninfected ($Y = 0$) cells are shown in Figure 7.

In this case shallow learning methods are no longer applicable. We used `Keras` for `R` interface (Chollet et al., 2017) with Tensorflow backend to apply CNN (Please refer Supplementary Material for the `R` Code). The images were available in random pixel sizes between $100 \sim 150$ pixels in height and width. We first processed the image data by resizing (100×100 pixels) and organizing the image data as required for CNN model in `Keras` for which we used the `R` package `EBImage` (Pau et al., 2010). Then the dataset was divided into training and test sets randomly with the ratio 8:2. The training data contains 22056 thin blood smear slide images while test data contains 5502 images after a random splitting. We then fitted CNN model to the training data (CNN model we used contains 2 sets of two convolution-pooling layers plus dropout layer and then the fully connected layer) and then predicted the disease outcome on test data.



(a) ROC curve (AUC=0.977)

(b) Decision curve

Figure 8: ROC and DCA curves for the CNN model on Malaria thin blood smear image dataset.

Performance of the CNN model was evaluated for classifying parasitized and uninfected cells with ROC and DCA. The ROC curve and decision curve are presented in Figures 8(a) and 8(b) respectively. The overall AUC is 0.977, suggesting a very accurate classification results for the test data.

To make a dichotomous classification for individual subjects, we may apply the methods introduced in Section 5 to select an appropriate decision threshold. In this case, we have $\hat{c}_J = 0.068$, $\hat{c}_D = 0.068$, $\hat{c}_{TF} = 0.005$ based on the ROC curve (Figure 8(a)). We can also choose a threshold based on the decision curve analysis. In fact, eyeballing Figure 8(b), we note that the net benefit function remains relatively flat around the wide range of threshold values, essentially suggesting that all these threshold values lead to a net benefit value between 0.4 and 0.5. In particular, the threshold corresponding to $\phi(c) = 0.46$ is at 0.307 for which the sensitivity and specificity are 0.93 and 0.96. The three cut-off values identified from the ROC curve all have similar benefit values 0.47.

7 Discussion

Machine learning and deep learning has created lot of buzz in science. To appreciate such advanced algorithms, basic understanding of machine learning and neural networks is necessary. In this paper we provided a brief review of the machine learning methods for clinical predictive analytics. It is important to evaluate these different methods in practical conditions and acknowledge the limitations of the currently available methods and research topics that are needed by the field. In particular, GLM and regularized GLM depend on linearity assumption and may be more restrictive than other nonparametric classifiers; decision tree is usually quite a weak classifier and needs to be coupled with bagging and/or boosting to achieve a satisfactory performance; LDA and Bayes methods rely on the distribution assumptions such as normality and may not be valid when the distribution is mis-specified; SVM and deep learning, though enjoying superior performance most of the time, may require the specification of a lot of hyper parameters. Furthermore, there is no theoretical guarantee that one method is consistently more predictive than other methods. We need to be mindful in choosing appropriate methods for the problems.

Sometimes using a single algorithm or method can mislead the decision. It is often more effective to consider multiple models and assess the prediction jointly. All the methods reviewed in our paper can serve as potential candidates for addressing the prediction issues in medicine and other scientific fields. Through this study, it is also helpful to note that a complex model is not always better. A simple shallow learning model like logistic regression and LDA can also perform quite well with high AUC values when their assumptions are met in an application. On the other hand, when dealing with complicated data sets such as the brain images, almost all shallow learning methods could fail and we have to invoke a well-designed deep learning computation to achieve reasonable risk predictions. Arranging accurate and reliable learning procedures for a real data will lead to more sensible decision threshold selection, using the methods we reviewed in this paper.

Supplementary Materials

Supplementary material online include: The review of different smoothers used in Generalized additive models, Installation details for `R` interface for Keras and Tensorflow, data and R code needed to reproduce the results.

Acknowledgments

We are grateful to the Editorial Board and two reviewers for their constructive comments.

Funding

The work was partly supported by Academic Research Funds R-155-000-205-114, R-155-000-195-114 and Tier 2 MOE funds in Singapore MOE2017-T2-2-082: R-155-000-197-112 (Direct cost) and R-155-000-197-113 (IRC).

