# An Evaluation of Supervised Classification Techniques for Thoracic Surgery Risk Prediction

Connor Goddard

Department of Computer Science, Aberystwyth University
Aberystwyth, Ceredigion, SY23 3DB
Email: clg11@aber.ac.uk

## I. Introduction

With medical diagnosis techniques now more comprehensive and accurate than ever before, health professionals are often faced with analysing an overwhelming quantity of data in support of patient diagnoses. Whilst it may be assumed that greater quantities of data equates to better knowledge, it is often the contrary that is found to represent reality. For data to become knowledge, it is not enough to simply be able to consume and observe the data; instead one must look to *understand* the patterns represented within the raw values if any meaningful findings are to be found. As data complexity begins to increase, identifying these patterns can quickly become a significant challenge for humans to overcome.

Owing to their innate capacity for mathematical calculation and analytical processing, computers are naturally well equipped toward sifting through large volumes of high-dimensional data in pursuit of discovering useful patterns. This process is known formally as *data mining*, and represents a specialist application of machine learning.

This paper presents an investigation examining the performance of three supervised classification algorithms as they seek to identify patterns in pre-operative clinical data for patients awaiting thoracic lung cancer surgery, before using this information to make future predictions as to the risk of patient death within a year of completing surgery.

All experimentation work conducted for this investigation was performed using the *Weka* open-source machine learning workbench (**v3.7.13**) published by the University of Waikato, New Zealand [1].

## II. Analysis of Thoracic Surgery Dataset

Inspection of the dataset revealed a total of 300 patient records, with each comprising of 16 attributes representing a quality associated with the health condition or lifestyle characteristics of a given patient [2]. A final attribute (bringing the total number to 17) represented the *class* that would become the predictive target for the resulting supervised classifier.

Whilst most attributes within the dataset were of a nominal type (i.e. represented by a finite range of values), a small proportion, namely *AGE*, *PRE4* and *PRE5*, adopted a numeric distribution. Figure 1 presents the scale distributions for each of the three numeric attributes, where the differences in range between each attribute becomes apparent.

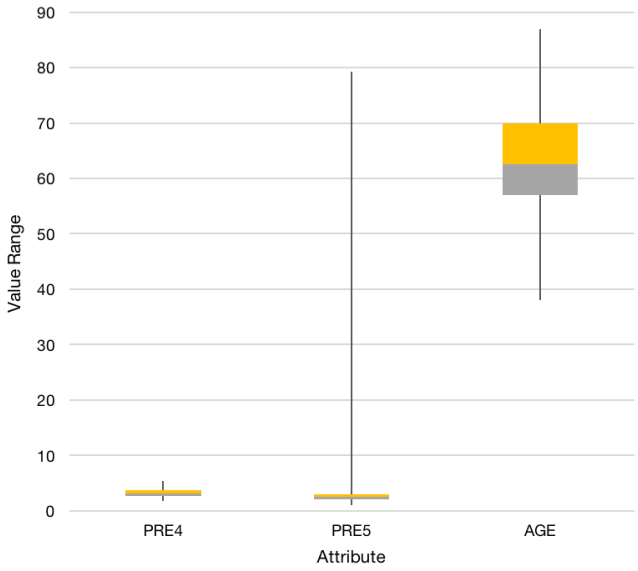



Fig. 1: Box plot showing scale distributions for numeric attributes contained within supplied Thoracic Surgery dataset.

In the case of *PRE5*, a clear irregularity originating in the upper quartile ($Q3$) indicates the presence of values that deviate away from the majority distribution within the attribute population. Figure 2 highlights the abnormal distribution of these values in relation to the upper ($UQ$) and lower ($LQ$) outlier limits denoted as [3]:

$$UQ = Q_3 + (1.5 \times IQR) \tag{1}$$

$$LQ = Q_1 - (1.5 \times IQR) \tag{2}$$

where the interquartile range ($IQR$) is represented by:

$$IQR = Q_3 - Q_1 \tag{3}$$

Upon closer inspection of the frequency distribution for *Risk1Yr* class attribute, a noticeable imbalance was found to exist between the two class values. Figure 3 highlights this imbalance between the number of negative cases (blue) and the number of positive cases (red) of patient death occurring within a year of undergoing thoracic surgery.

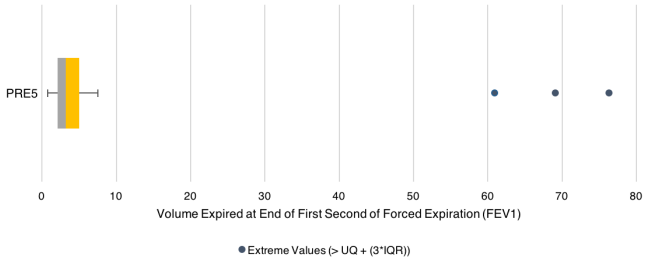



Fig. 2: Box plot showing anomalous extreme values for the *PRE5* attribute.

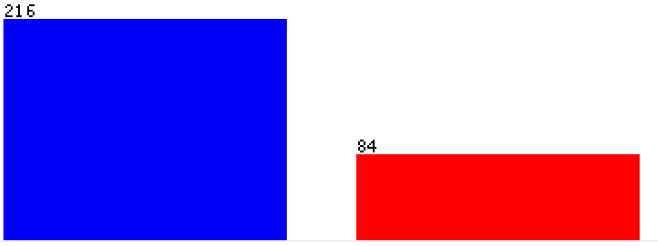



Fig. 3: Histogram distribution for the two class values representing the *Risk1Yr* attribute.

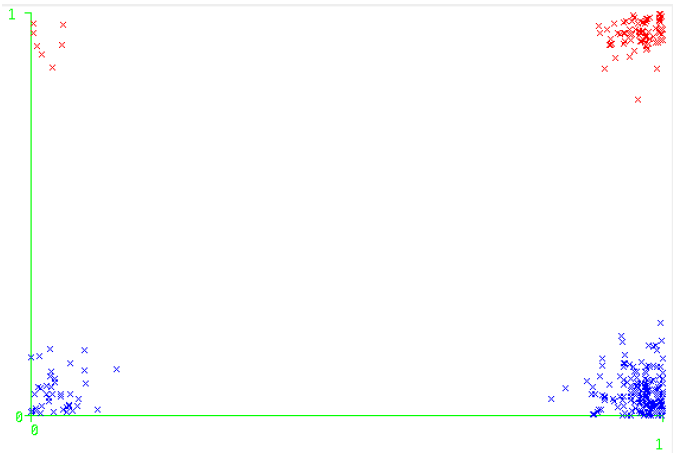

Fig. 4: Histogram distribution for the redundant *PRE32* attribute in which all 300 training instances were set to the same nominal value.

