EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

Performance evaluation of supervised learning algorithms with various training data sizes and missing attributes

Chaluemwut Noyunsan, Tatpong Katanyukul* and Kanda Saikaew

Department of Computer Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Received 31 May 2017
Accepted 15 August 2017

Abstract

Supervised learning is a machine learning technique used for creating a data prediction model. This article focuses on finding high performance supervised learning algorithms with varied training data sizes, varied number of attributes, and time spent on prediction. This studied evaluated seven algorithms, Boosting, Random Forest, Bagging, Naive Bayes, K-Nearest Neighbours (K-NN), Decision Tree, and Support Vector Machine (SVM), on seven data sets that are the standard benchmark from University of California, Irvine (UCI) with two evaluation metrics and experimental settings of various training data sizes and missing key attributes. Our findings reveal that Bagging, Random Forest, and SVM are overall the three most accurate algorithms. However, when presence of key attribute values is of concern, K-NN is recommended as its performance is affected the least. Alternatively, when training data sizes may be not large enough, Naive Bayes is preferable since it is the most insensitive algorithm to training data sizes. The algorithms are characterized on a two-dimension chart based on prediction performance and computation time. This chart is expected to guide a novice user to choose an appropriate method for his/her demand. Based on this chart, in general, Bagging and Random Forest are the two most recommended algorithms because of their high performance and speed.

Keywords: Supervised learning algorithms, Evaluation metrics, Performance comparison

1. Introduction

Machine learning has been developed to solve a wide range of problems, including data clustering and event prediction. Given the variety of algorithms, a novice user may need some guidance to find the best algorithm to suit a particular task. Supervised machine learning is an umbrella term for various machine learning algorithms designed for prediction tasks. In addition to obvious applications, machine learning has been used for prediction in several systems, such as image processing [1], text classification [2], object recognition [3], keyword extraction for scientific literature [4], and foreign exchange rate forecasting [5]. In supervised learning, historical data is used to build a prediction model that generally has several model parameters. The historical data of the underlying system is employed to find the best values for the model parameters. Using historical data to determine values for model parameters is called "training" or "learning". The historical data used for this matter is called "training data". Given a proper model and that it is well trained with historical data of the same underlying system, a reasonably accurate prediction should be made.

A prediction model with "learned" parameters is often called a "learned model" or a "trained model". Once a trained model is available, it can be used to make a prediction. The accuracy of the prediction model is tested on another set of data. This second set of data is of the same underlying system, but does not contain any data record of the training data. This second set is called "test data."

As an important part in the process of building a prediction model, the quality of training data correlates to prediction accuracy. A model with high prediction accuracy is preferable, thus high quality training data is essential. Training data is of high quality if it can well represent the underlying population. That is, the training data size is sufficiently large and shows the same distribution as the population of interest. However, in practice, training data size may be limited by cost, time, and the resources required for data acquisition. Additionally, available historical data may have some missing attributes. This could deteriorate the performance of a prediction model. Therefore, issues of training data sizes and missing attributes are of practical concern. These issues are factors investigated in our study. Additionally, prediction time may be important in some situations. For example, some cases may require that prediction is performed several times. Others may have prediction integrated into a large real-time system, whose critical response time greatly depends on prediction time, which is included in our evaluation metrics.

Previous studies, Caruana et al. [6], Caruana et al. [7], Kotsiantis et al. [8], and Dietterich [9] compared machine learning algorithms. They investigated many issues and aspects, such as high dimensionality, various evaluation

*Corresponding author.
Email address: tatpong@kku.ac.th
doi: 10.14456/easr.2018.28

metrics, and a number of application domains. Caruana et al. [6] compared 11 machine learning algorithms using multiple performance metrics. Caruana et al. [7] investigated the issue of high dimensionality. Dietterich [9] focused on three ensemble methods. Ahmed et al. [10] studied the issues using time series data. Most previous studies found that Bagging, Boosting, and Random Forest were the best performing models. However, there has not been any previous study investigating issues of various training data sizes and missing attributes.

Our study investigated issues of training data sizes and missing attributes on how seven widely used classification algorithms performed in various application domains. This article used the seven standard UCI benchmark data sets for testing and two metrics as performance criteria. The rest of the article is organized as the following: Section 2 describes background knowledge about studied algorithms. Section 3 discusses related work. Methodology and algorithm implementation are presented in Section 4. In Sections 5 and 6, experimental settings are described and experimental results are analyzed. Finally, our work is concluded in Section 7.

2. Background

2.1 Supervised machine learning algorithms

Supervised machine learning, or shortly supervised learning, is a machine learning approach to build prediction models. In a prediction task, given input (a set of attributes) *x*, the task is to predict output *y*. Prediction output may be a continuous value. Such a case is called a regression problem. When prediction output can take only a few discrete values, we call it a classification problem. Our study focuses on classification. A classification model is also called a classifier. A discrete value of a classification output is often called a "class" or a "label". A classifier usually has model parameters. Model parameters govern how the classifier relates the input x to the output y. Therefore, training data is sometimes called labelled data. There are several classification algorithms such as K-NN, Decision Tree, SVM, Random Forest, and Naive Bayes.

Generally, a classifier is a mapping function, $f: x \xrightarrow{\theta} \mathrm{y}$, where θ is a set of model parameters. Supervised learning aims to determine values of parameters by using a training data set or examples, $D = \{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \dots, \{x_N, y_N\}\}$, where $\{x_n, y_n\}, n = 1, \dots, N$ is the n[th] example data point consisting of input x_n and output y_n. The parameter values are determined by solving an optimization problem, $\theta^* = \underset{\theta}{\operatorname{argmax}} V(f, D)$, where V is a fitness function measuring correctness of prediction using model f on data D. After model f maps example input x_n to $\hat{y}_n$ for all $i's$, the prediction $\hat{y}_n's$ are compared to the example outputs $y_n's$. The more $\widehat{y_n}'s$ match $y_n's$, the higher value of V. How model f is specified, how the optimization is solved and how the fitness function V is defined depending on the specific algorithm.

