https://doi.org/10.48048/tis.2023.6881

Comparative Evaluation of Machine Learning Algorithms for Alzheimer's Disease Classification using Synthetic Transcriptomics Dataset

Mohammad Nasir Abdullah[1,*], Yap Bee Wah[2,3], Abu Bakar Abdul Majeed[4], Yuslina Zakaria[5], Norshahida Shaadan[3], Mohamed Imran Mohamed Ariff[6], Jufiza A. Wahab[6], Samsiah Ahmad[6], Mohd Sapuan Baharuddin[6] and Wan Noor Hayatie Wan Abdul Aziz[6]

[1]Mathematical Sciences Studies, College of Computing, Informatics and Media, Universiti Teknologi MARA, Perak Branch, Tapah Campus, 35400 Tapah, Perak, Malaysia
[2]UNITAR Graduate School, UNITAR International University, 47301 Petaling Jaya, Selangor, Malaysia
[3]College of Computing, Informatics and Media, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia
[4]Brain Research Laboratory, Faculty of Pharmacy, Universiti Teknologi MARA, Cawangan Selangor, Kampus Puncak Alam, 42300 Puncak Alam, Selangor, Malaysia
[5]Faculty of Pharmacy, Universiti Teknologi MARA, Cawangan Selangor, Kampus Puncak Alam, 42300 Puncak Alam, Selangor, Malaysia
[6]College of Computing, Informatics and Media, Universiti Teknologi MARA, Perak Branch, Tapah Campus, 35400 Tapah, Perak, Malaysia

([*]Corresponding author's e-mail: nasir916@uitm.edu.my)

Received: 20 March 2023, Revised: 26 April 2023, Accepted: 8 May 2023, Published: 28 August 2023

Abstract

Recent technological advancements have enabled the understanding of multi-omics data, including transcriptomics, proteomics, and metabolomics. Machine learning algorithms have shown promising results in classifying multi-omics data. The objective of this paper is to evaluate the performance of machine learning algorithms in classifying transcriptomics data for Alzheimer's disease (AD) patients and healthy control (HC) individuals. A Synthetic dataset of varying sample sizes, dimensionalities, effect sizes, and correlations was generated based on actual transcriptomics data for AD patients. The dataset consisted of 22,254 markers for 92 AD patients and 92 HC individuals. Four machine learning classifiers: naïve Bayes (NB), k-nearest neighbour (k-NN), support vector machine (SVM), and random forest (RF), were used to classify the data. The simulation was conducted using a parallel processing approach on a high-performance machine. Based on the error rate and F-measure, NB outperformed k-NN, SVM, and RF for high-dimensional data. However, SVM with a radial basis kernel (RBF) kernel performed better than NB only when the sample size was greater than 100 per group for all dimensions. The result suggests that machine learning algorithms, specifically NB, can effectively classify transcriptomics data for AD patients. SVM with an RBF kernel is a better option for large sample sizes. This study provides valuable insights for future research in the classification of transcriptomics data using machine learning algorithms.

Keywords: Alzheimer's disease, High-dimensional data, Machine learning, Multi-omics, Simulations, Transcriptomics

Introduction

High-throughput screening (HTS) is a method for scientific experimentation primarily used in drug discovery. HTS allows the collection of data from millions of chemical, genetic, or pharmacological tests [1-3]. Recent technological innovations have revolutionised the understanding of omics data, such as transcriptomics, proteomics, metabolomics, and other high-throughput technologies output (multi-omics). These data are high-dimensional and usually involve a few samples but with many features or variables [4]. Multi-omics studies specifically related to classification have been extensively explored recently [5-8]. The new RNA-Seq has replaced classical microarray as the technology for quantifying gene expression since it can provide less noisy data and a definite point for detecting novel biomarkers. It also does not need to pre-arrange the transcripts of interest [9].

The breakthrough of this technology allows scientists to monitor a large amount of data to identify potential biomarkers for a disease, understand protein functions and their interactions, as well as elucidate biochemical pathways [10]. These quantitative biological problems, which are high dimensional problems, have necessitated new statistical and computational interventions. In other words, high-dimensional data require the development of novel methods, innovation or improvement of existing machine learning (ML) algorithms or statistical models to optimise the knowledge output from such data [11]. ML algorithms have shown promising results for classifying multi-omics data [12]. Since this type of data typically consists of many features with a small sample size, the selection of ML classifiers needs to be acclimated to solving the p more than n (features greatly outnumber the sample size) and multicollinearity issues. Many classifiers have been reported to cope with these issues, but the 'there is no free lunch' axiom of optimisation avows that there is no ideal algorithm that can usurp all cases [12].

There are many different types of ML algorithms that have been developed, namely supervised, unsupervised, or semi-supervised learning. Every ML algorithm is based on distinctive mathematical approaches, and it is expected that, with variation in the types of samples, data, and area of application, one algorithm can significantly outperform others in terms of prediction accuracy [13]. With the increasing number of ML algorithms available, studies have been conducted to assess their performance in a particular situation.

Khondoker *et al.* [11] have highlighted several issues related to comparing ML studies, emphasising the inclusion of elements of unbiasedness and neutrality. Unbiasedness means the researcher should not pick a favourite online dataset to prove that some classifiers are better than others. In contrast, neutrality implies the use of real datasets to demonstrate the performance of ML algorithms. The problem with the actual dataset in comparing ML algorithms is the noise (sampling error), which will affect the performance measures of the classifier. Furthermore, most high-dimensional datasets, especially multi-omics datasets, will contain noise error problems [14].

Generating a synthetic dataset is a common exercise when the real dataset is difficult to obtain due to budgeting, time, and privacy constraints. Normally, a synthetic dataset is generated to test the algorithms and newly developed algorithms, and it also allows easy verification of the algorithm's efficiency in finding all distinct constructions. Numerous reasons lead to generating synthetic datasets: firstly, it provides a controlled testing environment for any algorithm and situation under consideration. The dataset is also useful for testing the scalability and robustness of new algorithms. It is safe to share openly with the public without worrying about the privacy and confidentiality aspects of real data [15].

Furthermore, the generation of a synthetic dataset would significantly overcome the replicability issues. It creates a completely new dataset that imitates the real dataset and protects its statistical characteristics and association between the features [16].

Thus, it is essential to make sure the synthetic dataset has the characteristics of an estimator at varying levels, such as variability, sample size, effect size, and correlation with the original dataset. There are 2 categories for assessing the utility of a synthetic dataset. Firstly, the general utility mimics the statistical characteristics and multivariable association between the synthetic and actual datasets. Secondly, the specific utility is used to fit the synthetic model, which can be accessed by computing the lack of fit against a similar model in the real dataset [16]. **Table 1** shows the list of related work on synthetic datasets for performance comparison with ML classifiers.

Table 1 Related work on generating synthetic dataset from a real dataset.

Authors	Characteristics of synthetic dataset
[11]	Consideration of the real dataset, with imposing biological variability, experimental variability, correlation structures, and effect size.
[16]	Replacing some or all the data by sampling from the probability distribution is similar to multiple imputations.
[15]	Generating data using visual characteristics to model the visual trends of an actual dataset. The edge length, graph length, and graph diameter were used to create the dataset.
[17]	Randomly choose the variable from the original dataset and change the value randomly with consideration of the probability distribution.
[18]	Replicating the SEIR model and imposing the binomial probability distribution on the synthetic dataset.

This study is intended to fill in the gap by developing a synthetic dataset that can imitate the actual dataset. Most of the previous research did not consider the element of correlation between independent variables. For the real multi-omics data, there are a lot of markers that are correlated with each other. Thus, the synthetic dataset should follow the distribution of the real dataset and the correlation effects between the markers.

Three schools of thought can be followed to compare the performance of ML algorithms. First, by analytically comparing the distribution theory with the performance of the estimator. Second, by using a statistical test to consider the sampling variability of the estimates (or estimators), Third, by repeatedly estimating the performance measure based on the simulated data and averaging the sampling variability from the estimated threshold.

Based on Hanczar and Dougherty [19], synthetic data permits the computation of true error for comparison of classification performance, whereas direct comparison or experimentation using actual datasets is usually inconclusive as the characteristics of datasets differ. Comparison of ML algorithms' performance using synthetic data via simulation is strongly suggested, as some parameters can be controlled and the pattern of the classifiers' performance can be evaluated [19].

Thus, this study investigates the performance of ML algorithms using synthetic data (simulated data) under the conditions of different sample sizes, the number of features (dimensionalities), effect size, and correlation. An extensive simulation procedure was employed to compare the performance of 4 ML algorithms, which are naïve Bayes (NB), *k*-Nearest Neighbour (*k*-NN), support vector machine (SVM), and random forest (RF). The simulation was carried out using a parallel processing approach on a high-performance machine. We evaluated the ML algorithms by considering several features (p), sample size (n), and effect size, which were correlation coefficient (r), biological variation (σ_b) and experimental variation (σ_e).

The novelty of this paper is as follows: 1) Development of a synthetic dataset that can imitate the real dataset for 2 groups and allow the adjustment of correlations and variations of the data based on the original dataset. 2) The foundation of the ML classifiers that work well on the high and low dimensional states of the data. This paper is organised as follows: Section 2 describes classification using ML algorithms for high-dimensional data, while the methodology is explained in detail in Section 3. The results and discussion of the findings are presented in Section 4, and Section 5 concludes the paper.

Machine learning for performance comparisons

Classification methods

Classification methods can be divided into 2 types: Probabilistic and non-probabilistic outputs. In the probabilistic output, we can compute the probability of the output variable [20]. Naïve Bayes (NB), support vector machine (SVM), and random forest (RF) produce the probability of the output variable, while *k*-nearest neighbour (*k*-NN) only determines the class label of the output variable. The 4 classification algorithms are described in detail in the following sections.

Naïve Bayes

Naïve Bayes (NB) classifier is widely used in many areas, such as text mining, pattern recognition, accounting, and biomedical research. It is in the family of simple probabilistic classifiers that depend on Bayes' theorem. The required assumption is that all variables are independent of each other, given a categorical output variable. It is remarkably robust in practical applications, and the performance is 'non-disappointing' [21,22].

NB classifier learns from the conditional probability of each feature X_i given that class label C ($\mathrm{P}(X_i|C)$) from the training data. The classification is performed by applying Bayes' rule to compute the probability of C given the specific features of $x_1, x_2, x_3, \ldots, x_d$. Then a case will be classified into a class label with the highest posterior probability.

Let C be a random variable representing k^{th} class label (which in this study, would be only class 1 and class 2), and $X_i = \{x_1, x_2, x_3, \ldots, x_d\}$ be a vector of random variables denoting the observed feature values. To predict the class of a test feature X_i, Bayes' theorem is used to assess the probability that $P(C = c_k|X_i = x_i) = \frac{P(C=c_k)P(X=x_i|C=c_k)}{P(X=x_i)}$ [23]. Since $P(X = x_i)$ is constant to all categories, $P(X_i = x_i)$ in the problem equation can be ignored, and the equation can be simplified to $P(C = c_k|X_i = x_i) \propto P(C = c_k)P(X_i = x_i|C = c_k)$ [24].