Figure 5 presents a selection of scatter graphs illustrating a selection of preeminent correlations identified between a different dataset attributes, and were generated using the visualisation tools provided in Weka.

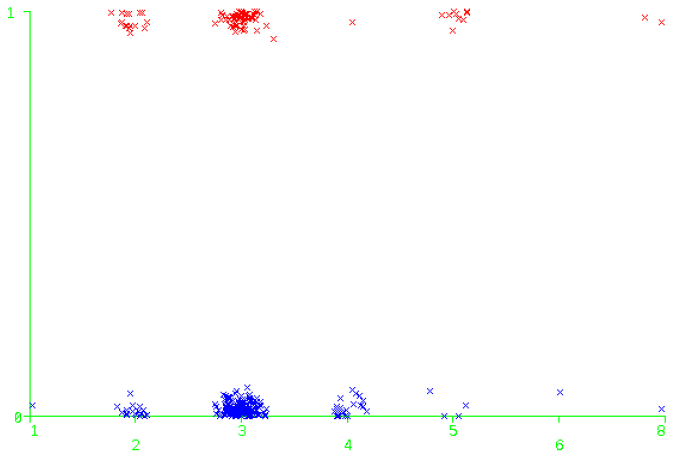
For the first plot (Figure 5a) a marked increase in the number of patient deaths is noted in the case of patients known to smoke prior to undergoing surgery. It is also found that a greater proportion of patients overall tend to be active smokers, which may shed light onto the reasons for the increase in death rate in this case.

In Figure 5b it is noted that the majority of patient diagnoses tend to amass toward the types represented by the 2, 3 and 4 ICD-10 codes, although however, patients that fall into the higher type categories (notably 5 and 8) tend to show the lowest overall survivability rates, with over 50% of the patients within these categories failing to survive longer than a year after surgery.

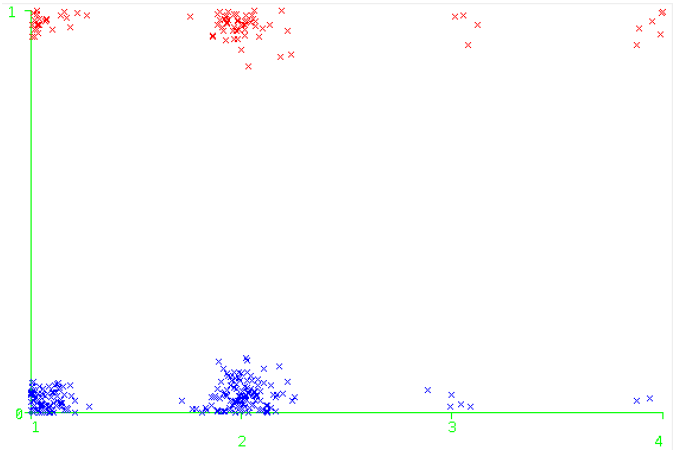
A similar picture emerges in the case of the third scatter plot




(a) Distribution correlation between smoking behaviour (*PRE30*) (X) and patient death within a year of thoracic surgery (*RiskYr1*) (Y).




(b) Distribution correlation between condition diagnosis type (*DGN*) (X) and patient death within a year of thoracic surgery (*RiskYr1*) (Y).




(c) Distribution correlation between original tumour size (*PRE14*) (X) and patient death within a year of thoracic surgery (*RiskYr1*) (Y).

Fig. 5: Selection of pairwise scatter plots generated in Weka, showing significant correlations between attribute pairs and their mapping to class values.

(Fig 5c), which sees a strong negative correlation between the original tumour size and the subsequent patient survival rate.

## III. Selection of Supervised Classifiers

Having considered the findings arising from the dataset analysis, three supervised classification algorithms were selected to undergo further evaluation of their predictive performance when trained and validated against the supplied dataset.

### A. Naive Bayes

The Naive Bayes algorithm is one of a family of probabilistic classifiers that employ Bayesian theorem to predict for new cases of a classification problem, the probability that a given instance belongs to a particular class based upon the conditional probabilities of each attribute belonging to that given class [4]. Expressed mathematically, Bayes' theorem is denoted as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{4}$$

where $A$ and $B$ are probability events; $P(A)$ and $P(B)$ are the probabilities of $A$ and $B$ in isolation; and $P(A|B)$ and $P(B|A)$ are the conditional probabilities of $A$ occurring given $B$ and $B$ occurring given $A$ respectively [4].

The algorithm is known as "naive", owing to the often invalid assumption that it holds toward conditional independence between each set of attributes used in predicting new cases. Whilst this assumption rarely holds true in real-life scenarios, the algorithm is often found to perform well despite the simplified prediction model, with fast execution and transparent modelling arising as key benefits [5]. For the particular circumstances surrounding this investigation - where highly critical healthcare decisions are being taken upon sensitive medical information - the latter characteristic becomes a particularly important aspect under the consideration of classifier suitability.

### B. Decision Tree

A decision tree is a constructive classification model in which the entire hypothesis space is represented using a structure of nodes and branches to map tests conducted on the characteristics of attributes to outcomes denoting the assignment of a particular class label [6]. As the fields of machine learning and data mining have grown, a number of algorithms have emerged for generating decision tree structures from categorical and continuous datasets. For this investigation, the popular C4.5 algorithm [7] would become the focus of examination, although no particular preference was given toward selecting this specific algorithm outside of the fact that it represented an implementation readily provided as part of Weka (specifically, the open-source '*J48*' Java implementation [8]).