Decision Tree is a machine learning algorithm that creates a logical tree from training data, and uses the tree for classification. Generally, Decision Tree predicts a binary output $y \in \{+1, -1\}$ of input $x \in R^d$ by employing multiple rule-based binary functions cascading in a tree-like structure. Each rule-based function can be called a "node."

Given a set of attribute indices $\Omega = \{1,2,3, \dots, \mathrm{d}\}, \mathrm{d} > 1$, a node is defined as:

$$b_\Omega(x) = \begin{cases} b^+_{\Omega-\{i\}}(x), x_i \geq \tau; \\ b^-_{\Omega-\{i\}}(x), x_i < \tau, \end{cases} \quad (1)$$

where x_i is the i^{th} attribute of x, τ and i are node parameters, $b^+_{\Omega-\{i\}}(x)$ and $b^-_{\Omega-\{i\}}(x)$ are other nodes. A leaf node is defined as a node with a specific value, $B_\Omega(x) \in \{+1, -1\}$. It should be noted that a node with an empty index set must be a leaf node, $b_\phi(x) = B_\phi(x)$.

A classifier of Decision Tree maps input x to binary labels, $\{+1, -1\}$, starting at a node function with a full set of attribute indices, $\{1,2,3, \dots, d\}$, and successively applying subsequent node functions until reaching a final value at the leaf node. Values of node parameters are determined to maximize a fitness function by means of optimizing information gain. How values of node parameters are specifically determined and how model f is exactly defined depend on the specific implementation. Decision Tree has been developed in several versions such as ID3, C4.5, and CART. Scikit-learn uses CART algorithm to implement Decision Tree.

K-NN is a classifier whose output is determined from labels of K similar examples, where K is a user specific meta-parameter. A similar example is called a neighbor. K-NN predicts a label of datapoint x to be a label that most of its K nearest neighbors have. The K nearest neighbors are K examples whose attributes have the shortest distances to attributes of datapoint x. Euclidean distance can be used to measure the distance between a datapoint in question x and an example x_i. Figure 1 illustrates how K-NN with K = 4 determines a label datapoint x. Here, given a 2-dimensional problem (each datapoint has two attributes), datapoint x and examples $x_i's$ can be plotted on a 2-dimension plane. The cross represents datapoint x. Solid circles and squares represent two distinct labels of previous data, $x_i's$. The location of each mark represents values of its corresponding attributes. In the figure, four nearest neighbors are enclosed within a dashed circle. Since three out of four nearest neighbors have a square-class label, K-NN will predict a square-class label for datapoint x.

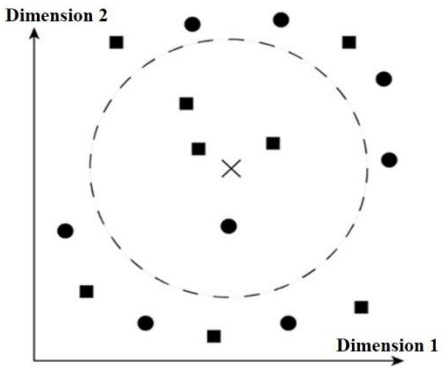

Figure 1 K-NN algorithm where *k* is 4 [11]

The Ensemble method is a widely used supervised machine learning approach [12]. It is used to create, in this case, a classifier by combining results from many base classifiers. Classifiers, such as Decision Tree, K-NN, SVM, can be used as base classifiers. Given k base classifiers, each classifier, C_i, is trained with samples, D_i, from the entire training data set, D. The output of an ensemble classifier is a label that gets most of the votes from all base classifiers. Equation 2 [11] shows an ensemble output,

$$C^* = Vote\big(C_1(D_1), C_2(D_2), \dots, C_k(D_k)\big) \quad (2)$$

where $C_i(D_i)$ represents an output of base classifier C_i trained with data D_i. $Vote()$ is a vote function whose output is a label that appears the most among outputs of base classifiers. A straightforward ensemble method is Bagging. Algorithm 1 [11] displays pseudo code of the Bagging algorithm. In the first step, *k* base classifiers are selected. The next step selects a sample D_i with replacement. In line 4, base classifier C_i is created based on data D_i. After we have created k base classifiers, $C_1, C_2, \dots, C_k$, this completes the training process. Then, the trained Bagging model can make a prediction (line 6) by identifying a predicted label with which most of base classifiers agree. It should be noted that line 6 is equivalent to equation 2. Variable x represents a set of attributes of an input under question. Variable y represents a label to be compared with an output of a base classifier.

Algorithm 1 Pseudo code of Bagging [11]

1: let k be the number of sample
2: for each do
3: create a sample D_i.
4: create a base classifier C_i on the sample D_i.
5: end for
6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta(C_i(x) = y)$ #prediction output
7: {δ() = 1 if its argument is true and 0 otherwise.}

Boosting is an ensemble approach, whose final prediction is made by combining predictions from base classifiers. However, unlike Bagging, in which each base classifier is trained on samples drawn with equal probability, the Boosting approach has a mechanism to focus on difficult examples. Boosting has been studied extensively. Various Boosting methods employ different mechanisms and schemes to combine base classifiers. AdaBoost [13] is a widely used Boosting algorithm. Algorithm 2 shows the pseudo code of Adaboost. A subscript, e.g., C_i, D_i, denotes an index of a successive round. A superscript may be omitted when the context is trivial. Given training data,$\{(x_i, y_j)\}, j = 1, \dots, N$, {(xj , yj)}, AdaBoost uses weight, w_j, to quantify how difficult an example (x_i, y_j) is. The algorithm first initializes all weights to be equal (line 1), and then successively adjusts the weights (line 12). Each weight w_j is adjusted based on a performance error. Performance error, line 7, is an approximate probability that a base classifier would incorrectly classify the data. As weights are successively updated, each set of weights is used in a process of sampling training set D_j (line 4). That is with a large corresponding example (x_m, y_m) with a smaller weight, w_m. Therefore, classifier C_i, which is trained on data set D_i, would be likely to work better on complicated examples than its previous classifier C_{i-1}. In making final output, AdaBoost uses a voting score, α_i, to credit base classifier C_i for its performance. A good performing classifier C_i (indicated by low ε_i) gets a large voting score, α_i, and vice versa. The output is a predicted label that receives the highest total score (line 15).