References

Allyn J, Allou N, Augustin P, Philip I, Martinet O, Belghiti M, et al. (2017). A comparison of a machine learning model with EuroSCORE II in predicting mortality after elective cardiac surgery: A decision curve analysis. *PLoS ONE*, 12(1): e0169772.

Alvarez I, Bernard S, Deffuant G (2007). Keep the decision tree and estimate the class probabilities using its decision boundary. In: *IJCAI*, 654–659.

Andrychowicz M, Denil M, Colmenarejo SG, Hoffman MW, Pfau D, Schaul T, et al. (2016). Learning to learn by gradient descent by gradient descent. *CoRR*, arXiv preprint: https://arxiv.org/abs/1606.04474.

Baker SG, Kramer BS (2007). Peirce, Youden, and receiver operating characteristic curves. *American Statistician*, 61(4): 343–346.

Bengio Y (2012). Practical recommendations for gradient-based training of deep architectures. In: *Neural Networks: Tricks of the Trade* (G Montavon, G Orr, KR Müller, eds.), 437–478. Springer.

Bottou L (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8): 12.

Bottou L (2010). Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010* (Y Lechevallier, G Saporta, eds.), 177–186. Springer.
Breiman L (1996). Bagging predictors. *Machine Learning*, 24(2): 123–140.
Breiman L (2001). Random forests. *Machine Learning*, 45(1): 5–32.
Breiman L (2017). *Classification and Regression Trees*. Routledge.
Breiman L, Friedman J, Stone C, Olshen R (1984). *Classification and Regression Trees. The Wadsworth and Brooks-Cole Statistics-Probability Series*. Taylor & Francis.
Chang CC, Lin CJ (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3): 1–27.
Chen J, Huang H, Tian S, Qu Y (2009). Feature selection for text classification with naïve Bayes. *Expert Systems with Applications*, 36(3): 5432–5435.
Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, et al. (2021). *xgboost: Extreme Gradient Boosting*. R package version 1.3.2.1.
Chollet F, Allaire J, et al. (2017). R interface to keras, https://github.com/rstudio/keras.
Cortes C, Vapnik V (1995). Support-vector networks. *Machine Learning*, 20(3): 273–297.
Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, et al. (2012). Large scale distributed deep networks. In: *Advances in Neural Information Processing Systems* (F Pereira, CJC Burges, L Bottou, KQ Weinberger, eds.), volume 25. Curran Associates, Inc.
Duchi J, Hazan E, Singer Y (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7): 2121–2159.
Erkanli A, Sung M, Costello E, Angold A (2006). Bayesian semi-parametric ROC analysis. *Statistics in Medicine*, 25: 3905–3928.
Fitzgerald M, Saville BR, Lewis RJ (2015). Decision curve analysis. *JAMA*, 313(4): 409–410.
Friedman J, Hastie T, Tibshirani R (2001). *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics. Springer, New York.
Friedman J, Hastie T, Tibshirani R (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1): 1.
Friedman JH (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5): 1189–1232.
Gardner MW, Dorling S (1998). Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14–15): 2627–2636.
Hastie T (2020a). *gam: Generalized Additive Models*. R package version 1.20.
Hastie T (2020b). *svmpath: The SVM Path Algorithm*. R package version 0.970.
Hastie T, Tibshirani R (1986). Generalized additive models. *Statistical Science*, 1(3): 297–310.
Hastie TJ (2017). Generalized additive models. In: *Statistical Models in S* (JM Chambers, TJ Hastie, eds.), 249–307. Routledge.
Hester J, Csárdi G, Wickham H, Chang W, Morgan M, Tenenbaum D (2021). *remotes: R Package Installation from Remote Repositories, Including 'GitHub'*. R package version 2.3.0.
Hothorn T, Hornik K, Zeileis A (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3): 651–674.
Hsu CW, Lin CJ (2002). A simple decomposition method for support vector machines. *Machine Learning*, 46(1): 291–314.
Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). kernlab – An S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9): 1–20.
Kass GV (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 29(2): 119–127.

Keller JM, Gray MR, Givens JA (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(4): 580–585.