In constructing a decision tree, individual attributes are assessed in order to determine the best 'splitting point' at which to create a new subset of decision branches corresponding to a particular aspect of the data under evaluation. At the core of this assessment (in the case of C4.5) lies the concept of *information entropy*, which provides a measure of data homogeneity (i.e. how 'similar' data values are from one another) set between the the value '0' in the case where data is 100% homogenous, and '1.0' in the case where data is equally divided [9]. In the case of the C4.5 algorithm, attributes are ranked based upon whichever returns the highest decrease in entropy after splitting the dataset at the current point in the tree structure [9]. This measure is known as the *Information Gain*, which is denoted mathematically as:

$$IG(T, X) = E(T) - E(T, X) \tag{5}$$

where $E(T)$ corresponds to level of entropy before splitting on the current attribute at the given location in the tree, and $E(T, X)$ corresponds to the level of entropy after the split has taken place [9]. Within any given decision tree structure generated by the C4.5 algorithm, attributes are selected in top-down fashion with the attribute exhibiting the greatest Information Gain designated as the root node.

Taking this all into account, a major advantage arising from the use of decision trees comes from the flexibility and robustness that they can offer with respect to the choice of structure, representation and transformation of attributes within training datasets. As a consequence, decision trees often require little in the way of dataset pre-processing in order to achieve satisfactory (and often greater) predictive performance [10].

Similarly to Naive Bayes, decision trees adopt a 'white-box' approach to derive their reasoning for arriving at a given classification outcome. The structures generated by algorithms like that of C4.5 and ID3 [7] are - in the majority of cases - easy for humans to interpret and explain, making them particularly well-suited to applications where human-led verification is likely to be an important factor in invoking confidence in the final outcomes.

One significant disadvantage often associated with decision trees is the tendency for models to overfit the training data [9]. Overfitting can occur when the structure representing the decision space becomes too closely-tied to the particulars of the training data, causing the model to no longer generalise effectively to future cases that may share the same underlying trends, but not the same exact values. A common approach to reduce the risk of overfitting is to undertake pruning of the decision tree structure which can reduce its overall complexity [9].

### C. Random Forest

Random Forests are an extension of traditional decision tree algorithms, which aim to reduce the level of variance caused by overfitting using a specialist type of ensemble process known as *Bagging* (Bootstrap aggregation).

The approach sees multiple decision trees trained on different samples of the original training set selected using random sampling with replacement [11]. Once a multitude of decision tree structures have been generated (each modelling a different

subset of the original training data), their predictions are collated and averaged to arrive at a final outcome.

This notion of averaging across the various predictions is what enables the random forest technique to exhibit lower variance in its prediction model than what could be achieved using a single tree alone. As the model for a random forest is essentially a combination of multiple decision trees, its bias remains the same as that of a single decision tree (which itself has generally low bias) [11]. The overall result of using a random forest therefore, is often a reduction in variance *without* significant change to the bias. This can often lead to random forests exhibiting markedly greater accuracy performance upon unlabelled test sets than that of C4.5 or ID3 [7].

## IV. Pre-processing of Training Data

This section outlines the work undertaken to pre-process the training dataset following the observations discussed in the Section 1.

In the interests of conducting a comprehensive evaluation of classifier performance, a separate dataset was created using the results obtained from performing *Principle Component Analysis* [12] to extract structures from the data that emphasise the strongest patterns and variation across the collection of attributes.

### A. Outlier Removal

Prior to performing any scaling or aggregation of attribute values, it was first important to remove any anomalous values that may lead to transformation results becoming skewed or misrepresented. Removing outliers and/or extreme values within the Thoracic Surgery dataset required the combined efforts of two unsupervised filters provided in Weka.

The first filter, `InterquartileRange`, sequentially assessed the value distribution for each attribute to detect any outliers or extreme values that fell outside of the interquartile range calculated in each case. In the event that an anomalous values was detected, this was subsequently added to one of two new 'meta-attributes' denoting the presence of the outliers and/or extreme values.

Having detected all of the outliers and extreme values across the dataset, these values were subsequently removed using the unsupervised `RemoveWithValues` filter to select attribute values based upon their inclusion within either the meta-attributes produced as a result of the IQR analysis.

Figure 6 shows the detection of the extreme values (denoted in red) within the distribution of values for *PRE5* attribute.

### B. Attribute Cleaning

A simple and effective method of immediately reducing the dimensionality of a dataset is to remove any attributes that fail to exhibit any variation in their range of values. In the case of the Thoracic Surgery dataset, the *PRE32* attribute was found to be constant as shown in Figure 4, and therefore represented a prime candidate for removal.

To assist in this task, Weka provides the unsupervised `RemoveUseless` filter, which synopsis reads "*A filter for removing attributes that do not vary at all or that vary too much.*" [13]. After applying this filter, the *PRE32* attribute was automatically removed from the training set, bringing the total number of attributes to 16.

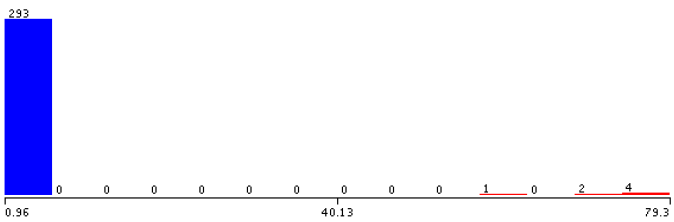



Fig. 6: Histogram indicating extreme values present in the value distribution for the *PRE5* attribute (EV are denoted in red).

### C. Normalisation

For many classification algorithms, working with datasets that use different types of scale across their numeric attributes can often lead to a reduction in classification performance, arising out of a mismatch of weights between the representations for each range of value distributions [14]. In looking to overcome this issue, *normalising* the values for each attribute - such that all distributions use the *same scale* - can often elicit immediate improvements to a classifier's performance .

Using the `Normalize` filter available in Weka, the distributions for three numeric attributes contained within the Thoracic Surgery dataset (*AGE*, *PRE4* and *PRE5*) were each transformed to a range between the values of 0 and 1.

### D. Discretisation

Discretisation is a mathematical process concerned with the transformation of values from using a continuous feature space, to one that is discrete [15]. This technique represents a common task undertaken within the pre-processing stage for many machine learning applications, owing to the enhanced support that many classification algorithms apply to the use of nominal over numeric values in training against a given dataset [14].

In Weka, one can choose to perform either supervised or unsupervised discretisation. In the supervised approach, the correlation between each attribute and the class attribute is used to advise the transformation process. In the unsupervised approach, simple value binning is performed, with no consideration given to the class attribute [16]. To investigate the differences between both approaches, separate datasets were created for each.