Random Forest is similar to a Bagging approach but it is designed to have Decision Trees as base classifiers. Similar to Bagging, to create a base classifier C_i, a sample D_i is taken from data D. Not only does Random Forest create sample D_i from randomly selected data points, but it also randomly selects input features to include for each sample D_i [11].

Algorithm 2 Pseudo code of Adaboost [11]

1: $\omega^{(1)} = \omega_j = \frac{1}{N} | j = 1,2, \dots, N$ # N initialize the weights for all N examples.
2: let k be the number of Boosting rounds.
3: for i=1to k do
4: create a training set D_i by sampling (with replacement) from D according to $\omega^{(i)}$.
5: train a base classifier C_i on D_i.
6: apply C_i to all examples in the original training set D.
7: $\mathcal{E}_i = \frac{1}{N}[\sum_j \omega_j \delta(C_i(x_j) \neq y_i)]$ #calculate the weighted error.
8: if $\mathcal{E}_i > 0.5$ then
9: reset weights and go to re-sampling (line 4).
10: end if
11: $\alpha_i = \frac{1}{2} \ln \frac{1-\varepsilon_i}{\varepsilon_i}$
12: update weight by $\omega_j^{(i+1)} = \frac{\omega_j^{(i)}}{z_i} x \begin{cases} e^{-\alpha_i}, C_i(X_j) = y_i \\ e^{-\alpha_i}, C_i(X_j) \neq y_i \end{cases}$,
13: where $Z_i$is a factor to normalize weights so that $\sum_{j=1}^{N} \omega_j^{(i+1)} = 1$.
14: end for
15: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \alpha_i \delta(C_i(x) = y)$ #prediction output

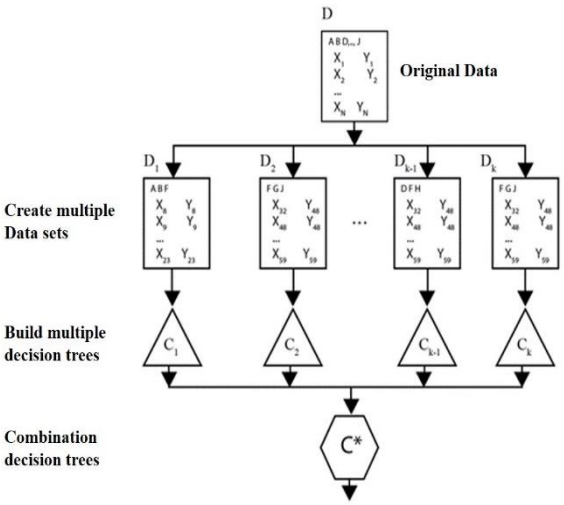

Figure 2 Graphical depiction of the Random Forest algorithm

Figure 2 illustrates a Random Forest procedure. Training data D with all features, A, B, C, J, is sampled to create subsets $D_1, \dots, D_k$. Sample D_1 contains data points, $(x_8, y_8), (x_9, y_9), \dots, (x_{23}, y_{23})$. Additionally, sample D_1 has only features, A, B, and F. Other samples may contain other data points and a different set of features. Therefore, each Decision Tree is trained on different samples and expected to be different base classifiers. Then, the output of Random Forest is combined from outputs of all Decision Trees.

Naive Bayes is a probabilistic classifier based on Bayes theorem. Given input features, Naive Bayes predicts a class with the highest estimated probability. Naive Bayes output can be written as shown in equation (3),

$$C(x) = \underset{c}{\operatorname{argmax}} P(y = c|x) \quad (3)$$

where $P(y = c|x)$ is a probability that an output class is c, given the data has input the posterior probability, $P(y|x)$, can be obtained using Bayes theorem. That is:

$$p(y|x) = \frac{p(y) \cdot p(x|y)}{p(x)} \quad (4)$$

where $p(y)$ is a prior probability of output class. Term $p(x|y)$ represents the conditional probability of input x, given class y. The denominator term $p(x)$ is a probability of input, which determines a proper class of input x. The comparison of multiple events are also divided by $p(x)$. Hence, it is reasonable to omit the denominator as follows:

$$p(y|x) \propto p(y) \cdot p(x|y) \quad (5)$$

Naive Bayes naively estimates the probability $p(x|y)$ by assuming the features of input $x = \{f_1, f_2, \dots, f_d\}$ are independent. Then, expression 6 can be written as:

$$p(x|y) \propto p(y) \prod_{i=1}^{d} p(f_i|y) \quad (6)$$

where $p(f_i|y)$ is a probability of input feature f_i, given class y. Naive Bayes estimates the probability $p(y)$ by the fraction of class y. For a categorical feature f_i, the probability $p(f_i|y)$ is estimated by the fraction of all datapoints in class y. For a continuous feature f_i, the probability $p(f_i|y)$ can be estimated based on a Gaussian distribution.

Support Vector Machine (SVM) [14] is a widely used classification technique. First, data is transformed into a kernel space such that the transformed instances would be easily classified (linearly separable). Then, SVM will find a decision boundary that maximizes margin between instances of two different classes. Figure 3 illustrates how SVM finds a decision boundary in a kernel space. A decision boundary will divide (transform) data into two classes. Given the training data, there may be many possible decision boundaries that can divide transformed instances into correct classes. SVM will select the boundary that has the maximum margin (the distance between each nearest instance of different classes). The line B1 in Figure 3 allows a larger margin than any other line does. Therefore, SVM will select B1 as the decision boundary. Once the decision boundary is determined, any new instance on the left side of B1 will be classified as class square. Any instance on the right side of B1 will be classified as a class circle.