The expected number of classification errors can be minimised by assigning data with variables vector X_i to the class c_k for which $P(C = c_k)$ is highest. However, we do not know the $P(c_k|X_i)$ and must estimate

it from the data, which is difficult to do directly [25]. Bayes' rule suggests that, instead of estimating $P(X_i|c_k)$, $P(c_k)$ and $P(X_i)$; we should combine those estimates to get an estimate of $P(c_k|X_i)$. Estimating $P(c_k|X_i)$ would be an issue for high-dimensional data because it would involve an astronomical number of possible values for $X_i = \{x_1, x_2, x_3, \dots, x_d\}$. The proposed approach is to assume that the distribution of X conditional on c_k can be decomposed in this manner for all c_k, where $P(X_i|c_k) = \prod_{i=1}^{d} P(X_i|c_k)$ [25]. So then $P(c_k|X_i) = P(c_k) \times \frac{\prod_{i=1}^{d} P(X_i|c_k)}{P(X_i)}$, where $P(X_i) = \sum_{k'=1}^{\varepsilon c} P(c_{k'}) \times \prod_{j'=1}^{d} P(X_{j'}|c_{k'})$ which would lead the problem to the estimation of NB to $P(\widehat{c_k|X_\iota}) = P(\widehat{c_k}) \times \frac{\prod_{i=1}^{d} P(\widehat{X_\iota|c_k})}{P(\widehat{X_\iota})}$. Since our objective for classification is to minimise the number of errors, then the classification of a case with vector X_i to the class c_k is based on the highest $P(\widehat{c_k|X_\iota})$. In addition, the $P(\widehat{X_\iota})$ in the $P(\widehat{c_k|X_\iota})$ problem is not explicitly computed for minimum error classification because it remains the same for all c_k. Thus, the classification would be accurate as long as the correct class has the highest value of $P(\widehat{c_k}) \times \prod_{i=1}^{d} P(\widehat{X_\iota|c_k})$. To summarise, NB classifiers have been utilised for the multi-omics dataset and show a promising result on the classifier's effectiveness in terms of its performance [26-28].

k-Nearest neighbour

The *k*-nearest neighbour (*k*-NN) algorithm is the most famous and widely used classification algorithm for predicting the class of a record or sample with an unspecified class based on the class of its neighbour. Since its inception [29,30], the *k*-NN has been the most preferred algorithm because it can manage classification and regression problems without involving complex mathematical procedures. In addition, the *k*-NN is a nonparametric method in a lazy learning algorithm, where it directly searches through all the training samples by calculating the distance between the testing sample and the training sample to identify its nearest neighbour and produce the output for classification [31-33]. Since *k*-NN is a nonparametric algorithm, it does not assume the parametric form of the decision boundary, and it can also handle both binary and multiclass data.

The logical operation of *k*-NN can be defined as follows: Firstly, it is to determine a set of training samples and a query, then to find a point that is the closest to the query, and next, to assign its class label to the query.

The training set is defined as $T = \{x_i, y_i\}_{i=1}^{k}$, where $x_i \in \mathbb{R}^d$ is the training vector in the d-dimensional feature space, and y_i is the class label $\{0,1\}$. There are 2 steps to assigning the unknown class y' from a query x'.

The first step is to set a k nearest similar labelled targets for the query x'. The selection of the neighbourhood size k has a significant impact on the performance of *k*-NN classifiers. If k is very small, the local estimate tends to be very poor due to being influenced by scattered and noisy data, and ambiguous or mislabelled points [30]. If k is too large, the estimate would lead to over-smoothing and lose the true patterns in the data. To solve the problem, we searched for the best k parameter using 10-fold cross-validation. In this step, we arranged the data to find the distance between the data points. The distance measures available in *k*-NN include Euclidean distance and standard Euclidean distance, Manhattan (city-block) distance, Chebychev distance, angle between vectors (Angle) distance, Pearson correlation distance, squared Euclidean distance, Mahalanobis distance, and Minkowski distance [34]. In our work, we let $T' = \{x_i, y_i\}_{i=1}^{k}$ be arranged in ascending order in terms of Euclidean distance $d(x', x_i)$ between x' and x_i; $d(x', x_i) = \sqrt{(x' - x_i)^T (x' - x_i)}$. Then in the second step, we predicted the class label by the majority voting of its nearest neighbours such that $y' = \arg\max \sum_{(x_i, y_i) \in T'} \delta(y = y_i)$, where y is the class label, y_i is the class label for the i[th] nearest neighbour among its k nearest neighbour. $\delta(y = y_i)$ is the Dirac delta function, taking value if $y = y_i$ and zero if otherwise.

The *k*-NN classifier was chosen in this simulation because it has shown promising results in its effectiveness and performance in handling multi-omics datasets in many previous literatures [35-38].

Random forest

The random forest (RF) is an ensemble learning algorithm useful for supervised high-dimensional feature problems [39,40]. It was developed by Breiman [41], and has become a prominent nonparametric method and machine learning approach for classification and regression [42]. RF uses an ensemble classification tree using a bootstrap sample of the data, and at each split, the candidate set of variables is a random subset of the variable. In this paper, we focus on the use of RF for classification tasks.

Assuming a training set T with p variables, the sample size n and T_k is a bootstrap training set sampled from T with replacement containing m random variables $(m \leq p)$ for n sample size. A random tree is a

tree drawn randomly from a set of possible trees for m random variables at each node [43]. The RF is a collection of random tree classifiers, using the CART (classification and regression tree) algorithm.

From the training set T, the number of m random variables are selected at random from the p variables, and the best variable from this set of m is used to split the node. These values of m are held constant during forest growth. Variable selection measures for m include Entropy reduction, Gini Index, information gain ratio, chi-square test, F-test and Variance reduction method [44]. In the CART algorithm used by the RF classifier, the Gini Index is used as a variable selection measure. The Gini Index can be written as $gini(\mathrm{T}) = 1 - \sum_{j=1}^{n} p_j^2$, where $p_j = \frac{|C_j;T|}{|T|}$ is the relative frequency of class j in dataset T. The Gini Index is the minimum value when all cases fall into a single target category and the maximum value when the cases are equally distributed to all classes. Besides, the weighted Gini Index of subset T_k, which will be resulting from the choice of partition A (a discrete-valued variable); $Gini_A(T) = \frac{|T_1|}{|T|} gini(T_1) + \frac{|T_2|}{|T|} gini(T_2)$ [45]. The split variable function is the difference between Gini Index and the weighted Gini Index and is called the Gini gain $\Delta Gini(A) = Gini(T) - Gini_A(T)$. The attribute that maximises the reduction in the impurity is selected as the splitting variable.

Each tree is grown to the maximum depth on new training data using a combination of the variables. These grown trees are not pruned [44], and the number of trees to be grown is subjective, and RF does not overfit when the number of trees increases [41,46]. In this paper, we chose to grow 5000 trees for RF.

Various studies have shown the practicability of the RF classifier in handling multi-omics datasets, including the ability to produce highly accurate and promising parsimonious results [47-51].

Support vector machine

Support vector machine (SVM) is a machine learning technique based on the statistical learning theory proposed by Vapnik [52]. The function of SVM is to create a gutter that separates the points of 2 classes using a hyperplane line [53]. It is a supervised machine learning algorithm that trains the classifier function using pre-labelled data [54]. SVM is commonly explained in terms of either linear optimisation or gradient descent. This section discusses a brief description of SVM, starting with linearly separated data and followed by kernel based SVM. A complete and detailed explanation of SVM can be found in books by Vapnik [52] and Cristianini and Shawe-Taylor [55].

For linear separated classification, suppose there is a p-dimensional and N-point training dataset represented as $\{x_i, y_j\}$, where $i = 1,2,3, \dots, l$, $y_i = \in \{-1,1\}$ and $x_i \in \mathbb{R}^d$. x_i are the data points and y_i is either -1 or 1, denoting that the sample can be in class 1 or class 2. SVM tries to put a line to separate the vector of y_i called the hyperplane by $\vec{\mathbf{w}} \cdot \vec{\mathbf{x}} + b = \sum_{i=1}^{n} w^T x_i + b = 0$, where b is the scalar that offsets from the origin [56] and $\vec{\mathbf{w}}$ is the n-dimensional vector [57]. Based on **Figure 1**, the optimal decision boundary (hyperplane) separates the samples, and the nearest support vector is maximum by $(w^T x_i + b) \geq 1$ for y_i is positive, otherwise, $(w^T x_i + b) \leq -1$. The optimal decision boundary (width of the gutter) is obtained by minimising the distance between the positive and the negative points across the hyperplane, which leads to $\min \frac{1}{2} \|\boldsymbol{w}\|^2$. By introducing Lagrange multipliers $\alpha_i (i = 1,2,3, \dots, n)$, from a convex quadratic optimisation problem, it leads to $\min_w L = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i^n \alpha_i y_i (\mathbf{w}^T x_i + b) + \sum_i^M \alpha_i$. This problem can be reduced to its dual problem by differentiating L with respect to $\boldsymbol{w}$ and b, $\frac{\delta L}{\delta \boldsymbol{w}} = \sum \alpha_i y_i x_i$, $\frac{\delta L}{\delta b} = -\sum \alpha_i y_i = 0$ respectively. The final dual problem is $\max_\alpha L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j$, where α_i are positive value for $i = 1,2,3, \dots, n$ and $\sum \alpha_i y_i = 0$. Based on the dual problem, the maximisation depends on the dot product of pairs of the samples $(\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j)$ [54,56-59].

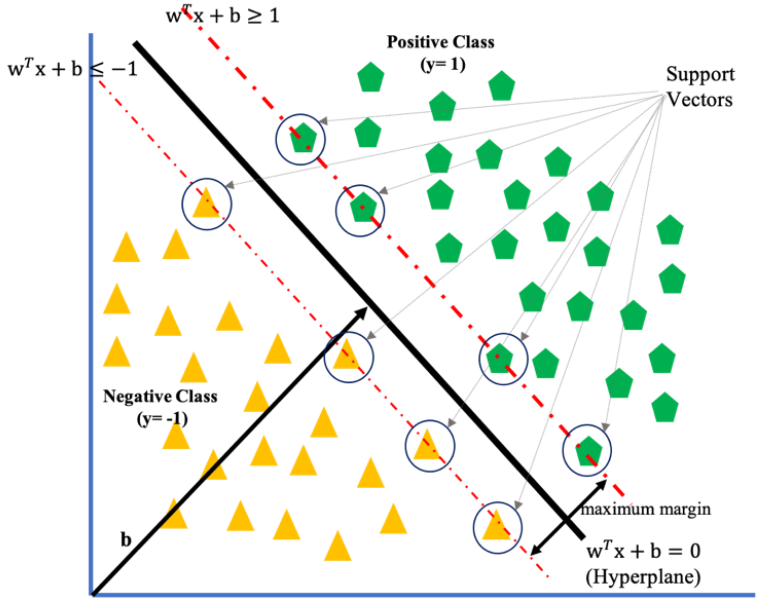


Figure 1 Optimal hyperplane on linear separated support vectors.