Kerr KF, Brown MD, Zhu K, Janes H (2016). Assessing the clinical impact of risk prediction models with decision curves: Guidance for correct interpretation and appropriate use. *Journal of Clinical Oncology*, 34(21): 2534.

Khirirat S, Feyzmahdavian HR, Johansson M (2017). Mini-batch gradient descent: Faster convergence under data sparsity. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2880–2887.

Kingma DP, Ba J (2014). Adam: A method for stochastic optimization. *CoRR*, arXiv preprint: https://arxiv.org/abs/1412.6980.

Krzanowski WJ, Hand DJ (2009). *ROC Curves for Continuous Data*. Chapman and Hall/CRC.

Kuhn M (2020). *caret: Classification and Regression Training*. R package version 6.0-86.

Li J, Fine JP (2010). Weighted area under the receiver operating characteristic curve and its application to gene selection. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 59: 673–692.

Li J, Gao M, D'Agostino R (2019). Evaluating classification accuracy for modern learning approaches. *Statistics in Medicine*, 38: 2477–2503.

Li J, Zhou XH (2009). Nonparametric and semiparametric estimation of the three way receiver operating characteristic surface. *Journal of Statistical Planning and Inference*, 139: 4133–4142.

Liaw A, Wiener M (2002). Classification and regression by randomforest. *R News*, 2(3): 18–22.

Majka M (2019). *naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R*. R package version 0.9.7.

Mehta M, Rissanen J, Agrawal R (1995). Mdl-based decision tree pruning. In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, KDD'95* (U Fayyad, R Uthurusamy, eds.), 216–221. AAAI Press.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2019). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-2.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2021). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-6.

Mika S, Ratsch G, Weston J, Scholkopf B, Mullers K (1999). Fisher discriminant analysis with kernels. In: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)* (YH Hu, ed.), 41–48.

Molinaro AM, Lostritto K, van der Laan M (2010). partdsa: deletion/substitution/addition algorithm for partitioning the covariate space in prediction. *Bioinformatics*, 26(10): 1357–1363.

Nakas CT, Alonzo TA, Yiannoutsos CT (2010). Accuracy and cut-off point selection in three-class classification problems using a generalization of the Youden index. *Statistics in Medicine*, 29: 2946–2955.

Nakas CT, Dalrymple-Alford JC, Anderson TJ, Alonzo TA (2012). Generalization of Youden index for multiple-class classification problems applied to the assessment of externally validated cognition in Parkinson disease screening. *Statistics in Medicine*, 95: 995–1003.

Nelder JA, Wedderburn RW (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A. General*, 135(3): 370–384.

Niblett T, Bratko I (1987). Learning decision rules in noisy domains. In: *Proceedings of Expert Systems '86, The 6th Annual Technical Conference on Research and Development in Expert*

Systems III (MA Bramer, ed.), 25–34. Cambridge University Press, USA.

O'Malley A, Zou K (2006). Bayesian multivariate hierarchical transformation models for ROC analysis. *Statistics in Medicine*, 25: 459–479.

Pau G, Fuchs F, Sklyar O, Boutros M, Huber W (2010). Ebimage—an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7): 979–981.

Pepe MS, et al. (2003). *The Statistical Evaluation of Medical Tests for Classification and Prediction*. Medicine.

Perkins Neil J, Schisterman Enrique F (2006). The inconsistency of "optimal" cut-points using two roc based criteria. *American Journal of Epidemiology*, 163: 670–675.

Quinlan JR (1993). *C4.5: Programming for Machine Learning*. Morgan Kauffmann, 38: 48.

Reddi SJ, Kale S, Kumar S (2019). On the convergence of adam and beyond. *CoRR*, arXiv preprint: https://arxiv.org/abs/1904.09237.

Roberts DW (2020). *optpart: Optimal Partitioning of Similarity Relations*. R package version 3.0-3.

Rousson V, Zumbrunn T (2011). Decision curve analysis revisited: overall net benefit, relationships to ROC curve analysis, and application to case-control studies. *BMC Medical Informatics and Decision Making*, 11(1): 1–9.

Ruder S (2016). An overview of gradient descent optimization algorithms. *CoRR*, arXiv preprint: https://arxiv.org/abs/1609.04747.