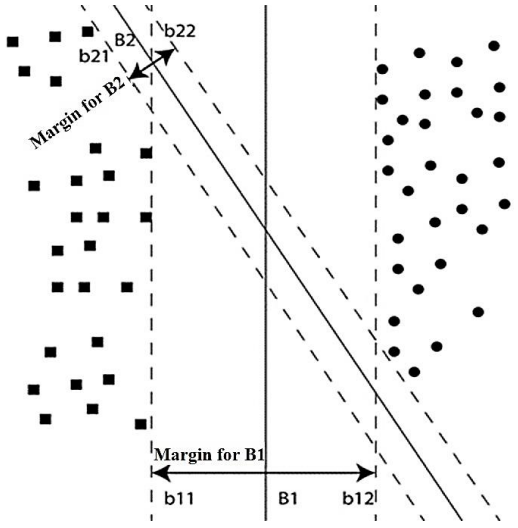

Figure 3 How a SVM determines a decision boundary (this figure is modified from Pang et al. [11]'s Figure 5.25.)

2.2 Evaluation metrics

Our study employs two metrics: accuracy and an F1-score. These two metrics are widely accepted in various domains, including information retrieval. Accuracy (ACC) measures how well a classifier predicts in a straightforward manner, as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

The terms TP, TN, FP, and FN stand for true positive, true negative, false positive, and false negative, respectively. For binary classification, true positive is a number of instances that a classifier predicts positive labels and they are correct. True negative is a number of instances that a classifier correctly predicts negative labels. False positive is a number of instances that a classifier predicts positive labels, but they are incorrect. False negative is a number of instances that a classifier incorrectly predicts negative labels. The chosen second metric is F1-score (FSC), which is a performance measurement for classification. FSC is formulated in such a way that performance for a skewed data set is well represented. FSC can be computed as follows:

$$F1 = 2 \times \left(\frac{precision \times recall}{precision + recall}\right) \quad (8)$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

3. Related work

Many researchers have compared the performance of machine learning algorithms. They all compared algorithms by separating data sets in two parts, training data and test data. They attempted to find the best model using k-fold cross validation in the training data.

The STATLOG [15] project compared seventeen machine learning algorithms and used twelve data sets in 1995. They randomly split data into training and test sets, but did not vary training data sizes.

LeCun [16] compared machine learning algorithms for handwritten digit recognition. It was developed by Bell Laboratories. They used data sets from NISTs Special Database 1 and Special Database 3. They considered accuracy, training time, recognition time, and memory requirements and found that Boosting LeNet 4 had the best performance.

Caruana and Niculescu-Mizil [6] made a comparison of new supervised machine learning algorithms, such as Bagging, Boosting, SVM, and Random Forest. They used 11 data sets and eight metrics for comparison with the fixed size of 5,000 training data records. They studied only performance, but article investigates the computation time.

Caruana et al. [7] compared supervised machine learning algorithms. They compared data which had high dimensionality (750-700K dimensions). They used 11 data sets which had data dimensionality between 761 and 685,569. Three metrics and ten supervised machine learning algorithms were used for comparison. The best overall performance was for Random Forest while the worst performing was Naive Bayes. A comparison metric used in their work was the high dimensionality of the data. They split the data into two equal parts, training and testing. The comparison metrics in our work are training data sizes and the number of missing attributes.

Bauer et al. [17] compared supervised ensemble machine learning algorithms. They used 20 ensemble machine learning algorithms, 19 data sets from the UCI benchmark, and mean-squared errors for comparison. Random Forest was the best performing model overall.

Sood et al. [18] compared two functions of neural networks including Back propagation multilayer perceptron (MLPNN) and radial basic function (RBF) in Electroencephalogram (EEG). They varied the number of input layers, output layers, and hidden layers. MLPNN performed better than RBF as MLPNN had 99.6% accuracy while RBF had 96.8% accuracy.

Fernandez-Delgado et al. [19] compared 179 machine learning classifiers from 17 families and used 121 data sets. Random Forest was the best algorithm. They compared this algorithm using many tools, such as Weka, R, Matlab, LibSVM, and SVMLight.

Omidiora [20] compared four machine learning classifiers using two data sets. They used 1,000 images for training data and test data. Four machine learning algorithms were compared, Naive Bayes, Instance Based Learner, Decision Tree, and Neural Network. They found that Instance Base Learner was the best performance because it yielded the highest accuracy.

El-Halees [9] compared six supervised machine learning algorithms which include maximum entropy, Decision Tree, Naive Bayes, Artificial Neural Nets, Support Vector Machine, and K-Nearest Neighbours. They used these algorithms to filter spam e-mails mixing Arabic and English text. Stemming and feature selection before classification yielded better results. Stemming and feature selection are preprocessing steps before machine learning methods are applied. Stemming is the process that removes punctuation, stop words, and non-letter characters and that converts a word to its root by using the service at http://www.qamuns.org. They found that Support Vector Machine had the best performance with a 98.32% F1-measure.

All of the above research used fixed data set sizes and had no missing values. A training data size was usually fixed to a specific value, from 50% to 70% of total available data. This research compared supervised learning algorithms with various training data sizes and missing attributes.

4. Methodology

Regarding algorithm implementation, Scikit-learn [21] was used for the implementation of Boosting, Bagging, Naive Bayes, K-Nearest Neighbours (K-NN), Decision Tree, and Random Forest. LibSVM [22] was used for implementation of SVM. Scikit-learn is a Python library for various machine learning algorithms while LibSVM is a widely used SVM library. Although Scikit-learn has SVM functionality, based on our experimental tests, its response was very slow. Thus, for studying SVM algorithms, this work used LibSVM rather than Scikit-learn.