In most cases, linear separated data rarely occurs in 2 classes of classification. In the case of non-linear separation or classification, the slack variable ($\xi_n \geq 0$) is introduced into each training sample, where $\xi = |y - (w^T \emptyset(x) + b)|$. The misclassified points are represented by $\xi_n \geq 1$, while $\xi_n \leq 1$ represents points within the margin and $\xi_n = 0$ represents points on the boundary. These problems can be expressed by minimising the $C \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2$ subject to $y_i(w^T x_i + b) \geq 1 - \xi_i$, where C is the cost parameter that controls the penalty of the points outside the margin. The non-linear mapping and kernel functions transformed the input data into high-dimensional space by $g(\mathbf{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \emptyset(\mathbf{x}_i)^T \emptyset(\mathbf{x})) + b$, where in this case, the $\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j$ were replaced by $\emptyset(\mathbf{x}_i)^T \emptyset(\mathbf{x})$. However, we do not need to know the functional form of $\emptyset(\mathbf{x}_i)$ because it is defined by the selected kernel, $K(\mathbf{x}_\text{i}, \mathbf{x}_\text{j}) = \emptyset(\mathbf{x}_i)^T \emptyset(\mathbf{x})$ and hence leads our mapping function to $g(\mathbf{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_\text{i}, \mathbf{x}_\text{j})) + b$. Kernels used in SVM classification include linear, polynomial, radial based, sigmoid, hyperbolic tangent, Bessel, ANOVA radial basis, and linear splines in 1 dimension. In our SVM classifier simulation, we used the linear, polynomial, and radial basis kernels. The functions of selected kernel types are presented in **Table 2**.

Table 2 Widely used kernel in SVM classification.

Kernel types	Kernel functions
Linear kernel	$K(\mathbf{x}_\text{i}, \mathbf{x}_\text{j}) = \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j$
Polynomial kernel	$K(\mathbf{x}_\text{i}, \mathbf{x}_\text{j}) = ((\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j) + 1)^d$
Radial basis kernel	$K(\mathbf{x}_\text{i}, \mathbf{x}_\text{j}) = \exp(-\gamma \|x - x_i\|^2)$

The SVM classifiers were chosen to be part of the testing classifiers in this work because of their ability to identify features in multi-omics datasets and classify various diseases [60-64].

Methodology

Real transcriptomics dataset

The synthetic data used in this study were generated based on 3 parameters: Base intensity, random effects for biological patients, and experimental noise. The base intensity is based on a real dataset, obtained from the "Towards Useful Ageing (TUA): Neuroprotective Model for Healthy Longevity Among the Malaysian Elderly" research project funded by the Long-Term Research Grant Scheme (LRGS) by the Ministry of Education, Malaysia. The ethical approval for the TUA programme was obtained from the ethics committees of both Universiti Teknologi MARA (reference no. 600-RMI [5/1/6/01]) and the

University of Malaya Medical Center (reference no: PPUM HU-61/12/1-1]). The dataset consists of 92 people living with Alzheimer's disease (AD) and 92 healthy controls (HC), recruited from the Memory and Geriatric Clinic of the University of Malaya Medical Center (UMMC).

Simulation

To identify the most effective classifier for handling high dimensional data, we examined the performance of each classifier via a simulation study using a real dataset in the model. The dataset contained a huge number of features to generate realistic gene expression data where we can systematically vary different data characteristics (such as variability, effect size, and correlation) to investigate the effect of the characteristics on the performance of the classifier algorithms.

In classification problems where high-dimensional data is prominent, 'realistic' data is crucially required to examine the possible characteristics involved in the data set, particularly in microarray gene expression data. The term realistic means that it not only has variable creation and its correlation becomes a prescribed structure but also that it is noisy [65].

Therefore, we designed our simulation model to use real high dimensional data sets to generate realistic gene expression data in which we can systematically vary different data characteristics (such as variability, effect size, and correlation) and evaluate the performance of the ML algorithms.

Generating realistic synthetic dataset

Three important steps should be taken to generate a realistic synthetic dataset: 1) establishing independent markers; 2) initiating a correlation structure; and 3) correlating the independent markers.

First step: Establishing independent markers

To simulate realistic omics data, we based our simulation on real transcriptomics for AD patients (hereinafter referred to as the "base of intensity"). The base of intensity contained a gene expression dataset for 22,254 markers in 92 AD individuals (case group) and 92 healthy control (HC) individuals (control group). We assumed that the base of intensity was free from any systematic or random variability as the data was averaged over many sample observations. Therefore, it was proportional to the true expression levels of the corresponding markers in the set.

To initiate the simulation, we followed the work of Khondoker *et al.* [11], and it was assumed that the mean of the simulated variables, μ was close to the true markers. Then a random effect model was used to represent the pre-specific extent of stochastic noise in the dataset. Next, 2 levels of stochastic variability (between and within subject variation) were applied as biological and technical variations in the multi-omics literature. Each marker (p) was created independently (uncorrelated) based on the random effect model: $x_{ij} = \mu + b_i + \epsilon_{ij}$ [11]; where x_{ij} denotes the simulated base of an intensity value for a marker in the j^{th} replicate $(j = 1,2,3, \dots r)$ of the i^{th} sample $(i = 1,2,3, \dots, n)$. The parameter μ is randomly selected from the base of intensity taken from the transcriptomics dataset, and it is assumed to be proportional to the true markers, b_i is the random effect for i^{th} biological patients and ϵ_{ij} is the random experimental noise. In this simulation, we assume b_i and ϵ_{ij} are independent and identically distributed $(i.i.d)$ random variables distributed according to a multivariate normal distribution with mean equal to zero and the variance varies such that $b_i \sim N(0, \sigma_b)$ and $\epsilon_{ij} \sim N(0, \sigma_\epsilon)$, respectively.

These random effect models allow generating independent and uncorrelated biomarkers in the multi-omics dataset with various amounts of stochastic noise controlled by the parameter σ_b and σ_ϵ. After that, the simulation begins by generating every feature as an independent (uncorrelated) variable, where it is treated as a univariate variable [65]. The number of independent markers generated in this step was 100, 200, 300, 400, and 500. The variations of the sample size, random effect, and random experimental noise are presented in **Table 3**.

Second step: Initiating correlation structure

In the central dogma of statistics and biology, independent variables for high dimensional data are essential in multi-omics fields, where groups of genes, proteins, and metabolites operate together to perform specific biological functions [11]. Thus, the synthetic datasets that we created in the first step should be correlated with each other. After creating each variable and sample in the first step, the data on each marker was averaged over replicates. A multivariate structure was then introduced to the data from multiple (p) markers by imposing a p-dimensional covariance structure via Cholesky root transformation.

Next, a block diagonal correlation structure was developed using Hub-Toeplitz (HT) for h^{th} block. The idea of correlation structure was proposed by Hardin *et al.* [65], where the hybridization of the Toeplitz

matrix and Hub observation model supported the simulation of classification and discriminant analysis to model the correlations between case and control. In the Toeplitz correlation matrix, adjacent pairs of observations are highly correlated, whereas those that diverge are less correlated. However, in the Hub observation model, each case and control observations are correlated with the hub observation on a decreasing strength (this is from the operator provided's maximum correlation to a specified minimum correlation). The HT correlation structure is as follows:

$$R_h = \begin{bmatrix} 1 & r_{h,2} & r_{h,3} & r_{h,4} & \cdots & r_{h,d2} \\ r_{h,2} & 1 & r_{h,2} & r_{h,3} & \cdots & r_{h,d_h-1} \\ r_{h,3} & r_{h,2} & 1 & r_{h,2} & \cdots & r_{h,d_h-2} \\ r_{h,4} & r_{h,3} & r_{h,2} & 1 & \cdots & r_{h,d_h-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{h,d_h} & r_{h,d_h-1} & r_{h,d_h-2} & r_{h,d_h-3} & \cdots & 1 \end{bmatrix}$$

where $h = 1,2,3, \ldots, H$ correlation block. The symbol r represents the correlation value to be input during the simulation process. The value of each $r_{h,j} = r_{max} - \left(\frac{l-2}{d_h-2}\right)^{\gamma} (r_{max} - r_{min})$ and it generates correlations that decreases from the maximum correlation (r_{max}) to the minimum correlation (r_{min}) where $2 \leq l \leq d_h$. The parameter γ in $r_{h,j}$ was set to be equal to 1 (linear) to control the declination rate for each block. HT correlation structure assumes a predefined correlation between a network hub, especially in the first markers and other variables within the block where the correlation between the hub and l^{th} markers decline as l increases. We set r_{min} to be varied between 0.2 and 0.4 randomly, where the r_{max} also varied between 0.6 and 0.8 randomly to explore the impact of feature correlations [11,66].

Third step: Correlating the uncorrelated markers

In the following steps, the number of blocks was randomly selected from the set $\left[1,2,3, \ldots, {}^{p}/_{3}\right]$ [11]. If the number of markers is a multiple of correlation blocks (H), the whole blocks were considered to have the same dimension ($d_h = p/H$). If the number of markers is not a multiple of H, then the first block was selected to have dimension $d_1 = \frac{p}{H} + mod(p, H)$. In contrast, the rest of the markers would be $d_h = p/H$, where mod represented the remainder of the division.

Then, Cholesky root transformation was used to force the block-diagonal HT structure on the uncorrelated data by $R = diag[R_1, R_2, R_3, R_4, \ldots, R_H]$. The Cholesky root was computed based on $C = \sqrt{V} \times R \times \sqrt{V}$ and data were transformed with the desired covariance structure by $\bar{Y} = \bar{X}C$, where $\bar{X}$ is the average of the markers over the replicates and $V = (\sigma_b^2 + \frac{\sigma_e^2}{r})I_p$. The I_p is a p-dimensional identity matrix. Unlike Khondoker *et al.* [11], this process was done for both the AD and HC groups separately. Then, both groups were combined into one dataset.

Simulating threshold

The purpose of the simulation was to investigate the effect of sample size (n) feature set size (p) on fixed biological variation (σ_b) and experimental variation (σ_e). We considered feature sets of sizes 100, 200, 300, 400, and 500. We believed that the range of sizes was reasonable to understand the effects of the feature set size on the classification performance. Even though the microarray data are high-throughput data and can have more markers, the ultimate aim of classification for this type of data is to achieve good class prediction performance with the smallest number of markers [11]. The combination values for sample size, biological variation, and experimental variation are presented in **Table 3**. All simulations were repeated 500 times, and the average classification performance, such as sensitivity, specificity, accuracy, error rate, and F-measure, was recorded over 500 simulated datasets.