### E. Class Balancing

When faced with addressing an imbalanced training set, one is presented with two main choices: (a) gather additional results that can increase the overall number associated with the minority class; or (b) artificially adjust the proportions of each class within the existing collection of data.

In undertaking the latter approach, one can elect to apply under-sampling of the majority class; oversampling of the minority class; or a combination of both. Often this choice will depend on the number of instances that exist prior to balancing. Given that the original Thoracic Surgery dataset did not hold a particularly large collection of data, it was decided that oversampling of the minority class would represent the most appropriate course of action in this case.

Rather than simply replicating instances from the minority class - which can often increase the risk of overfitting [17], this investigation generated brand-new synthetic instances that shared some - but not all - of the traits from the original minority set. Creating these synthetic instances was performed using the Weka implementation of *Synthetic Minority Over-sampling Technique* (SMOTE) algorithm [18], resulting in a doubling of the original number of minority class instances (equivalent to a sample frequency of 200%).

A final, yet critical consideration related to addressing class imbalance, was to ensure that balancing took place *during* cross-validation rather than before it. This is because performing cross-validation upon pre-balanced data invokes an increased bias upon the minority class that is likely to cause symptoms of overfitting when later applied to the test set [19]. In Weka, this was achieved through the use of a `FilteredClassifier` [20].

## V. Optimisation of Classifier Hyper-Parameters

When applying machine learning algorithms to a specific dataset, it is often the case that the parameters used to configure those algorithms will require some form of additional tuning in order to elicit the best possible performance within the context of the defined problem space [21].

For the three classifiers examined in this investigation, each would be subject to a range of tests looking at the effect on their relative performance following changes made to their configuration.

### A. Naive Bayes

In Weka, the Naive Bayes implementation offers a comparatively limited selection of tuning options: `useSupervisedDiscretization` - which specifies whether numeric attributes should be converted to nominal attributes prior to building the learning model; and `useKernelEstimator` - which specifies whether a numeric attribute should be assessed using *kernel density estimation* rather than assuming a normal distribution [22]. This parameter is applicable in cases where the distribution for a numeric attribute may include multiple peaks or exhibits symptoms of skewing [23].

Whilst no numeric attributes were found to adopt a continuous distribution within this investigation, for the sake of completeness both parameters were subject to testing as part of the classifier evaluation.

### B. Decision Tree (J48)

In the case of the J48 Decision Tree implementation, particular investigative focus was given toward the parameter settings relating to the pruning method used to adjust the size and complexity of the generated tree structure.

These tests would focus on the more subtle aspects of pruning behaviour that can influence the size and complexity of the final model. One parameter associated with the *post-pruning* of the tree structure is the *confidence factor* - which represents a threshold for controlling the likelihood that a given node is pruned based on the estimation of classification error that is compared between the cases where: (a) the node and any children are kept; and (b) the children are removed and replaced with a class label assigned to the majority class [24].

### C. Random Forest

Similarly to that of the Decision Tree algorithm, investigation of the hyper-parameters linked to the Random Forest classifier would primarily focus on studying the effects caused to the overall accuracy following adjustments made to the size and complexity of the collaborative tree model.

In undertaking this study, two parameters in particular would be subject to investigation using the configuration tools provided through Weka. The first parameter: `numFeatures` - allows for modification of the number of potential attributes that the Random Forest algorithm will consider whilst randomly selecting the attribute upon which to perform the next split during the incremental generation of the current sub-tree structure [25].

The second parameter: `numTrees` - controls the number of trees that are produced in support of building the collaborative classification model. Whilst increasing this parameter can lead to a notable improvement in classification performance, it often comes with a rise in computational cost; proportional to the increase in model complexity [26].

To address issues surrounding lengthy training times, one final parameter considered as part of the investigation was the number of parallel execution slots (i.e. threads) used in order to generate the final Random Forest model. Parallelisation of Random Forests is made possible; given that each tree structure is generated independently of one another [27]

## VI. Experiments & Results

Under this investigation, a total of 10 experiments were conducted into one of two assessment areas relating to the evaluation of performance between the Naive Bayes, Decision Tree and Random Forest classifiers, when operated under a range of different assessment conditions.

The first set of experiments would focus on evaluating the effect that alternative forms of pre-processing and dimensionality reduction applied to the training set can have upon the accuracy for each of the three classification algorithms.

In the second set of experiments, a systematic assessment into the impacts caused by tuning of classifier hyper-

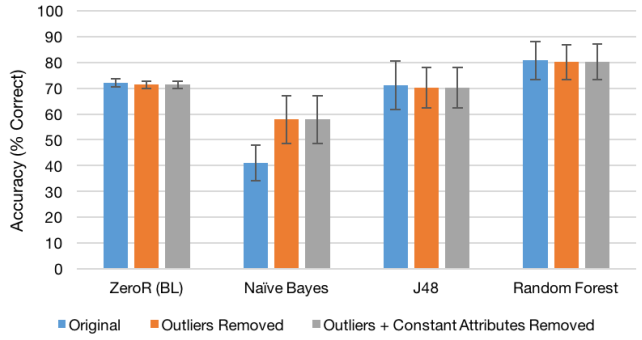
parameters was conducted. These adjustments would be undertaken in line with the plan discussed in Section 5.

Across all experiments involving comparison between the three subject classifiers, the *ZeroR* classifier was employed to act as the performance benchmark. The classifier is often adopted for this role, given that it exhibits no predictive capabilities and instead invariably selects the majority class for all its classification outcomes [9].

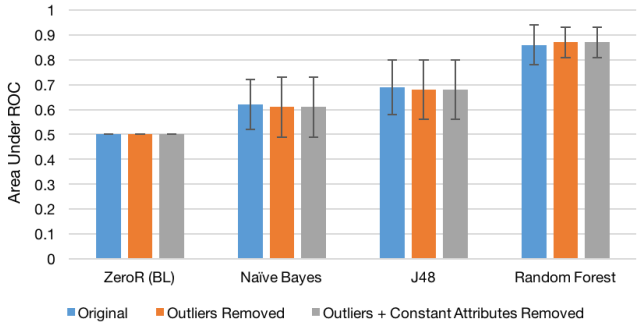
All tests were conducted using the *Experimenter* tool provided as part of the Weka platform using 10-fold cross validation. To maximise fairness, each test run was repeated over 10 iterations (equivalent to 100 runs per test case) and their results averaged.