This study evaluated seven classification algorithms, Bagging, Boosting, Random Forest, Decision Tree, Naive Bayes, K-NN, and SVM. It determined their accuracy, F1-score, and required computational time using seven different data sets from the UCI repository. The data sets were selected to cover various domains, i.e., medicine (heart disease), image analysis (image segmentation, Landsat satellite images, vehicle silhouettes, letter recognition), and finance (Australian credit approval, German credit approval). Meta-information of each data set is shown in Table 1.

Table 1 Meta-information of each data set

Data sets name	Number of instances	Number of features
Heart Disease	303	14
Letter Recognition	20,000	16
Australian Credit Approval	690	14
German Credit Data	1,000	20
Landsat Satellite	6,435	36
Image Segmentation	2,310	19
Vehicle Silhouettes	946	18

The heart disease data set contains information about heart disease diagnoses from the Cleveland Clinic Foundation [23]. The letter recognition data set [24] contains numerical attributes (statistical moments and edge counts) of images for each of the 26 letters of the English alphabet and their corresponding labels. The Australian credit approval data set [25] contains attributes about credit card applications and approval results. The German credit data set [25] contains information about bank customers and their corresponding credit risks. The Landsat satellite data set contains attributes of satellite imagery and corresponding soil types [25]. The image segmentation data set [25] contains numerical attributes of images, each displaying one of seven objects, e.g., brick face, sky, foliage, and the corresponding object labels. The vehicle silhouettes data set [26] contains attributes, e.g., hollows and compactness, obtained from a Hierarchical Image Processing System (HIPS). The task is to classify silhouettes into one of four types of vehicles: Opel, Saab, bus, and van.

In our classification task, we followed the approach of Caruana et al. [6] of turning all multi-class tasks into binary classifications. Among the seven data sets, the letter recognition, Landsat satellite, image segmentation, and vehicle silhouettes are multi-class classification in nature. The heart disease data set output values are {0, 1, 2, 3, 4} while the value of 0 represents absence of the disease. Other values simply mean presence of the disease at different levels of severity. Turning the heart disease data set to a binary classification is straightforward. Other multi-class data sets were turned into binary classification by grouping multiple classes into two super-classes. We arbitrarily selected classes for each super-class. For the letter recognition data set, we converted the letters, a through m, into positive classes and the remainder letters into negative classes. For Landsat satellite data set, we converted land types 1 (red soil), 2 (cotton crop), and 3 (grey soil) to positive classes and the remainder to negative classes. For the image segmentation data set, we converted the image labels, brick face, sky, and foliage into positive classes and the remainder to negative classes.

Figure 4 displays the system overview. For each algorithm, meta-parameters were selected using 5-fold cross validation. Each model was trained and then evaluated on a test data set by randomly using 25% of a data set. In the data preparation process, two issues were investigated, various training data sizes and absence of attributes. With respect to training data sizes (dimension 1), a classifier was examined when trained on 75%, 50%, and 25% of the data set. Studying the second issue (dimension 2), a classifier was examined when trained on a complete set of attributes and when trained on a set of attributes without the three most relevant attributes. ANOVA was used to identify these three attributes in each data set. Each treatment was repeated 50 times. The performance of each treatment was measured for accuracy and F1-score.

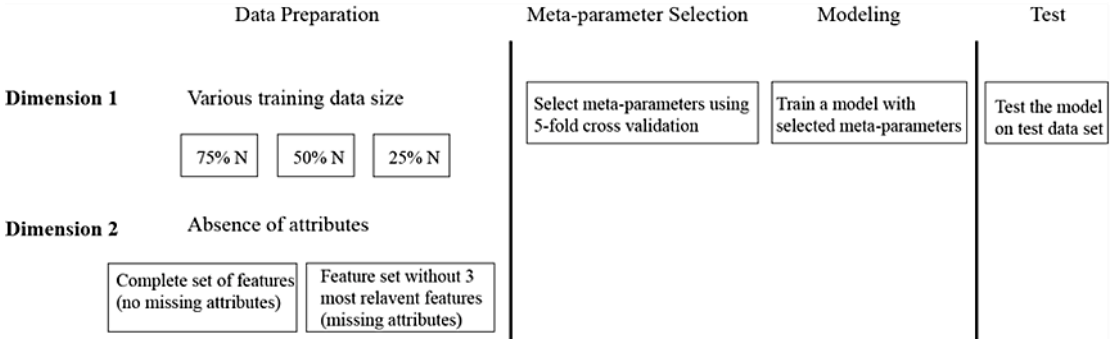

Figure 4 System Overview

Table 2 Parameter Values

Parameters	Values set up in our experiment
Bagging: Number of base classifiers, k (in pseudo code of Algorithm 1.)	2, 4, 6, 8, 32, 64, 128, 256, 1024
Boosting: Number of base classifiers, k (in pseudo code of Algorithm 2.)	2, 4, 6, 8, 32, 64, 128, 256, 1024
Random Forest: Number of trees, k (in Figure 2)	2, 4, 6, 8, 32, 64, 128, 256, 1024
K-NN: Number of neighbours, k (in Figure 1)	2, 3, 4,…, max
SVM: Kernel type	Linear, degree-2 polynomial, degree-3 polynomial, radial basis, sigmoid

Table 3 Best Performing Algorithms on Each Dataset with Various Data Sizes

Data sets	25%	50%	75%
Heart Disease	Naive Bayes (0.755)	Naive Bayes (0.776)	Naive Bayes (0.794)
Letter Recognition	SVM (0.951)	SVM (0.967)	SVM (0.973)
Australian Credit Approval	SVM (0.842)	Random Forest (0.851)	Bagging (0.856)
German Credit Data	Naive Bayes (0.507)	Naive Bayes (0.557)	Naive Bayes (0.565)
Landsat Satellite Images	K-NN (0.939)	K-NN (0.945)	K-NN (0.949)
Image Segmentation	Random Forest (0.945)	Random Forest (0.965)	Random Forest (0.974)
Vehicle Silhouettes	SVM (0.941)	SVM (0.961)	SVM (0.969)

Parameter values for each algorithm were chosen based on training set evaluation results. The parameters yielding the best performance were used in the algorithms. Their values are shown in Table 2.