Table 3 Combination values for sample size (n), biological variation (σ_b) and experimental variation (σ_e) considered in the simulation.

n	σ_b	σ_e
10	4.5	4.5
20	4.0	4.0
30	3.5	3.5

n	σ_b	σ_e
40	3.0	3.0
50	2.5	2.5
100	2.0	2.0
200	1.5	1.5
300	1.0	1.0
400	0.5	0.5
500	0.1	0.1

Algorithm

We evaluated the SVM classifier with 3 kernels, namely linear, radial basis (RBF), and polynomial. The performance of each kernel was measured using the F-measure, sensitivity, specificity, accuracy, and error rate to identify better-performing kernels for high-dimensional data analysis. The F-measure evaluates the harmonic mean of precision and recall for binary classification, and it is used as a conventional method to measure the performance of the classifier [67]. We used R [68] with RStudio [69] as the IDE and the "e1071" package to analyse the performance of SVM with these kernels [70]. We performed a grid search of the chosen kernels to find the best cost and gamma parameters to build the SVM model for each repetition in the simulation. We searched for the combination of cost and gamma, with cost ranging from 1 to 100 and gamma ranging from 0.000001 to 0.1. The model with parameters that gave a high F-measure value with a minimum error rate was selected for each repetition in the simulation.

The same R-package, "e1071" was used to analyse NB performance in classifying high-dimensional data. Besides, we used the "trainControl" and "train" functions [71] in the R-package "caret" to search for the best *k* parameter for the *k*-NN classifier, and then the models were developed using the "knn" function in the "class" package [72].

For the RF classifier, we used "randomForest" packages developed by Liaw and Wiener. We searched for the best number of variables to be selected in building a single tree using the "mtry" function before building the classification tree (we call it RF-Optimised) and compared it with the "mtry" output, which was $\sqrt{p}$ (RF-Standard), as suggested by Breiman [41]. Sensitivity, specificity, accuracy, error rate, and F-measure were calculated using the "caret" package in RStudio to assess the performance of the classifiers [71].

The simulation analysis was carried out using a high-performance machine provided by the High-Performance Cloud Computing Center (HPCCC), Universiti Teknologi Petronas, Malaysia. The machines have 25 CPU cores and 100 GB of memory. The repetitions in the simulation were done using a parallel computing approach by implementing the "foreach" [73] and "doParallel" [74] packages in R to reduce processing time. Using the stated equipment to complete a single simulation for a certain classifier was still time-consuming, but the performance was better than the traditional computational method.

For cross-validation purposes, the simulated synthetic data was split into training and testing data by 70 and 30 %, respectively. **Figure 2** represents the flowchart of the whole simulation process. The R syntax for this study is available upon request by writing to the corresponding author.

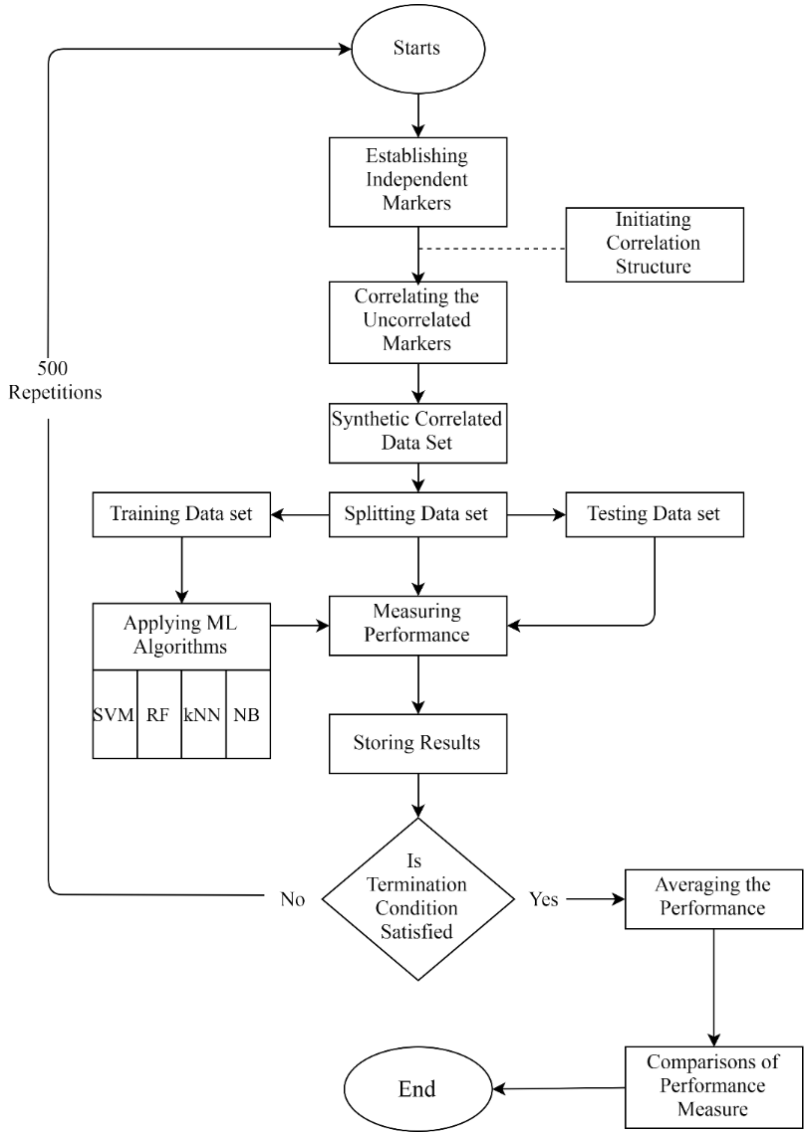


Figure 2 Flowchart of the simulation process.

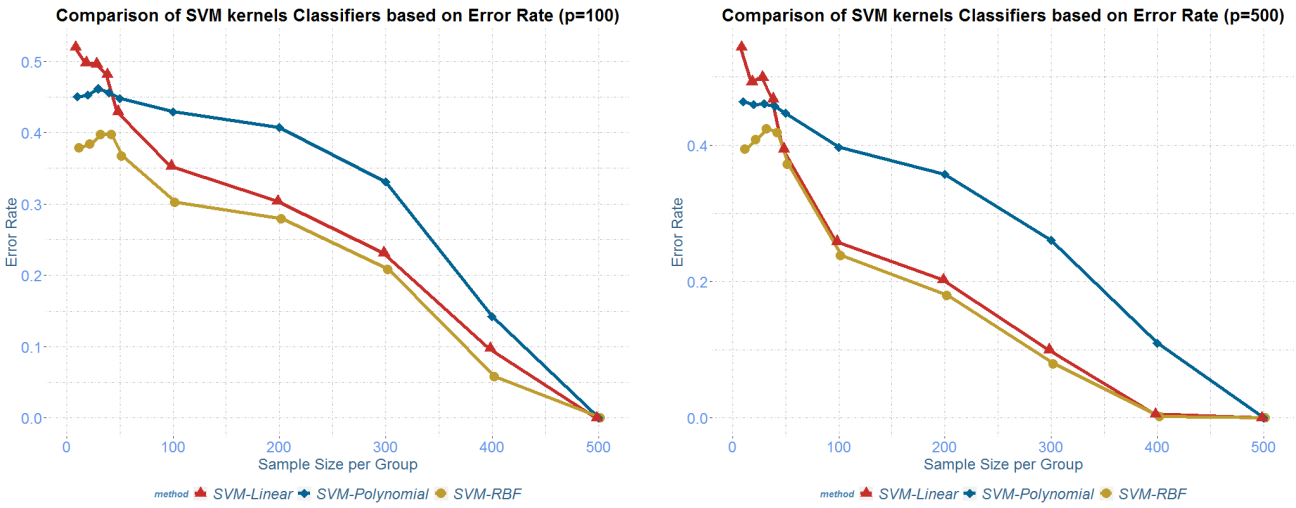
Results and discussion

The main objective of this paper was to compare classifier performance, i.e., SVM, RF, NB, and *k*-NN, in handling high-dimensional data. We performed a simulation of high-dimensional data from a transcriptomic dataset of AD patients and healthy controls provided by the TUA programme. The number of dimensions considered for the comparison of ML algorithms was 100, 200, 300, 400, and 500. The sample size generated for each dimension is presented in **Table 3**. Each analysis was repeated 500 times to get robust statistical measures.

The performance of the SVM family kernels under study for 100 and 500 dimensions is presented in **Table 4**. Complete performance measures for the 200, 300, and 400 dimensions are presented in **Appendix 1** (**Table A1** and **Figure A1**). SVM family kernels demonstrated better overall performance when the sample size increased. When the sample size was less than or equal to 50, the error rate and F-measure for the classifiers did not show any trends or stable performance for the 3 kernels. Khondoker *et al.* [11] reported a similar finding when they performed a simulation study using SVM. However, when the sample size reached above 100 per group, the error rate decreased in parallel with the sample size increment, regardless of dimension (p). We also found that the RBF kernels performed better than the polynomial and linear kernels, and the performance was consistent for all dimensions. Furthermore, the performance of SVM with a linear kernel increased with the rise in the number of dimensions. The visual presentation of the performance is illustrated in **Figure 3**.

Table 4 SVM model performance (p = 100 and p = 500).

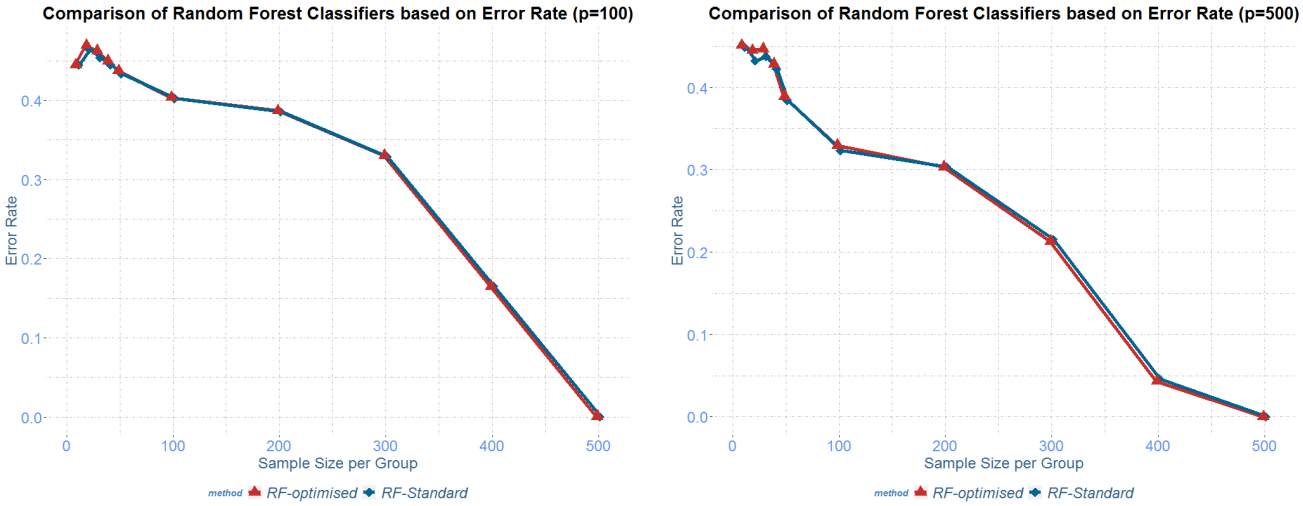
Number of dimensions	Sample size per group	Error Rate (F-measure)		
		Linear	RBF	Polynomial
100	10	0.520 (0.520)	0.379 (0.663)	0.450 (0.659)
	20	0.498 (0.495)	0.384 (0.647)	0.453 (0.622)
	30	0.496 (0.497)	0.398 (0.634)	0.461 (0.605)
	40	0.481 (0.513)	0.398 (0.635)	0.456 (0.610)
	50	0.429 (0.564)	0.368 (0.658)	0.448 (0.613)
	100	0.353 (0.646)	0.303 (0.707)	0.429 (0.638)
	200	0.304 (0.699)	0.279 (0.726)	0.407 (0.650)
	300	0.231 (0.769)	0.208 (0.794)	0.331 (0.705)
	400	0.097 (0.902)	0.058 (0.942)	0.142 (0.869)
	500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)
500	10	0.544 (0.504)	0.395 (0.668)	0.464 (0.671)
	20	0.493 (0.499)	0.408 (0.657)	0.460 (0.640)
	30	0.499 (0.496)	0.424 (0.648)	0.461 (0.626)
	40	0.468 (0.523)	0.419 (0.652)	0.457 (0.638)
	50	0.394 (0.602)	0.372 (0.684)	0.447 (0.622)
	100	0.259 (0.741)	0.238 (0.771)	0.397 (0.642)
	200	0.203 (0.797)	0.180 (0.821)	0.357 (0.699)
	300	0.099 (0.900)	0.079 (0.921)	0.261 (0.775)
	400	0.005 (0.994)	0.002 (0.998)	0.110 (0.903)
	500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)