### *A. Experiment Set 1: Pre-Processing & Dimensionality Reduction*

The first pre-processing step to be assessed involved the removal of any outliers and extreme values alongside any 'useless' attributes that provided no variation across their value distributions. Figure 7 shows the performance recorded before and after the dataset was cleaned.




(a) Classifier accuracy (measured as the percentage of correct classifications)

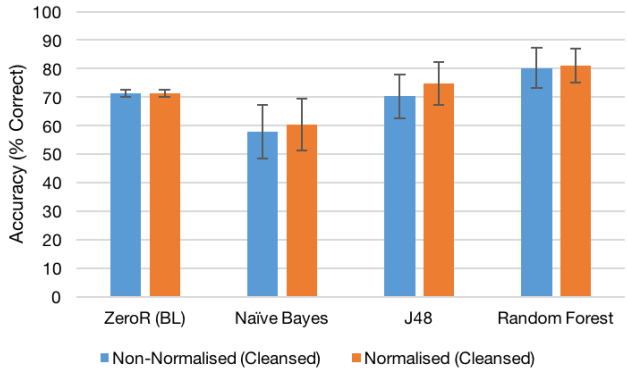



(b) ROC analysis of classifier performance

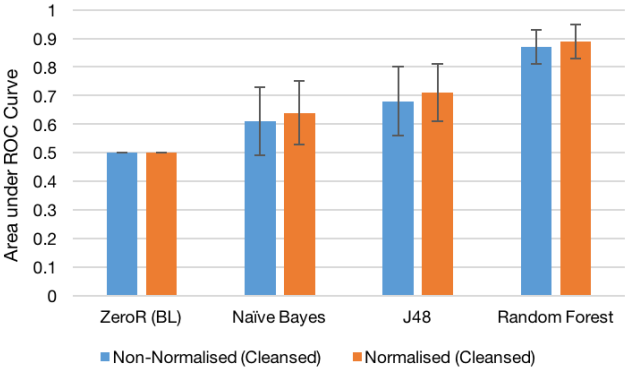
Fig. 7: Results of classification performance between datasets excluded-from or subjected-to outlier removal.

Most notable is the significant rise in accuracy performance for the Naive Bayes classifier after the removal of outliers. This result is not reciprocated in the case of the other two classifiers, owing to the fact that outliers exhibit little influence on the outcome of node splitting, in the same way that the median exhibits robustness to outliers as a measure of central tendency within probability distributions [28]. The results also confirm predictions that removing constant attributes warrants no effect on classifier performance, thereby advocating this simple form of dimensionality reduction.

The next test investigated the impact of normalising each of the numeric attributes to share a common distribution scale. Figure 8 shows evidence of a very slight improvement in the performance of all three classifiers, but none were found to be statistically significant in this case.



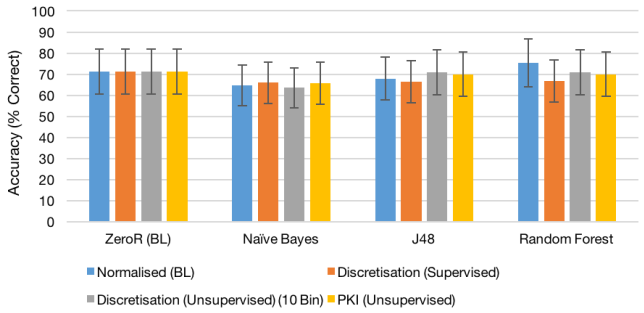

(a) Classifier accuracy (measured as the percentage of correct classifications)



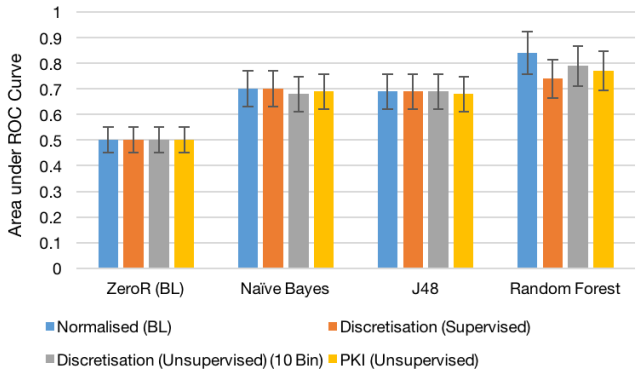

(b) ROC analysis of classifier performance

Fig. 8: Results of classification performance between datasets excluded-from or subjected-to normalisation.

Next, focus was directed toward the sampling strategy used to balance the distribution of class labels across the dataset. This was assessed in conjunction with alternative methods for the discretisation of numeric attribute values to provide greater scope of their combined contributions toward affecting classifier performance. Figures 9 and 10 present the accuracy and AUC results for the under-sampling and over-sampling strategies respectively.

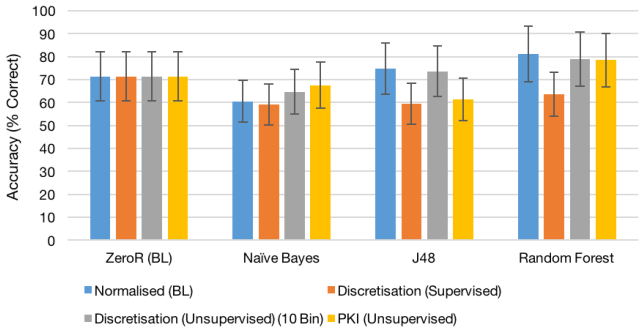



(a) Classifier accuracy (measured as the percentage of correct classifications)




(b) ROC analysis of classifier performance

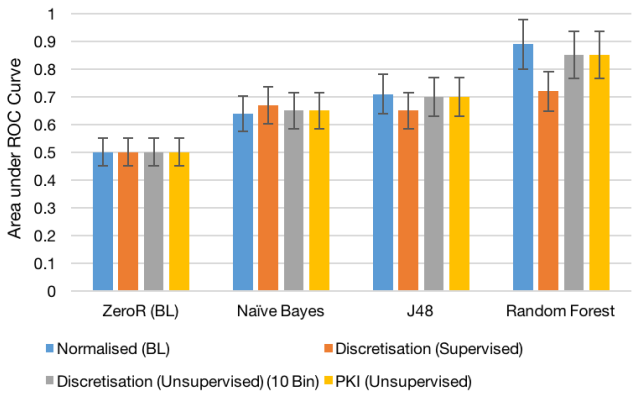
Fig. 9: Results of classification performance for under-sampled datasets employing alternative forms of discretisation.