Table 2 shows parameter values evaluated for each algorithm. Decision Tree and Naive Bayes did not have any user specified parameters. Bagging, Boosting, Random Forest, and K-NN each had only one user specified parameter. Decision Tree was used to base the classifiers of Bagging and Boosting following a Scikit-learn specification. K-NN was evaluated with parameter k from k = 2 to the maximal value that the available data allowed. For example, letter data set had 2,000 instances. Its training set had 1,500 instances (75% of the total). Using 5-fold cross validation, this left us with 1,200 instances available ($=\frac{4\times1,500}{5}$= 1,200). Therefore, K-NN was evaluated with k = 2 to k = 1,200. For SVM, the algorithm was evaluated with five different kernel types, as shown in the table. Other SVM parameter values were set to the default values that LibSVM provided.

The parameters were used in each algorithm as follows. The k parameter values of 64, 256, 512, and 23 were used with Bagging, Boosting, Random Forest, and K-NN, respectively. A Linear kernel were used with SVM.

5. Experimental results and discussion

Table 3 shows the machine learning algorithms that perform the best in each data set for all training sizes. For example, SVM performed the best in vehicle silhouettes data for 25%, 50%, and 75% of training data sizes.

Table 4 summarizes performance measures, accuracy and F1-score, of each classifier with various training data sizes. Columns 25%, 50%, and 75% indicate setting when the corresponding percentage of data is used for training. Column average indicate an average measure over the three settings. Each entry represents average performance measure of the corresponding classifier over seven data sets, each with the same setting. For example, the accuracy of Random Forest trained on 25% data size was 0.874. That is the average accuracy of applying Random Forest over seven data sets. Each data set had the same setting. In this case, 25% of that data set was for training.

Table 5 shows percentage relative difference of accuracy and F1-scores when training data size decreases. Column 25% ←50% indicates percentage of relative difference of performance measures when training data size decreases from 50% to 25%. Column 50% ←75% is defined in a similar manner. A larger number indicates that decreasing training data size has a larger negative effect on performance of the algorithm.

From our experimental results, Random Forest provided the most accurate results across different sizes of training data (Table 5). Decreasing training data sizes negatively affects the performance of all algorithms. Decreasing training data sizes from 75% to 50% showed less negative effects than decreasing from 50% to 25% (Table 5). That was

Table 4 Overall of Accuracy and F1-score

	Accuracy				F1-score			
Classifier	**25%**	**50%**	**75%**	**Average**	**25%**	**50%**	**75%**	**Average**
Random Forest	0.874	0.892	0.899	**0.888***	0.816	0.840	0.851	**0.836***
Bagging	0.870	0.890	0.899	**0.887***	0.810	0.838	0.850	**0.833***
SVM	0.863	0.881	0.884	**0.876***	0.820	0.839	0.843	**0.834***
Boosting	0.844	0.858	0.865	0.855	0.793	0.814	0.823	0.810
Decision Tree	0.822	0.838	0.852	0.837	0.778	0.797	0.812	0.796
K-NN	0.792	0.812	0.823	0.809	0.670	0.704	0.722	0.699
Naive Bayes	0.737	0.759	0.764	0.753	0.727	0.738	0.742	0.736

Table 5 Differences in Accuracy and F1-scores

	Performance difference			
	Accuracy		F1-score	
Classifier	**25% ←50%**	**50% ←75%**	**25% ←50%**	**50% ←75%**
Random Forest	-0.018	-0.008	-0.024	-0.011
Bagging	-0.020	-0.009	-0.028	-0.012
SVM	-0.019	**-0.003**	**-0.018**	**-0.004**
Boosting	**-0.014**	-0.007	-0.021	-0.010
Decision Tree	**-0.016**	-0.013	-0.018	-0.015
K-NN	-0.020	-0.012	-0.035	-0.018
Naive Bayes	-0.021	**-0.006**	**-0.011**	**-0.004**

Table 6 Performance of Naive Bayes on Each Data Set with Various Data Sizes

	25%		50%		75%	
Data set	**ACC**	**FSC**	**ACC**	**FSC**	**ACC**	**FSC**
Heart Disease	0.789	0.755	0.808	0.776	0.825	0.794
Letter Recognition	0.710	0.711	0.712	0.713	0.712	0.714
Australian Credit Approval	0.839	0.813	0.839	0.811	0.840	0.812
German Credit Data	0.567	0.507	0.695	0.557	0.718	0.565
Landsat Satellite	0.878	0.863	0.878	0.863	0.878	0.863
Image Segmentation	0.627	0.680	0.627	0.683	0.628	0.685
Vehicle Silhouettes	0.751	0.760	0.752	0.762	0.750	0.762

as expected, because the training data size was relatively small. Inherent information carried in each data point would become more crucial. Regarding robustness against decreasing training data size, K-NN had distinctively larger F1-score differences than others.

This implies that K-NN is the most sensitive algorithm to training data size. Therefore, K-NN is not recommended when there is little training data available. Naive Bayes and SVM are more robust than other algorithms, as their F1-score difference was relatively smaller.

Close investigation of both SVM and Naive Bayes reveals a strikingly consistent performance of Naive Bayes across a wide range of data sizes. Table 6 shows that Naive Bayes has relatively constant performance across training data sizes in some data sets. This contradicts our first expectation that decreasing training data size to 25% would severely affect classification performance. Experimental results of Naive Bayes were examined for each data set to investigate the issue. Naive Bayes uses training data to determine parameters, means (μ's) and variances (σ's) of each continuous feature. Then, those parameters are used to estimate the probabilities that the data point in question belongs to each class. Downsizing a training data set with random sampling slightly affects the means and variances of the smaller training data set. It slightly affected the behavior and performance of the Naive Bayes model.