(a) $p = 100$ (b) $p = 500$

Figure 3 Comparison of SVM classifiers based on error rate according to the number of dimensions (p) and sample size.

For RF classifiers, we found that the performance of RF-Optimised was slightly better than RF-Standard. As illustrated in **Figure 4**, the lines for RF-optimised and RF standards are almost overlapping for the synthetic data (p=100 and p=500). The complete figure and table of performance measures are given in **Appendix 2** (**Table A2** and **Figure A2**).



(a) $p = 100$ (b) $p = 500$

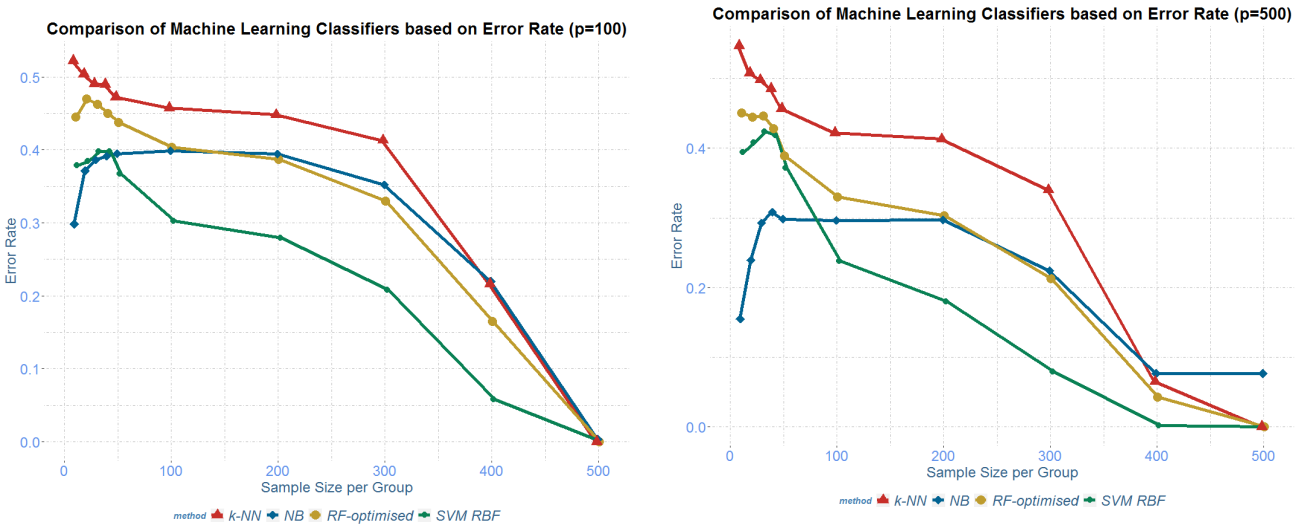
Figure 4 Comparison of RF classifiers based on error rate according to the number of dimensions (p) and sample size.

Next, we compared SVM RBF, RF-optimised, *k*-NN, and NB classifiers to identify the better-performing classifiers for high-dimensional datasets. In **Table 5**, we summarise the results of the ML classifier's performance based on error rate and F-measure for 100 and 500 dimensions. Complete performance measures for the 200, 300, and 400 dimensions for these classifiers were presented in **Appendix 3** (**Table A3** and **Figure A3**). Based on **Table 5**, the performance of SVM RBF is consistently better than the others, and the error rate for this classifier is always at its lowest point when the sample size is larger than 100 per group. The NB classifier performs better when the sample size is less than 50 per group, and its error rate is the lowest when the number of variables is higher than the sample size. For all dimensions under study, based on error rate and F-measure, when the sample size was smaller than the number of dimensions, NB always performed well compared to other classifiers. As illustrated in **Figure 5**, *k*-NN has the lowest performance for high-dimensional data. When the sample size exceeds the number of dimensions, the performance of RF-optimised systems improves as compared to NB. The RF-optimised performance shifted when the sample size was more than 200 per group.

Table 5 Comparison of 4 ML classifiers (p = 100 and p = 500).

Number of dimensions	Sample size per group	Error Rate (F-measure)			
		RF-Optimised	**SVM RBF**	***k*-NN**	**NB**
100	10	0.445 (0.589)	0.379 (0.663)	0.522 (0.543)	0.298 (0.715)
	20	0.470 (0.529)	0.384 (0.647)	0.504 (0.488)	0.371 (0.621)
	30	0.463 (0.530)	0.398 (0.634)	0.491 (0.499)	0.386 (0.607)
	40	0.450 (0.546)	0.398 (0.635)	0.490 (0.497)	0.392 (0.602)
	50	0.437 (0.554)	0.368 (0.658)	0.473 (0.514)	0.395 (0.600)
	100	0.404 (0.595)	0.303 (0.707)	0.458 (0.538)	0.399 (0.597)
	200	0.387 (0.613)	0.279 (0.726)	0.448 (0.560)	0.394 (0.604)
	300	0.330 (0.670)	0.208 (0.794)	0.413 (0.613)	0.352 (0.650)
	400	0.165 (0.835)	0.058 (0.942)	0.216 (0.810)	0.220 (0.781)
	500	0.0003 (1.000)	0.000 (1.000)	0.000 (1.000)	0.003 (0.996)

Number of dimensions	Sample size per group	Error Rate (F-measure)			
		RF-Optimised	SVM RBF	*k*-NN	NB
500	10	0.451 (0.582)	0.395 (0.668)	0.547 (0.532)	0.155 (0.841)
	20	0.445 (0.553)	0.408 (0.657)	0.508 (0.490)	0.239 (0.754)
	30	0.446 (0.553)	0.424 (0.648)	0.498 (0.487)	0.293 (0.706)
	40	0.428 (0.564)	0.419 (0.652)	0.485 (0.509)	0.308 (0.685)
	50	0.389 (0.608)	0.372 (0.684)	0.457 (0.530)	0.298 (0.696)
	100	0.330 (0.670)	0.238 (0.771)	0.422 (0.573)	0.296 (0.702)
	200	0.304 (0.696)	0.180 (0.821)	0.413 (0.602)	0.297 (0.702)
	300	0.213 (0.787)	0.079 (0.921)	0.340 (0.694)	0.224 (0.776)
	400	0.043 (0.957)	0.002 (0.998)	0.065 (0.940)	0.076 (0.923)
	500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.076 (0.923)



a) $p = 100$ b) $p = 500$

Figure 5 Comparison of 4 ML classifiers based on error rate.

Conclusions

Real-life data, such as microarray and molecular genetic-related data, is generally more complex and harder to get since the sequencing process incurs high costs and is time-consuming. To accurately identify potential biomarkers of any disease on microarray or omics datasets, selecting the right analytical methods is crucial to ensuring the analysis result has a low prediction error but high accuracy and sensitivity. This simulation study found that the NB was the best classifier for high-dimensional data, especially when the sample size was less than or equal to 50 per group. However, the SVM with the RBF kernel only performs better when the sample size exceeds 100 per group for all dimensions.

The sample size is an essential element in determining the choice of ML classifiers when dealing with high-dimensional data. When the sample size for high-dimensional models is small (less than 50), the performance of the ML classifiers is poor. The recommended sample size for classification analysis for high-dimensional data is at least 100 per group since the error rates for all classifiers keep on reducing as the sample size increases.

This study contributes to the theory where the comparison of ML classifier performance has made a remarkable finding in identifying the ML classifiers that work well on high-dimensional states of data and which ML classifiers work well on high-dimensional and low-dimensional states by comparing a combination of several sample sizes. Additionally, the synthetic dataset that we have developed can

significantly mimic the real dataset in 2 groups. The difference between our synthetic dataset and other datasets was the consideration of variability and correlation in the dataset based on each group.

Acknowledgements

This research was supported by university research funding under the BESTARI Grant (600-IRMI/DANA 5/3/BESTARI (113/2018)). We also acknowledge the research team of the LRGS project "Towards Useful Ageing (TUA): Neuroprotective Model for Healthy Longevity Among the Malaysian Elderly" (600-RMI/LRGS 5/3 [3/2012]) for their data and support. The authors would like to thank Universiti Teknologi MARA (UiTM) Perak, Tapah Campus, and UiTM Shah Alam, Malaysia for providing the facilities to conduct this study.

References

[1] W Zhang, Y Zeng, M Jiao, C Ye, Y Li, C Liu and J Wang. Integration of high-throughput omics technologies in medicinal plant research: The new era of natural drug discovery. *Front. Plant Sci.* 2023; **14**, 1073848.

[2] W Chiu, J Schepers, T Francken, L Vangeel, K Abbasi, D Jochmans, SD Jonghe, HJ Thibaut, V Thiel, J Neyts, M Laporte and P Leyssen. Development of a robust and convenient dual-reporter high-throughput screening assay for SARS-CoV-2 antiviral drug discovery. *Antiviral Res.* 2023; **210**, 105506.

[3] SB Koby, E Gutkin, F Gusev, CM Narangoda, O Isayev and MG Kurnikova. High-throughput binding free energy simulations: Applications in drug discovery. *Biophys. J.* 2023; **122**, 142a-143a.

[4] G Romeo and M Thoresen. Model selection in high-dimensional noisy data: A simulation study. *J. Stat. Comput. Simul.* 2019; **89**, 2031-50.

[5] P Gong, L Cheng, Z Zhang, A Meng, E Li, J Chen and L Zhang. Multi-omics integration method based on attention deep learning network for biomedical data classification. *Comput. Methods Programs Biomed.* 2023; **231**, 107377.