(a) Classifier accuracy (measured as the percentage of correct classifications)

Here we find discretisation performs worse in the case of both tree-based classifiers, and provides only marginal benefit for Naive Bayes. Over-sampling elicits a minor boost in performance for cases involving normalised datasets, however is often found to reduce performance across discretised sets, particularly when this has been conducted using a supervised approach.

The final set of tests were concerned with investigating the impact of conducting Principle Component Analysis to extrapolate the most important patterns from the range of existing dataset attributes. Figure 11 shows the results of PCA attribute




(b) ROC analysis of classifier performance

Fig. 10: Results of classification performance for over-sampled datasets employing alternative forms of discretisation.

ranking conducted using the `PrincipleComponents` attribute selection filter, where we see that the top-three ranked attributes all preserve over 70% of the total variance found across the dataset (measured by their corresponding *eigenvalues*).

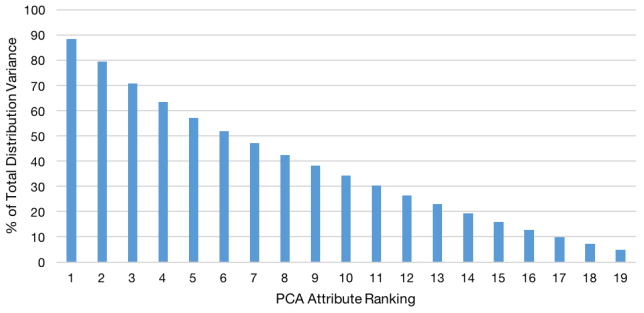
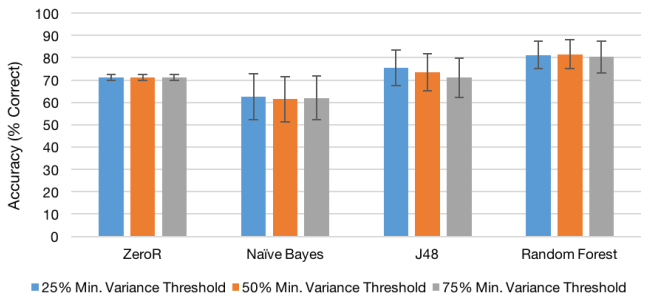


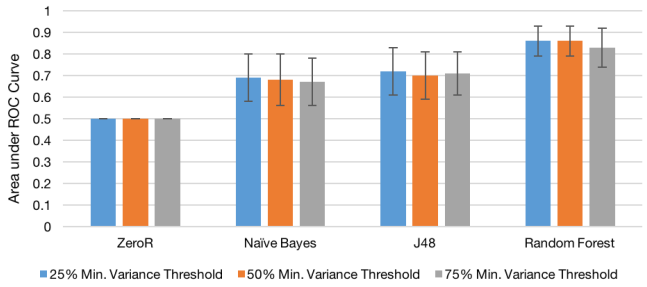

Fig. 11: PCA attribute ranking conducted against the total distribution variance captured for each across the entire dataset.

Next, an evaluation of classifier performance was conducted after applying a range of variance thresholds to limit the number of PCA attributes available within the training dataset. The results presented in Figure 12 indicate a near-equal level of performance achieved between the group of top-twelve attributes (each preserving at least 25% of the total variance), and the group of top-three attributes (each preserving at least 75% of the total variance). In Figure 13 this comparison is further extended to other existing types of pre-processed datasets, where for all three classifiers PCA reduction performs commendably against the current top-performer.

Both sets of results make for a compelling case toward the application of PCA with the Thoracic Surgery dataset, showing that comparative classifier performance can still be maintained even after reducing the total number of attributes by over 80%. When considering this same idea from the other direction, the additional overheads associated with applying PCA reduction (predominately time) failed to provide any




(a) Classifier accuracy (measured as the percentage of correct classifications)




(b) ROC analysis of classifier performance

Fig. 12: Results of classification performance for datasets containing PCA attributes exhibiting a range of minimum variance thresholds.

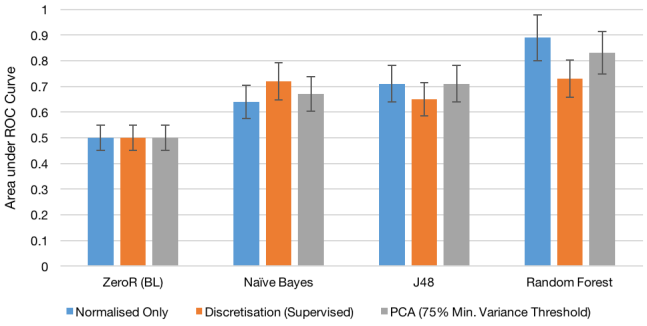


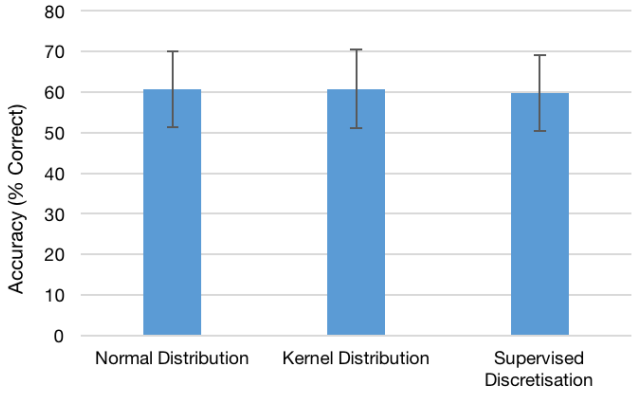

Fig. 13: Results of classification performance achieved between normalised, discretised and PCA datasets.

significant advantage over continuing to utilise the original collection of attributes found within the dataset.