In Figure 5, the computational time shows normalized time to classify the data used by each algorithm. Normalized time is computed by averaging times required by each algorithm over all experiments and then normalizing with the longest average time. With normalized time of 0.001 and 0.004, the Decision Tree and Naive Bayes algorithms are the two fastest algorithms. SVM requires a large amount of time for prediction. Ensemble methods (Bagging, Boosting, and Random Forest) and K-NN requires relatively similar computational times.

Computation time presented here is the time required for prediction. Time spent in prediction is more important than time spent in model building. Generally, the model is created only while then prediction will be made multiple times.

All experiments were conducted on Ubuntu 10.04 server, IntelXeon(R) CPU ES2640@2.5GHz x 12 with RAM 16GB.

Figure 6 charts each algorithm based on prediction accuracy and time. The chart is divided into four zones:
(1) slow and inaccurate zone,
(2) slow and accurate zone,
(3) fast and inaccurate zone, and
(4) fast and accurate zone.

For illustration purposes, zone boundaries, as shown as dash lines in Figure 6, were assigned arbitrarily. None of the algorithms is in the slow and inaccurate zone. Naive Bayes and K-NN algorithms are in the fast and inaccurate zone. SVM is in the slow and accurate zone. All ensemble methods and Decision Tree algorithms are in the fast and accurate zone, which is the most desirable.

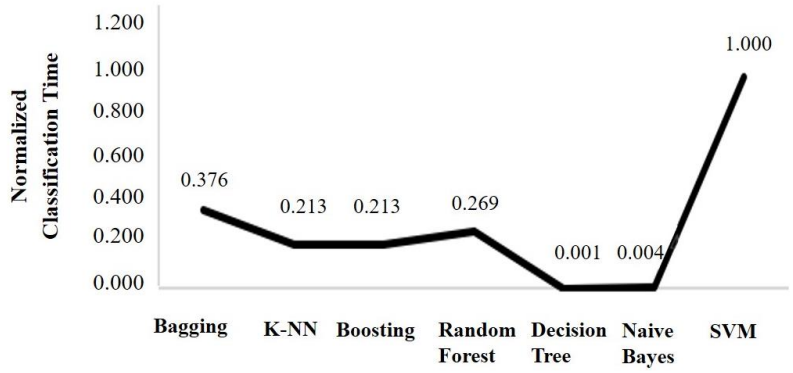

Figure 5 Computation Time

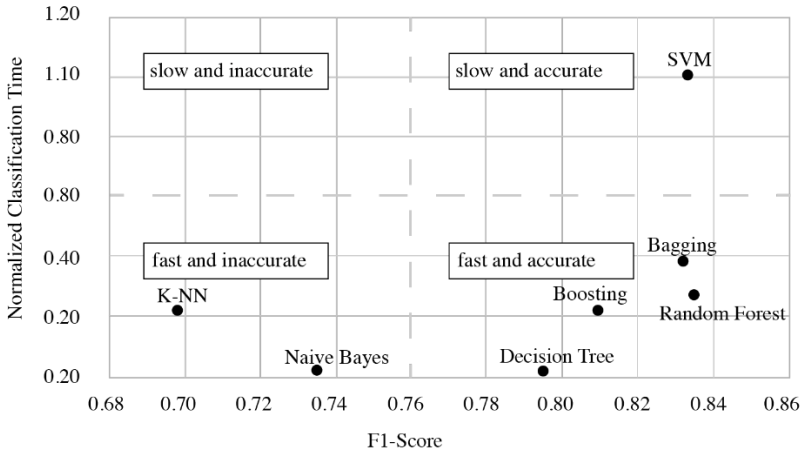

Figure 6 Comparison of prediction time and accuracy of machine learning methods

Table 7 Performance Degradation When Three Key Attributes Were Missing[1].

	Normal		Missing Values		Performance Degradation			
Machine learning	**ACC**	**FSC**	**ACC**	**FSC**	**ACC**	**FSC**	**%ACC Degradation**	**%FSC Degradation**
Random Forest	0.888	0.836	0.850	0.777	0.038	0.059	4.318%	7.091%
Bagging	0.887	0.833	0.850	0.777	0.037	0.055	4.154%	6.638%
SVM	0.876	0.834	0.818	0.742	0.058	0.092	6.566%	11.018%
Boosting	0.855	0.810	0.811	0.737	0.044	0.073	5.160%	8.973%
Decision tree	0.837	0.796	0.791	0.742	0.046	0.053	5.539%	6.702%
K-NN	0.809	0.699	0.786	0.659	0.023	0.039	**2.878%**	**5.614%**
Naive Bayes	0.753	0.736	0.708	0.673	0.045	0.063	6.000%	8.531%

Note: [1]ACC is Accuracy. FSC is F1-score.

6. Missing attributes

The three most relevant attributes were removed based on the ANOVA F-value. Such relevant attributes are obtained using SelectKBest function of Scikit-learn. In Table 7, ACC and FSC are the average values for accuracy and F1-score. The values in the "Normal" column are the values of ACC and FSC when no attributes were removed. "Missing Values" column shows the average ACC and FSC values when some attributes were removed. The last column, "Performance Degradation", is the percentage of difference between "Normal" column and "Missing Values" column.

Regarding the issue of missing attributes, all algorithms show decreasing performance when key attributes are missing. A relatively high percentage of degradation (last two columns in Table 7) implies that K-NN suffers the least when key attributes are missing.