[6] Z Liu, Y Zhao, P Kong, Y Liu, J Huang, E Xu, W Wei, G Li, X Cheng, L Xue, Y Li, H Chen, S Wei, R Sun, H Cui, Y Meng, M Liu, Y Li, R Feng, X Yu, R Zhu, Y Wu, L Li, B Yang, Y Ma, J Wang, W Zhu, D Deng, Y Xi, F Wang, H Li, S Guo, X Zhuang, X Wang, Y Jiao, Y Cui and Q Zhan. Integrated multi-omics profiling yields a clinically relevant molecular classification for esophageal squamous cell carcinoma. *Cancer Cell* 2023; **41**, 181-95.

[7] Y Chen, J Meng, X Lu, X Li and C Wang. Clustering analysis revealed the autophagy classification and potential autophagy regulators' sensitivity of pancreatic cancer based on multi-omics data. *Cancer Med.* 2023; **12**, 733-46.

[8] SA Qureshi, L Hussain, U Ibrar, E Alabdulkreem, MK Nour, MS Alqahtani, FM Nafie, A Mohamed, GP Mohammed and TQ Duong. Radiogenomic classification for MGMT promoter methylation status using multi-omics fused feature space for least invasive diagnosis through mpMRI scans. *Sci. Rep.* 2023; **13**, 3291.

[9] G Zararsiz, D Goksuluk, S Korkmaz, V Eldem, GE Zararsiz, IP Duru and A Ozturk. A comprehensive simulation study on classification of RNA-Seq data. *PLoS One* 2017; **12**, e0182507.

[10] J Fan and R Li. Statistical challenges with high dimensionality: Feature selection in knowledge discovery. *In*: Proceedings of the International Congress of Mathematicians, Madrid, Spain. 2006, p. 595-622.

[11] M Khondoker, R Dobson, C Skirrow, A Simmons and D Stahl. A comparison of machine learning methods for classification using simulation with multiple real data examples from mental health studies. *Stat. Methods Med. Res.* 2016; **25**, 1804-23.

[12] CA Bobak, AJ Titus and JE Hill. Comparison of common machine learning models for classification of tuberculosis using transcriptional biomarkers from integrated datasets. *Appl. Soft Comput. J.* 2019; **74**, 264-73.

[13] DA Gredell, AR Schroeder, KE Belk, CD Broeckling, AL Heuberger, SY Kim, DA King, SD Shackelford, JL Sharp, TL Wheeler, DR Woerner and JE Prenni. Comparison of machine learning algorithms for predictive modeling of beef attributes using rapid evaporative ionization mass spectrometry (REIMS) data. *Sci. Rep.* 2019; **9**, 5721.

[14] T Ma, and A Zhang. Integrate multi-omics data with biological interaction networks using Multi-view Factorization AutoEncoder (MAE). *BMC Genom.* 2019; **20**, 944.

[15] G Albuquerque, T Lowe and M Magnor. Synthetic generation of high-dimensional datasets. *IEEE Trans. Vis. Comput. Graph.* 2011; **17**, 2317-24.

[16] DS Quintana. A synthetic dataset primer for the biobehavioural sciences to promote reproducibility and hypothesis generation. *eLife* 2020; **9**, e53275.
[17] K Moradzadeh, S Moein, N Nickaeen and Y Gheisari. Analysis of time-course microarray data: Comparison of common tools. *Genomics* 2019; **111**, 636-41.
[18] J Andrade and J Duggan. An evaluation of Hamiltonian Monte Carlo performance to calibrate age-structured compartmental SEIR models to incidence data. *Epidemics* 2020; **33**, 100415.
[19] B Hanczar and E Dougherty. On the comparison of classifiers for microarray data. *Curr. Bioinform.* 2010; **5**, 29-39.
[20] S Rogers and M Girolami. *A first course in machine learning*. CRC Press, Florida, 2016.
[21] Ahmed, Shahjaman, MM Rana and MNH Mollah. Robustification of naïve Bayes classifier and its application for microarray gene expression data analysis. *Biomed. Res. Int.* 2017; **2017**, 3020627.
[22] CR Stephens, HF Huerta, and AR Linares. When is the Naive Bayes approximation not so naive? *Mach. Learn.* 2018; **107**, 397-441.
[23] B Chandra and M Gupta. Robust approach for estimating probabilities in Naïve-Bayes classifier for gene expression data. *Expet. Syst. Appl.* 2011; **38**, 1293-8.
[24] J Liu and S Bo. Naive Bayesian classifier based on genetic simulated annealing algorithm. *Proc. Eng.* 2011; **23**, 504-9.
[25] DD Lewis. *Naive (Bayes) at forty: The independence assumption in information retrieval*. *In*: C Nédellec and C Rouveirol (Eds.). European conference on machine learning. Springer Berlin, Heidelberg, 1998, p. 4-15.
[26] L Zhang, C Lv, Y Jin, G Cheng, Y Fu, D Yuan, Y Tao, Y Guo, X Ni and T Shi. Deep learning-based multi-omics data integration reveals two prognostic subtypes in high-risk neuroblastoma. *Front. Genet.* 2018; **9**, 477.
[27] Z Ahmed. Practicing precision medicine with intelligently integrative clinical and multi-omics data analysis. *Hum. Genom.* 2020; **14**, 35.
[28] Y Gao, R Zhou and Q Lyu. Multiomics and machine learning in lung cancer prognosis. *J. Thorac. Dis.* 2020; **12**, 4531-5.
[29] E Fix, JL Hodges and Jr. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *Int. Stat. Rev.* 1989; **57**, 238-47.
[30] J Gou, L Du, Y Zhang and T Xiong. A new distance-weighted k-nearest neighbor classifier. *J. Inform. Comput. Sci.* 2012; **9**, 1429-36.
[31] J Hua, Z Xiong, J Lowey, E Suh and ER Dougherty. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics* 2005; **21**, 1509-15.
[32] M Kuhkan. A method to improve the accuracy of k-nearest neighbor algorithm. *Int. J. Comput. Eng. Inf. Tech.* 2016; **8**, 90-5.
[33] J Luengo, S García, I Triguero, J Maillo, F Herrera and D García-Gil. Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. *Wiley Interdiscipl. Rev. Data Min. Knowl. Discov.* 2018; **9**, e1289.
[34] S Draghici. *Data analysis tools for DNA microarrays*. Chapman and Hall/CRC, New York, 2003.
[35] X Dong, L Lin, R Zhang, Y Zhao, DC Christiani, Y Wei and F Chen. TOBMI: Trans-omics block missing data imputation using a k-nearest neighbor weighted approach. *Bioinformatics* 2019; **35**, 1278-83.
[36] M Song, J Greenbaum, J Luttrell, W Zhou, C Wu, H Shen, P Gong, C Zhang and HW Deng. A review of integrative imputation for multi-omics datasets. *Front. Genet.* 2020; **11**, 570255.
[37] QR Xing, NO Cipta, K Hamashima, YC Liou, CG Koh and YH Loh. Unraveling heterogeneity in transcriptome and its regulation through single-cell multi-omics technologies. *Front. Genet.* 2020; **11**, 662.
[38] DA Cusanovich, AJ Hill, D Aghamirzaie, RM Daza, HA Pliner, JB Berletch, GN Filippova, X Huang, L Christiansen, WSD Witt, C Lee, SG Regalado, DF Read, FJ Steemers, CM Disteche, C Trapnell and J Shendure. A single cell atlas of *in vivo* mammalian chromatin accessibility. *Cell* 2019; **174**, 1309-24.
[39] S Karimi, AA Sadraddini, AH Nazemi, T Xu and AF Fard. Generalizability of gene expression programming and random forest methodologies in estimating cropland and grassland leaf area index. *Comput. Electron. Agr.* 2018; **144**, 232-40.
[40] W Chen, X Xie, J Wang, B Pradhan, H Hong, DT Bui, Z Duan and J Ma. A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility. *Catena* 2017; **151**, 147-60.
[41] L Breiman. Random forests. *Mach. Learn.* 2001; **45**, 5-32.

[42] AJ Sage, 2018. *Random forest robustness, variable importance, and tree aggregation*. Ph. D. Dissertation. Iowa State University, Iowa.
[43] TM Oshiro, PS Perez and JA Baranauskas. *How many trees in a random forest? In*: P Perner (Ed.). International workshop on machine learning and data mining in pattern recognition. Springer Berlin, Heidelberg, 2012, p. 154-68.
[44] M Pal. Random forest classifier for remote sensing classification. *Int. J. Remote Sens.* 2005; **26**, 217-22.
[45] L Rutkowski, M Jaworski, L Pietruczuk and P Duda. The CART decision tree for mining data streams. *Inform. Sci.* 2014; **266**, 1-15.
[46] MN Abdullah, BW Yap and Y Zakaria. Metabolites selection and classification of metabolomics data on Alzheimer's disease using random forest. *In*: M Berry, AH Mohamed and B Yap (Eds.). Communications in Computer and Information Science. Springer, Singapore. 2016
[47] A Acharjee, B Kloosterman, RGF Visser and C Maliepaard. Integration of multi-omics data for prediction of phenotypic traits using random forest. *BMC Bioinformatics* 2016; **17**, 363-73.
[48] IN Jamil, J Remali, KA Azizan, NA Nor Muhammad, M Arita, HH Goh and WM Aizat. Systematic Multi-Omics Integration (MOI) approach in plant systems biology. *Front. Plant Sci.* 2020; **11**, 944.
[49] N Holzscheck, J Söhle, B Kristof, E Grönniger, S Gallinat, H Wenck, M Winnefeld, C Falckenhayn and L Kaderali. Multi-omics network analysis reveals distinct stages in the human aging progression in epidermal tissue. *Aging* 2020; **12**, 12393-409.
[50] W Tao, AN Concepcion, M Vianen, ACA Marijnissen, FPGJ Lafeber, TRDJ Radstake and A Pandit. Multi-omics and machine learning accurately predicts clinical response to Adalimumab and Etanercept therapy in patients with rheumatoid arthritis. *Arthritis Rheumatol.* 2021; **71**, 212-22.
[51] NA Bokulich, P Łaniewski, DM Chase, J Gregory Caporaso and MM Herbst-Kralovetz. Integration of multi-omics data improves prediction of cervicovaginal microenvironment in cervical cancer. *PLoS Comput. Biol.* 2022; **18**, e1009876.
[52] VN Vapnik. *The nature of statistical learning theory*. Springer, New York, 2000.
[53] R Kharoubi, K Oualkacha and A Mkhadri. The cluster correlation-network support vector machine for high-dimensional binary classification. *J. Stat. Comput. Simul.* 2019; **89**, 1020-43.
[54] B Ghaddar and J Naoum-Sawaya. High dimensional data classification and feature selection using support vector machines. *Eur. J. Oper. Res.* 2018; **265**, 993-1004.
[55] N Cristianini and J Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, 2014.
[56] Liming Shen, H Chen, Z Yu, W Kang, B Zhang, H Li, B Yang and D Liu. Evolving support vector machines using fruit fly optimization for medical data classification. *Knowledge Based Syst.* 2016; **96**, 61-75.
[57] M Singh and AG Shaik. Faulty bearing detection, classification and location in a three-phase induction motor based on Stockwell transform and support vector machine. *Meas. J. Int. Meas. Confed.* 2019; **131**, 524-33.
[58] S Huang, CAI Nianguang, PP Pacheco, S Narandes, Y Wang and XU Wayne. Applications of support vector machine (SVM) learning in cancer genomics. *Cancer Genom. Proteomics* 2018; **15**, 41-51.
[59] AJ Smola and BSCH Olkopf. A tutorial on support vector regression. *Stat. Comput.* 2004; **14**, 199-222.
[60] B Ray, W Liu and D Fenyo. Adaptive multiview nonnegative matrix factorization algorithm for integration of multimodal biomedical data. *Cancer Inform.* 2017; **16**, 1-12.
[61] QA Hathaway, SM Roth, MV Pinti, DC Sprando, A Kunovac, AJ Durr, CC Cook, GK Fink, TB Cheuvront, JH Grossman, GA Aljahli, AD Taylor, AP Giromini, JL Allen and JM Hollander. Machine-learning to stratify diabetic patients using novel cardiac biomarkers and integrative genomics. *Cardiovasc. Diabetol.* 2019; **18**, 78.
[62] B Ma, F Meng, G Yan, H Yan, B Chai and F Song. Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data. *Comput. Biol. Med.* 2020; **121**, 103761.
[63] Y Liu, F Liu, X Hu, J He, and Y Jiang. Combining genetic mutation and expression profiles identifies novel prognostic biomarkers of lung adenocarcinoma. *Clin. Med. Insights Oncol.* 2020; **14**, 1-10.
[64] S Ma, J Ren and D Fenyö. Breast cancer prognostics using multi-omics data. *AMIA Jt. Summits Transl. Sci. Proc.* 2016; **52**, 52-9.
[65] J Hardin, SR Garcia and D Golan. A method for generating realistic correlation matrices. *Ann. Appl. Stat.* 2013; **7**, 1733-62.
[66] Y Zhang, QV Liao and B Srivastava. Towards an optimal dialog strategy for information retrieval using both open- and close-ended questions. *In*: Proceedings of the 23rd International Conference on