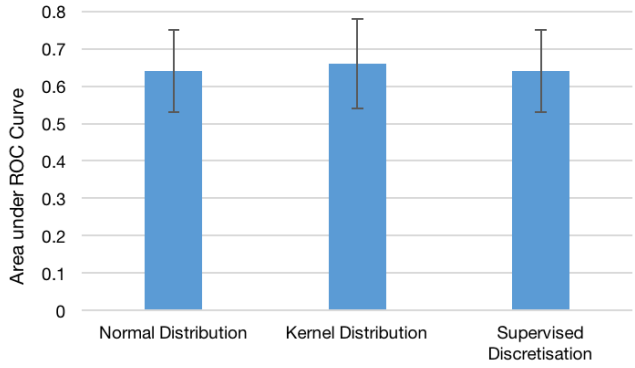
### B. Experiment Set 2: Hyper-parameter Optimisation

In the second set of experiments, tests were conducted to evaluate the performance arising through adjustment of the classifier hyper-parameters outlined in Section 5. To allow for a clear and fair investigation, each classifier was assessed independently using a common training dataset (pre-processed to remove outliers/useless attributes and normalise numeric attributes).

*1) Naive Bayes:* Figure 14 presents the performance results for the Naive Bayes classifier after switching between the three configuration options associated with the handling of numeric attributes. This shows very little in the way of performance changes, although adopting kernel density estimation does appear to provide a marginal boost to the AUROC, however this increase is likely to be counteracted by the additional time and memory overheads that are often beared as a consequence [23].




(a) Classifier accuracy (measured as the percentage of correct classifications)



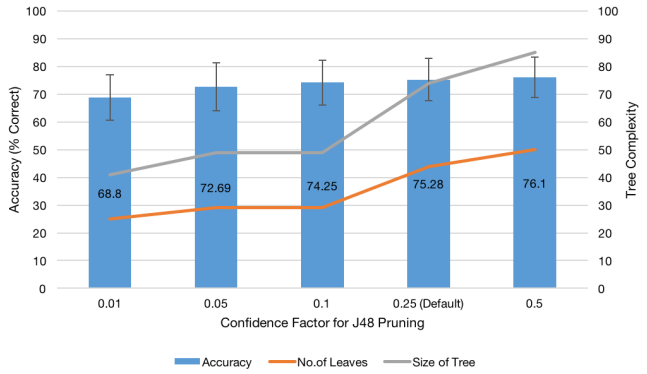

(b) ROC analysis of classifier performance

Fig. 14: Performance results for Naive Bayes classifier following tuning of selected hyper-parameters relating to the processing of numeric value distributions.

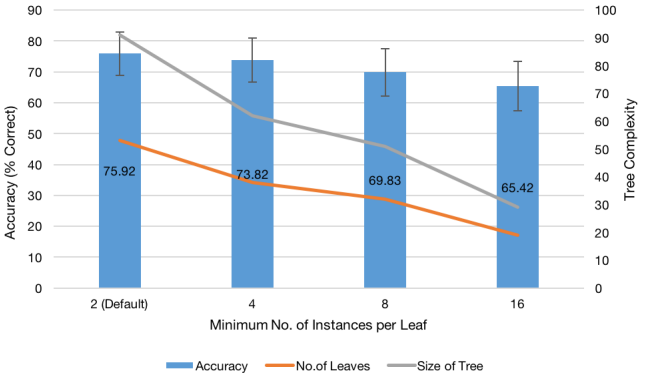
*2) Decision Tree (J48):* In Figure 15, we see the relationship between adjustments made to parameters controlling the size and complexity of the tree model, and the overall change in classification performance exhibited as a result. For the confidence factor (Figure 15a), we see the values '0.05' and '0.1' provide the optimum balance between increased accuracy performance and overall tree size, after which no tangible benefits are found to arise from what-becomes a significant increase in tree complexity.

In Figure 15b, we see that increasing the threshold for creating new decision nodes causes a dramatic drop in overall tree complexity, but at the sacrifice of classification accuracy. In this case, a minimum threshold of '4' instances per leaf

gives the best overall result, causing the tree complexity to drop by over 20%, with only a marginal loss of accuracy.




(a) Results following adjustments to the confidence factor for J48 pruning.




(b) Results following adjustments to the confidence factor for J48 pruning.

Fig. 15: Performance results for J48 Decision Tree classifier following tuning of hyper-parameters relating to pruning behaviour.

*3) Random Forest:* The final set of hyper-parameter experiments conducted for the Random Forest classifier focussed on the relationship between model complexity, algorithm training times and resultant accuracy performance, along with an assessment into the use of parallelisation as a means of accelerating training and testing times.

Figure 16 presents the results following adjustments to the number of trees generated within the ensemble representing the final model. Here we find reducing the number of trees to be of a clear advantage, with half of the default number of trees demonstrating equivalent performance to that of the larger structures whilst also enabling faster training performance (approx. 20ms increase from the default setting).

A similar set of results are found in the case of Figure 17, where on this occasion the default number of considered attributes appears to provide the best compromise between classification accuracy and training time complexity.

In the final set of tests, attention turned to the utilisation of multiple CPU cores to allow for the training and execution of the Random Forest classifier to be conducted in parallel.

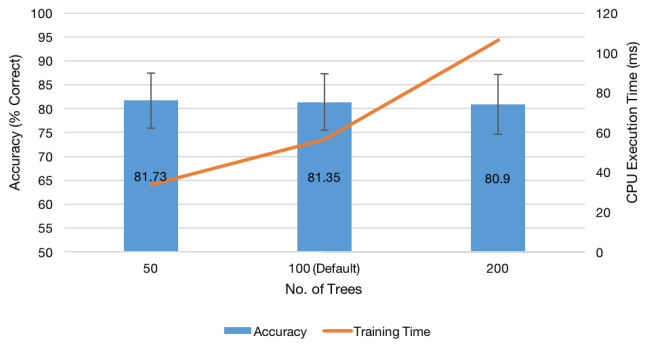



Fig. 16: Performance results for Random Forest classifier following tuning adjustments made to the number of trees generated for the final classification model.

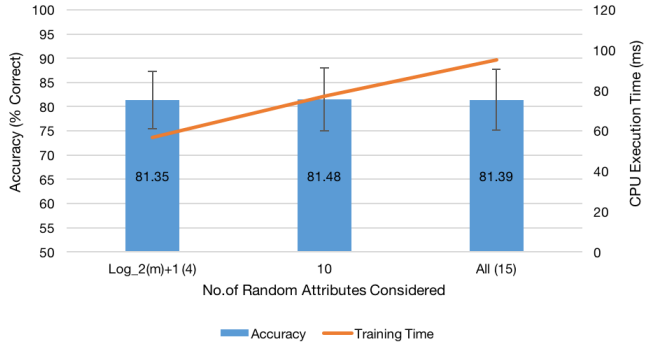



Fig. 17: Performance results for Random Forest classifier following tuning adjustments made to the number of random attributes considered when conducting a new split on an existing leaf node.