Based on classification performance, computational time required, and robustness against decreasing training data size and missing key attributes, Random Forest is recommended for general classification. Additionally, Random Forest has a strong potential of scaling up for highly dimensional data. When building base classifiers, not only are data points sampled, but a set of features are also randomly selected. This mechanism would allow Random Forest to tackle high dimensional data with relatively constant computational time. It should be noted that the Bagging method outperforms the Boosting method overall, as shown in Table 4 and when there are missing attributes, as shown in Table 7.

7. Conclusions

This work evaluated machine learning algorithms using Scikit-learn and LibSVM. Overall, Random Forest and Bagging are the two best performing methods. Both methods have high accuracy and robustness. SVM also delivers a similar level of accuracy to Random Forest and Bagging, but SVM performance drops significantly when key attributes are missing. Therefore, SVM should not be used when completeness of attribute values is a concern. Additionally, SVM requires a substantially longer time than other methods. Decision Tree and Naive Bayes are the two fastest methods, but they are not as accurate or robust. In order to provide an easy guideline, the algorithms are characterized into four groups, which are (1) slow and inaccurate, (2) slow and accurate, (3) fast and inaccurate, (4) fast and accurate groups. All ensemble methods (Bagging, Boosting, and Random Forest) are in the fast and accurate group. In order to study the aspect of robustness, issues of various training data sizes and absence of three key attributes are investigated. The performance of every algorithm decreases when the training data size decreases or there are missing attributes.

Nevertheless, K-NN is affected the least when there are missing attributes. The prediction accuracy deterioration due to decreasing training data size is more severe when training data is already small. K-NN is particularly sensitive to decreasing training data size and therefore not recommended when training data size is an issue. Naive Bayes is somewhat insensitive to decreasing training data size. Therefore, when using Naive Bayes, the training data size is not an issue.

8. References

[1] Marée R, Geurts P, Visimberga G, Piater J, Wehenkel L. A comparison of generic machine learning algorithms for image classification. In: Coenen F, Preece A, Macintosh A, editors. The twenty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence; 2003 Dec 15-17; Cambridge, UK. London: Springer; 2004. p. 169-82.

[2] Maruf S, Javed K, Babri H. Improving text classification performance with random forests-based feature selection. Arab J Sci Eng Springer Sci Bus Media BV. 2016;41(3):951-64.

[3] Peralta B, Caro LA. Improved object recognition with decision trees using subspace clustering. J Adv Comput Intell Intell Inform. 2016;20(1):41-8.

[4] Wu C, Marchese M, Jiang J, Ivanyukovich A, Liang Y. Machine learning-based keywords extraction for scientific literature. J Univers Comput Sci. 2007;13(10):1471-83.

[5] Rout AK, Dash PK. Forecasting foreign exchange rates using hybrid functional link RBF neural network and Levenberg-Marquardt learning algorithm. Intell Decis Technol. 2016;10(3):299-313.

[6] Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms. Proceedings of the 23rd international conference on machine learning; 2006 Jun 25-29; Pennsylvania, USA. USA: ACM; 2006. p. 161-8.

[7] Caruana R, Karampatziakis N, Yessenalina A. An empirical evaluation of supervised learning in high dimensions. Proceedings of the 25th international conference on Machine learning; 2008 Jul 5-9; Helsinki, Finland. USA: ACM; 2008. p. 96-103.

[8] Kotsiantis SB. Supervised machine learning: a review of classification techniques. Informatica. 2007;31:249-68.

[9] El-Halees A. Filtering spam e-mail from mixed arabic and english messages: a comparison of machine learning techniques. Int Arab J Inf Technol. 2009;6(1):52-9.

[10] Ahmed NK, Atiya AF, Gayar NE, El-Shishiny H. An empirical comparison of machine learning models for time series forecasting. Econom Rev. 2010;29(5-6):594-621.

[11] Tan PN. Introduction to data mining. India: Pearson Education India; 2006.

[12] Lughofer E, Kazienko P. Hybrid and ensemble methods in machine learning. J Univers Comput Sci. 2013;19(4):457-61.

[13] Freund Y, Schapire RE. A desicion-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci Int. 1997;55:119-39.

[14] Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273-97.

[15] King RD, Feng C, Sutherland A. Statlog: comparison of classification algorithms on large real-world problems. Appl Artif Intell Int J. 1995;9(3):289-333.

[16] LeCun Y, Jackel LD, Bottou L, Brunot A, Cortes C, Denker JS, et al. Comparison of learning algorithms for handwritten digit recognition. International conference on artificial neural networks; 1995; Perth, Australia. 1995. p. 53-60.

[17] Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn. 1999;36(1):105-39.

[18] Sood M, Kumar V, Bhooshan SV. Comparison of Machine Learning Methods for prediction of epilepsy by Neurophysiological EEG signals. Int J Pharm Bio Sci. 2014;5(2):6-15.

[19] Fernández-Delgado M, Cernadas E, Barro S, Amorim D. Do we need hundreds of classifiers to solve real world classification problems. J Mach Learn Res. 2014;15(1):3133-81.

[20] Omidiora EO, Adeyanju IA, Fenwa OD. Comparison of machine learning classifiers for recognition of online and offline handwritten digits. Comput Eng Intell Syst. 2013;4(13):39-47.

[21] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825-30.

[22] Chang C-C, Lin C-J. LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol TIST. 2011;2(3):1-27.

[23] Detrano R, Janosi A, Steinbrunn W, Pfisterer M, Schmid J-J, Sandhu S, et al. International application of a new probability algorithm for the diagnosis of coronary artery disease. Am J Cardiol. 1989;64(5):304-10.

[24] Frey PW, Slate DJ. Letter recognition using Holland-style adaptive classifiers. Mach Learn. 1991;6(2):161-82.

[25] Bache K, Lichman M. UCI Machine Learning Repository [Internet]. USA: University of California; 2013 [cited 2016 Mar 13]. Available from: http://archive.ics.uci.edu/ml .

[26] Siebert JP. Vehicle recognition using rule based methods. Project report. Glasgow, Scotland: Turing Institute; 1987.