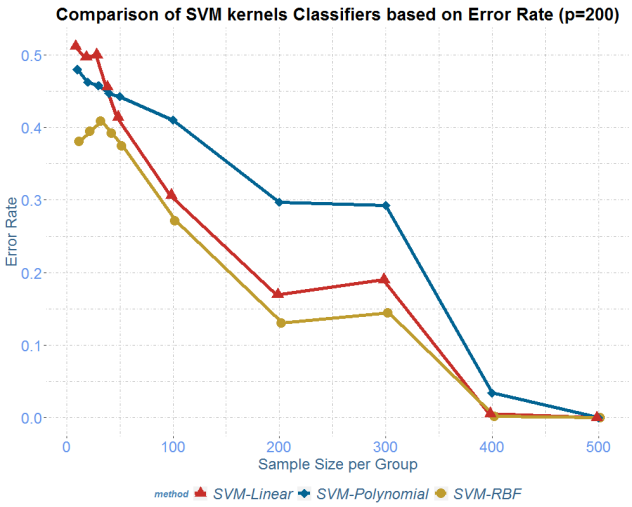
Intelligent User Interfaces, Tokyo, Japan.
[67] SAP Parambath, N Usunier and Y Grandvalet. *Optimizing F-measures by cost-sensitive classification. In*: Z Ghahramani, M Welling, C Cortes, N Lawrence and KQ Weinberger (Eds.). advances in neural information processing systems 27, Curran, New York, 2014.
[68] R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.
[69] RStudio Team. RStudio: Integrated development environment for R. RStudio, PBC, Boston, MA, 2021, Available: http://www.rstudio.com, accessed December 2022.
[70] D Meyer, E Dimitriadou, K Hornik, A Weingessel, F Leisch, CC Chang, CC Lin and MD Meyer. Package e1071, Available at: https://cran.r-project.org/web/packages/e1071/index.html, accessed December 2022.
[71] M Kuhn. Building predictive models in R using the caret package. *J. Stat. Software* 2018; **28**, 1-26.
[72] A Liaw and M Wiener. Classification and regression by random forest. *R News* 2002; **2**, 18-22.
[73] S Weston. Using the foreach package. R Foundation for Statistical Computing, Vienna, Austria, 2017.
[74] R Calaway, S Weston and D Tenenbaum. doParallel, Available: https://cran.r-project.org/package=doParallel, accessed December 2022.

APPENDIXES

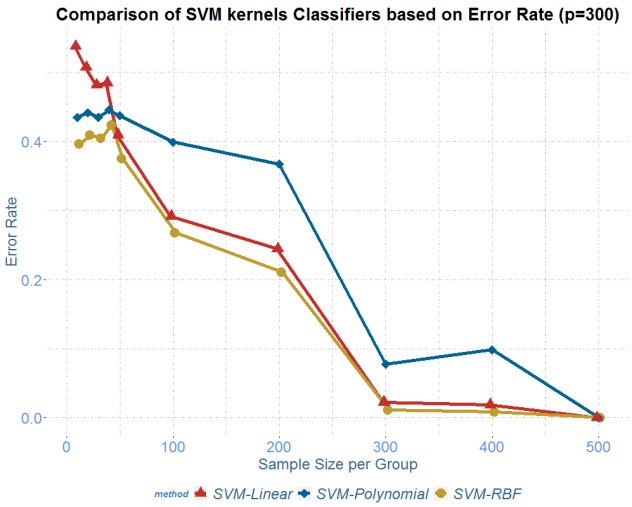
Appendix 1

Table A1 Comparison of SVM family kernel performance for combinations of synthetic dataset (p = 200, p = 300 and p = 400).

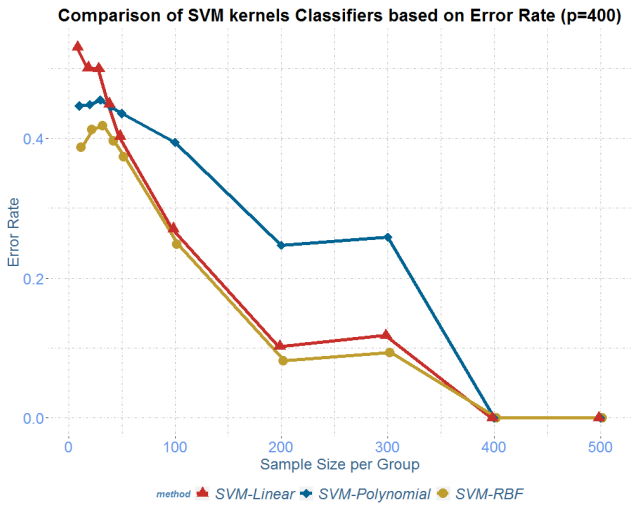
Sample Size per group	Error rate (F-measure)								
	p = 200			p = 300			p = 400		
	Linear	RBF	Polynomial	Linear	RBF	Polynomial	Linear	RBF	Polynomial
10	0.512 (0.521)	0.381 (0.665)	0.480 (0.645)	0.537 (0.522)	0.396 (0.665)	0.435 (0.682)	0.530 (0.521)	0.387 (0.675)	0.446 (0.674)
20	0.497 (0.500)	0.395 (0.655)	0.462 (0.634)	0.507 (0.485)	0.409 (0.651)	0.441 (0.647)	0.501 (0.490)	0.413 (0.652)	0.448 (0.651)
30	0.500 (0.495)	0.409 (0.639)	0.458 (0.607)	0.481 (0.518)	0.404 (0.658)	0.435 (0.637)	0.500 (0.488)	0.419 (0.650)	0.455 (0.634)
40	0.456 (0.541)	0.392 (0.650)	0.447 (0.618)	0.484 (0.506)	0.424 (0.640)	0.445 (0.625)	0.449 (0.544)	0.397 (0.662)	0.445 (0.632)
50	0.414 (0.580)	0.374 (0.667)	0.442 (0.628)	0.409 (0.585)	0.376 (0.673)	0.437 (0.613)	0.403 (0.593)	0.374 (0.679)	0.435 (0.63)
100	0.306 (0.692)	0.272 (0.736)	0.410 (0.645)	0.292 (0.706)	0.267 (0.748)	0.399 (0.661)	0.270 (0.728)	0.249 (0.762)	0.394 (0.651)
200	0.170 (0.830)	0.130 (0.870)	0.297 (0.727)	0.244 (0.755)	0.211 (0.790)	0.367 (0.688)	0.102 (0.897)	0.082 (0.918)	0.247 (0.763)
300	0.190 (0.810)	0.145 (0.856)	0.293 (0.740)	0.022 (0.977)	0.012 (0.988)	0.077 (0.923)	0.118 (0.882)	0.094 (0.906)	0.259 (0.773)
400	0.006 (0.994)	0.002 (0.998)	0.034 (0.967)	0.019 (0.981)	0.008 (0.992)	0.099 (0.910)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)
500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)



Number of dimensions are 200



Number of dimensions are 300



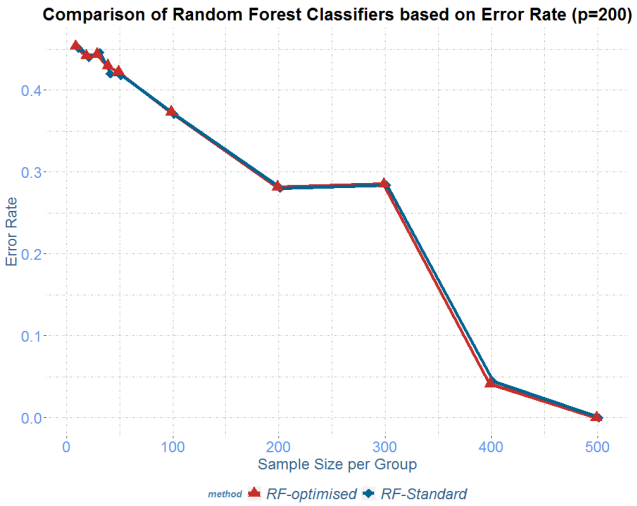
Number of dimensions are 400

Figure A1 Comparison of three types of SVM classifiers based on error rate and sample size.

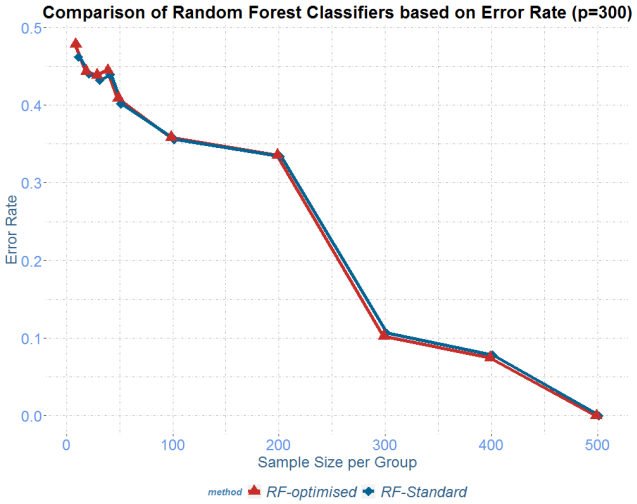
Appendix 2

Table A2 Comparison between RF standard and RF optimized performance for combination of synthetic dataset (p = 100, p = 200 p = 300, p = 400 and p = 500).