Safavian SR, Landgrebe D (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3): 660–674.

Salzberg SL (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 16(3): 235–240.

Sanchez IE (2016). Optimal threshold estimation for binary classifiers using game theory. *F1000Research*, 5.

Sande SZ, Li J, D'Agostino R, Yin Wong T, Cheng CY (2020). Statistical inference for decision curve analysis, with applications to cataract diagnosis. *Statistics in Medicine*, 39(22): 2980–3002.

Schalkoff RJ (1997). *Artificial Neural Networks*. McGraw-Hill Higher Education.

Schapire RE, Freund Y, Bartlett P, Lee WS, et al. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651–1686.

Schaul T, Zhang S, LeCun Y (2013). No more pesky learning rates. *Proceedings of Machine Learning Research* 28(3): 343–351.

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). Rocr: visualizing classifier performance in R. *Bioinformatics*, 21(20): 7881.

Sjoberg DD (2021). *dca: Decision Curve Analysis*. R package version 0.1.0.9000.

Smith JW, Everhart J, Dickson W, Knowler W, Johannes R (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: *Proceedings of the Annual Symposium on Computer Application in Medical Care*, volume 261. American Medical Informatics Association.

Steyerberg EW, Vickers AJ, Cook NR, Gerds T, Gonen M, Obuchowski N, et al. (2010). Assessing the performance of prediction models: a framework for some traditional and novel measures. *Epidemiology*, 21(1): 128.

Talluri R, Shete S (2016). Using the weighted area under the net benefit curve for decision curve analysis. *BMC Medical Informatics and Decision Making*, 16(1): 94.

Therneau T, Atkinson B (2019). *rpart: Recursive Partitioning and Regression Trees*. R package

version 4.1-15.

Van Calster B, Vickers AJ, Pencina MJ, Baker SG, Timmerman D, Steyerberg EW (2013). Evaluation of markers and risk prediction models: Overview of relationships between NRI and decision-analytic measures. *Medical Decision Making*, 33(4): 490–501.

Van Calster B, Wynants L, Verbeek JF, Verbakel JY, Christodoulou E, Vickers AJ, et al. (2018). Reporting and interpreting decision curve analysis: A guide for investigators. *European Urology*, 74(6): 796–804.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. 0-387-95457-0.

Vickers AJ, Cronin AM, Gönen M (2012). A simple decision analytic solution to the comparison of two binary diagnostic tests. *Statistics in Medicine*, 32(11): 1865–1876.

Vickers AJ, Elkin EB (2006). Decision curve analysis: A novel method for evaluating prediction models. *Medical Decision Making*, 26(6): 565–574. PMID: 17099194.

Vickers AJ, van Calster B, Steyerberg EW (2019). A simple, step-by-step guide to interpreting decision curve analysis. *Diagnostic and Prognostic Research*, 3(1): 1–8.

Weihs C, Ligges U, Luebke K, Raabe N (2005). klaR analyzing German business cycles. In: *Data Analysis and Decision Support* (D Baier, R Decker, L Schmidt-Thieme, eds.), 335–343. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wood SN (2003). Thin-plate regression splines. *Journal of the Royal Statistical Society, Series B*, 65(1): 95–114.

Youden WJ (1950). Index for rating diagnostic tests. *Cancer*, 3(1): 32–35.

Yu T, Li J, Ma S (2017). Accounting for clinical covariates and interactions in ranking genomic markers using ROC. *Communications in Statistics. Simulation and Computation*, 46(5): 3735–3755.

Zeiler MD (2012). ADADELTA: An adaptive learning rate method. *CoRR*, arXiv preprint: https://arxiv.org/abs/1212.5701.

Zhang Z, Rousson V, Lee WC, Ferdynus C, Chen M, Qian X, et al. (2018). Decision curve analysis: a technical note. *Annals of Translational Medicine*, 6(15): 308.

Zhou XH, McClish DK, Obuchowski NA (2009). *Statistical Methods in Diagnostic Medicine*, volume 569. John Wiley & Sons.

Zou K, Liu A, Bandos A, Ohno-Machado L, Rockette H (2012). *Statistical Evaluation of Diagnostic Performance: Topics in ROC Analysis*. CRC Press.