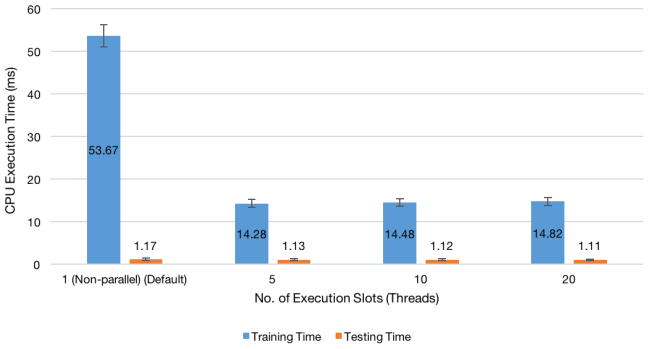



Fig. 18: Performance results for Random Forest classifier following tuning adjustments made to the number of simultaneous execution threads used to generate the final classification model.

Figure 18 shows the results of these tests assessing performance over a range of thread counts, including the default value that limited execution to only a single thread (i.e. non-parallel execution). Whilst the benefits of multi-threading are apparent, we see this performance increase rapidly level-off

as a result of reaching the maximum number of CPU cores available within the testing platform.

## VII. Conclusions & Future Work

This paper has provided an investigation into the factors that can affect supervised classification methods when used to predict the risk of death in patient candidates selected to undergo surgical treatment for lung cancer.

From the three supervised classifiers selected for evaluation, this investigation finds the ensemble-based Random Forest classifier to demonstrate the most effective tradeoff between predictive accuracy and computational cost when trained in parallel using a dataset pre-processed to remove outliers and normalise attributes expressing numerical distributions. Whilst Principle Component Analysis proved to be an effective approach for reducing the dimensionality of the training set, no compelling empirical evidence was found to warrant employment of this technique over the continued use of existing dataset attributes (excluding any removed as a result of pre-processing).

Proposals for future work include the enhancement of the Naive Bayes and J48 classifiers via the boosting and bagging ensemble methods, and a widening of the investigation scope to include other types classifier including instance-based methods (e.g. K-Nearest Neighbour [29] & KStar [30]) and support vector machines [31].

## References


[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[2] M. Zikeba, J. M. Tomczak, M. Lubicz, and J. 'Swikatek, "Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients," *Applied Soft Computing*, 2013.

[3] J. W. Tukey, "Exploratory data analysis," 1977.

[4] C. Lu, "Bayesian Decision Theory and Bayesian Networks - CSM6420," Mar. 2016.

[5] T. Mitchell, *Machine learning*. McGraw-Hill Boston, 1997.

[6] C. Lu, "Decision Tree Learning - CSM6420," Mar. 2016.

[7] J. R. Quinlan, "C4.5: Program for Machine Learning," *San Mateo, CA, USA*, 1993.

[8] E. Frank, "J48 - Weka," http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html, 2016.

[9] S. Sayad, *An introduction to Data Mining*. University of Toronto, 2011.

[10] M. Nashvili, "Data Mining With Decision Trees," http://decisiontrees.net, May 2016.

[11] S. Fortmann-Roe, "Understanding the Bias-Variance Tradeoff," Jun. 2012.

[12] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[13] R. Kirkby, *Weka - RemoveUseless*, http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/RemoveUseless.html, University of Waikato, 2016.

[14] J. Brownlee, "How to Prepare Data For Machine Learning," http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/, Dec. 2013.

[15] J. Dougherty, R. Kohavi, M. Sahami, and Others, "Supervised and unsupervised discretization of continuous features," in *Machine learning: proceedings of the twelfth international conference*, vol. 12, 1995, pp. 194–202.

[16] Fracpete, *Weka Wiki - Discretizing datasets*, https://weka.wikispaces.com/Discretizing+datasets, University of Waikato, Jan. 2013.

[17] Y. Mi, "Imbalanced classification based on active learning SMOTE," *Research Journal of Applied Science Engineering and Technology*, vol. 5, pp. 944–949, 2013.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.

[19] M. Altini, "Dealing with imbalanced data: undersampling, oversampling and proper cross-validation." http://bit.ly/1VO8HfG, Aug. 2015.

[20] L. Trigg, *Weka - FilteredClassifier*, http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/FilteredClassifier.html, University of Waikato, 2016.

[21] J. Brownlee, "How to Improve Machine Learning Results," http://machinelearningmastery.com/how-to-improve-machine-learning-results/, Dec. 2013.

[22] L. Trigg, *Weka - NaiveBayes*, http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/FilteredClassifier.html, University of Waikato, 2016.

[23] (2016, Apr.) Naive Bayes Classification. http://uk.mathworks.com/help/stats/naive-bayes-classification.html. MathWorks Inc.

[24] S. Drazin and M. Montag, "Decision tree analysis using WEKA," *Machine Learning-Project II, University of Miami*, pp. 1–3, 2012.

[25] J. Brownlee, "Tune Machine Learning Algorithms in R - Random Forest case study," http://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/, Feb. 2016.

[26] J. Ramon. (2013, Dec.) How to determine the number of trees to be generated in Random Forest algorithm? https://www.researchgate.net/post/How_to_determine_the_number_of_trees_to_be_generated_in_Random_Forest_algorithm.

[27] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, "Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?" in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. IEEE, 2012, pp. 232–239.

[28] A. M. Committee and Others, "Robust statistics: a method of coping with outliers," *Technical brief*, no. 6, 2001.

[29] N. S. Altman, "An introduction to kernel and nearest-neighbour nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[30] J. G. Cleary, L. E. Trigg, and Others, "K*: An instance-based learner using an entropic distance measure," in *Proceedings of the 12th International Conference on Machine learning*, vol. 5, 1995, pp. 108–114.

[31] K. Veropoulos, N. Cristianini, and C. Campbell, "The application of support vector machines to medical decision support: a case study," *Advanced Course in Artificial Intelligence*, pp. 1–6, 1999.