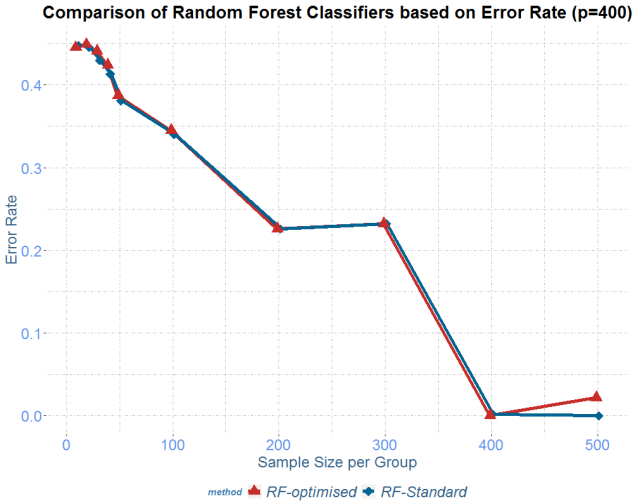
Sample size per group	Error rate (F-measure)									
	p = 100		p = 200		p = 300		p = 400		p = 500	
	RF Standard	RF Optimized	RF Standard	RF Optimized	RF Standard	RF Optimized	RF Standard	RF Optimized	RF Standard	RF Optimized
10	0.445 (0.591)	0.445 (0.589)	0.451 (0.578)	0.453 (0.577)	0.462 (0.587)	0.478 (0.567)	0.448 (0.587)	0.445 (0.595)	0.449 (0.593)	0.451 (0.582)
20	0.464 (0.539)	0.470 (0.529)	0.439 (0.559)	0.442 (0.554)	0.440 (0.559)	0.444 (0.554)	0.445 (0.548)	0.449 (0.545)	0.432 (0.557)	0.445 (0.553)
30	0.453 (0.537)	0.463 (0.530)	0.446 (0.544)	0.444 (0.547)	0.432 (0.565)	0.439 (0.558)	0.430 (0.559)	0.441 (0.545)	0.437 (0.561)	0.446 (0.553)
40	0.445 (0.550)	0.450 (0.546)	0.420 (0.575)	0.429 (0.564)	0.439 (0.554)	0.445 (0.547)	0.413 (0.581)	0.424 (0.571)	0.422 (0.570)	0.428 (0.564)
50	0.433 (0.561)	0.437 (0.554)	0.418 (0.577)	0.422 (0.574)	0.402 (0.593)	0.409 (0.586)	0.381 (0.616)	0.387 (0.610)	0.384 (0.611)	0.389 (0.608)
100	0.402 (0.596)	0.404 (0.595)	0.370 (0.629)	0.373 (0.626)	0.356 (0.644)	0.358 (0.640)	0.340 (0.658)	0.345 (0.653)	0.323 (0.675)	0.330 (0.670)
200	0.385 (0.615)	0.387 (0.613)	0.280 (0.718)	0.282 (0.717)	0.334 (0.665)	0.336 (0.664)	0.226 (0.774)	0.226 (0.774)	0.303 (0.696)	0.304 (0.696)
300	0.329 (0.671)	0.330 (0.670)	0.284 (0.716)	0.285 (0.715)	0.106 (0.894)	0.102 (0.898)	0.232 (0.767)	0.232 (0.768)	0.215 (0.785)	0.213 (0.787)
400	0.165 (0.834)	0.165 (0.835)	0.045 (0.955)	0.041 (0.959)	0.078 (0.922)	0.075 (0.925)	0.002 (0.998)	0.000 (1.000)	0.046 (0.954)	0.043 (0.957)
500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.022 (0.978)	0.000 (1.000)	0.000 (1.000)



Number of dimensions are 200



Number of dimensions are 300



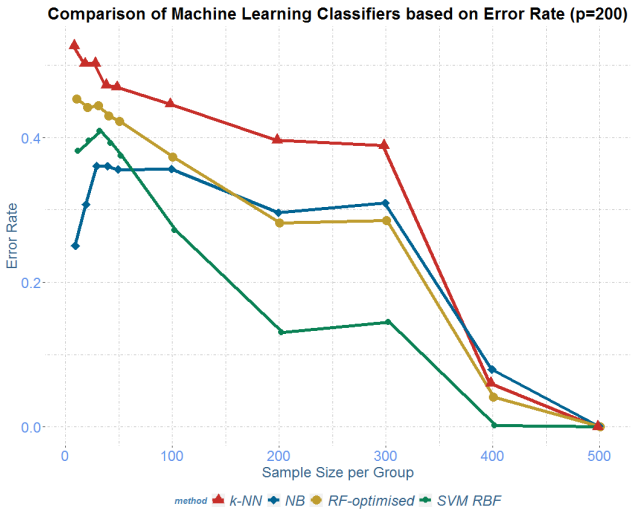
Number of dimensions are 400

Figure A2 Comparison of RF optimised and RF standard classifiers based on error rate and sample size.

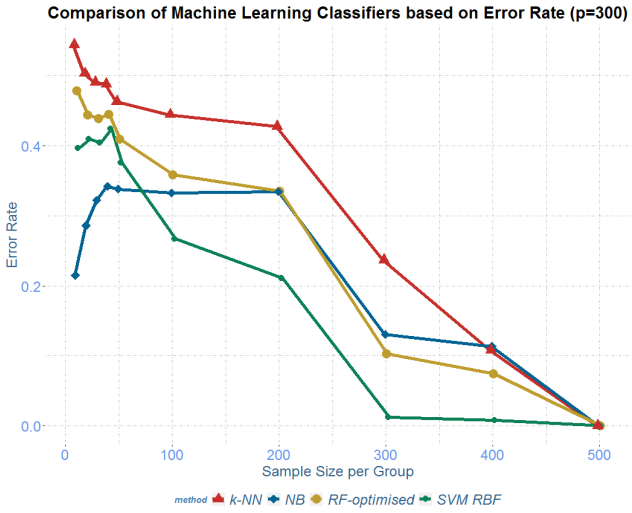
Appendix 3

Table A3 Comparison of 4 ML classifiers performance for combination of synthetic dataset (p = 200, p = 300 and p = 400).

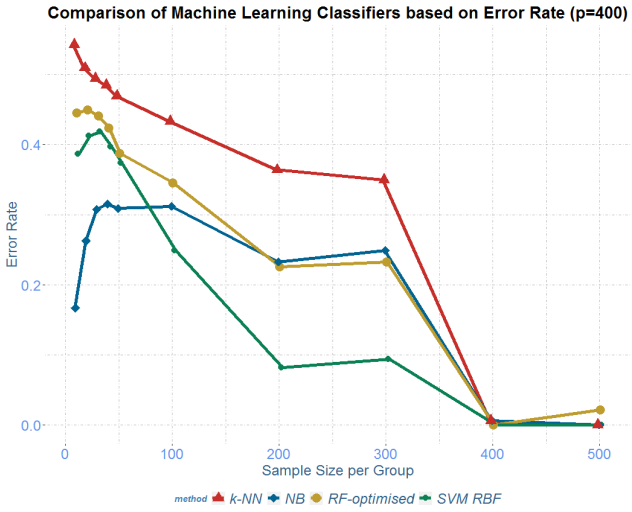
Number of dimensions	Sample size per group	Error rate (F-measure)			
		RF-Optimised	**SVM RBF**	***k*-NN**	**Naïve Bayes**
200	10	0.453 (0.577)	0.381 (0.665)	0.527 (0.534)	0.25 (0.755)
	20	0.442 (0.554)	0.395 (0.655)	0.502 (0.497)	0.308 (0.69)
	30	0.444 (0.547)	0.409 (0.639)	0.502 (0.478)	0.36 (0.633)
	40	0.429 (0.564)	0.392 (0.650)	0.472 (0.517)	0.36 (0.634)
	50	0.422 (0.574)	0.374 (0.667)	0.470 (0.518)	0.356 (0.642)
	100	0.373 (0.626)	0.272 (0.736)	0.446 (0.546)	0.356 (0.644)
	200	0.282 (0.717)	0.130 (0.870)	0.396 (0.601)	0.296 (0.704)
	300	0.285 (0.715)	0.145 (0.856)	0.389 (0.642)	0.309 (0.692)
	400	0.041 (0.959)	0.002 (0.998)	0.061 (0.942)	0.079 (0.920)
	500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.001 (0.999)
300	10	0.478 (0.567)	0.396 (0.665)	0.544 (0.514)	0.215 (0.792)
	20	0.444 (0.554)	0.409 (0.651)	0.503 (0.499)	0.286 (0.711)
	30	0.439 (0.558)	0.404 (0.658)	0.491 (0.503)	0.322 (0.673)
	40	0.445 (0.547)	0.424 (0.640)	0.488 (0.497)	0.342 (0.654)
	50	0.409 (0.586)	0.376 (0.673)	0.463 (0.521)	0.338 (0.658)
	100	0.358 (0.640)	0.267 (0.748)	0.444 (0.548)	0.332 (0.665)
	200	0.336 (0.664)	0.211 (0.790)	0.427 (0.585)	0.335 (0.664)
	300	0.102 (0.898)	0.012 (0.988)	0.237 (0.762)	0.131 (0.869)
	400	0.075 (0.925)	0.008 (0.992)	0.108 (0.902)	0.113 (0.886)
	500	0.000 (1.000)	0.000 (1.000)	0.000 (1.000)	0.001 (0.999)
400	10	0.445 (0.595)	0.387 (0.675)	0.543 (0.529)	0.167 (0.836)
	20	0.449 (0.545)	0.413 (0.652)	0.509 (0.489)	0.263 (0.733)
	30	0.441 (0.545)	0.419 (0.650)	0.494 (0.483)	0.307 (0.686)
	40	0.424 (0.571)	0.397 (0.662)	0.485 (0.504)	0.315 (0.679)
	50	0.387 (0.610)	0.374 (0.679)	0.469 (0.517)	0.309 (0.688)
	100	0.345 (0.653)	0.249 (0.762)	0.433 (0.562)	0.312 (0.686)
	200	0.226 (0.774)	0.082 (0.918)	0.364 (0.629)	0.233 (0.767)
	300	0.232 (0.768)	0.094 (0.906)	0.350 (0.683)	0.249 (0.750)
	400	0.000 (1.000)	0.000 (1.000)	0.006 (0.994)	0.006 (0.994)
	500	0.022 (0.978)	0.000 (1.000)	0.000 (1.000)	0.001 (0.999)



Number of dimensions are 200



Number of dimensions are 300



Number of dimensions are 400

Figure A3 Comparison of 4 ML classifiers based on error rate and sample size.