Predicting one-year postoperative death rates following thoracic surgery

Craig Heptinstall Crh13- 110005643
SEM6420
Institute of Computer Science
Aberystywth University

***Abstract*—Predicting medical data is becoming more and more vital towards helping reduce death rates, or reducing risks in medical procedures. With the use of tools such as WEKA [1], the ability to build machine learning classifiers has become easier, and has helped produced more reliable results. This paper looks at death rate prediction following thoracic surgery, and compares different means of prediction and configurations on a given test set of data.**

I. Introduction

Thoracic surgery is a means of lung resection, and is performed in order to remove part or all of a lung from a patient who is suffering or has suffered from lung cancer [2]. The data provided for training a machine learning approach to predicting the death rate one year after surgery contains 300 patients, each of which hold several attributes, including age and 'Risk1Yr'. The most latter of these states is the example patient survived (indicated by a 0) or did not (indicate by a 1). The aim of the classifier trained at the end of this report is to be able to predict the survival rate in conjunction with an identical test set of data.

For easier statistical reading, and the ability to easily try a range of classifiers on the training data, Weka was chosen to perform experiments and training. Weka allows pre-processing of data, classifiers and filters to be applied, and also provides good visualisation of training results, such as ROC curves. Throughout this paper, both graphs generated within Weka, and ones created from comparing different classifiers will help to reinforce result conclusions.

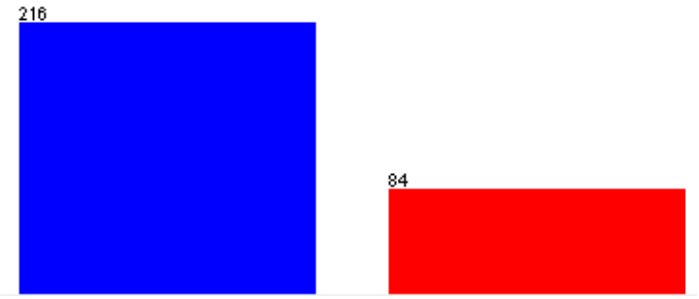
To visualise data quickly during the fine tuning and selection of classifiers during the start of this project, the Weka 3.7 Explorer application will be used, and then will be moved onto the experimenter during latter parts of the project in order to quickly run multiple experiments at once. Finally, the Weka Knowledge flow application will allow comparisons of the ROC curve of multiple solutions in order to choose the most suitable set up for predicting the test data results.

As stated in the assignment, more than one classifier will be selected and compared to see which one best suits predicting this type of data. Following that, more comparisons will be made looking at changing a range of hyper-parameters for chosen classifiers.

A. Data pre-processing

In Weka, and in any machine learning approach, some data pre-processing should be performed. Because the data set is real-life based, data is not always complete, or follow a linear pattern. Due to this, an issue known as class imbalance can hinder the performance of learning systems, and means that one instance of an attribute (in this example, surviving patients) is the majority sample causing a bias when training the machine learning algorithm. On first inspection of the data in the training set, it can be seen in Figure 1 that the 'Risk1Yr' attribute holds far more surviving patients that ones who did not.

Fig. 1. Visualisation of training set risk year 1 mortality rate. Left: Survivors, Right: Deaths



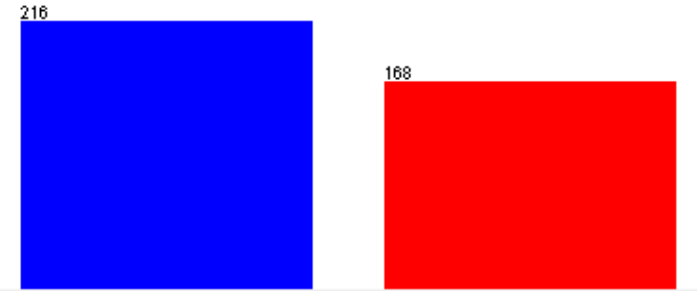
There are two means of settling imbalances in data however [3]:

1) Under-sampling where the size of the samples are reduced proportionally
2) Over-sampling where the size of samples are increased proportionally

In the case of Weka, there are filters available within the pre-processing stage of experiments allowing the addition of under or over-sampling. In this case, the SMOTE (Synthetic Minority Over-sampling Technique) [4] plugin for Weka offered an easy means of re-sampling data based on a given set of parameters. In an accompanying paper [5], SMOTE is described as creating 'synthetic examples' rather than replacements that are generated in feature space rather than data space. In the Weka library, synthetic samples are created by using a range of nearest neighbour minority samples, and placing new samples

between them. By default, the SMOTE filter uses five nearest neighbours to create new samples, and as this is recommended for this size dataset, this will remain the selected value.

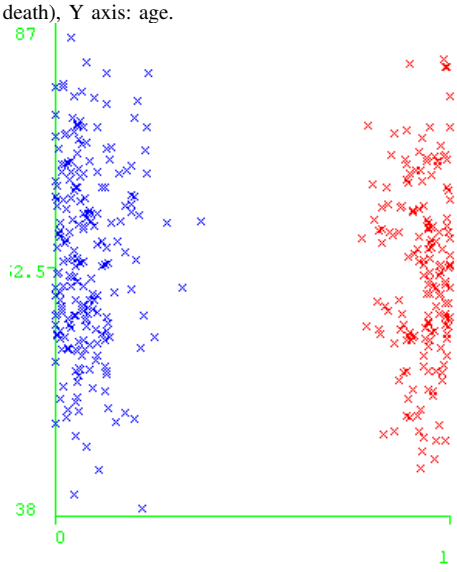
Fig. 2. Risk one year data after SMOTE filter applied.



Once the SMOTE filter was applied (to reproduce, this can be found in Appendix A), there was a clear view of change in the proportions of the survival and death rates. As Figure 2 shows, over-sampling has been applied to 200%, which has meant the minority of samples showing death rates only has 50 samples less than the majority as opposed to the 138 sample difference previously.

In addition to balancing data, viewing the data in Weka explorer on scatter gave good insight into any data trends, and also gave hints into which attributes carried more weighting towards the risk attribute than others. At first, both age and risk were plotted against each other to see if there were any trends towards being younger meant better survival rates (as shown in Figure 3). In order to view the scatter plot well enough, Jitter was added in Weka to plot points further away from each other.

Fig. 3. Scatter plot comparing age and survival rates. X axis: Death rate (0- survival, 1- death), Y axis: age.



Although the plot looks fairly evenly spread on age for both survival and mortality rates, there is a slight higher number of patients who survived aged around 50 than those who were older.

By looking at the scatter plots for various combinations of data, dimensionality reduction is something that can be used to both increase the accuracy and speed of the resulting algorithm. There are various means of dimensionality reduction, such as PCA (Principle Component Analysis) and SOM (Self Organising Maps) /citedimensionality, whilst in Weka, attributes can be ranked, in order to see which ones carry the most important in a data set.

For this experiment, the InfoGainAttributeEvaluator was used to rank attributes in Weka. This algorithm was selected because of the simple output of 'worth of an attribute' that allows the least useful attributes or features to be omitted from any classification later on. The attribute evaluator configuration run can be found in Appendix B From first run, and shown in Figure 3 is the results from the ranking. It appears that three attributes listed have no effect on the prediction of one year risk, so these will be omitted when testing classifiers in the following section.

Fig. 4. Ranked attributes using the Information gain attribute evaluator in Weka.

```
Ranked attributes:
 0.07451525   1 DGN
 0.03291264   3 PRE5
 0.02731049  10 PRE14
 0.01366042  14 PRE30
 0.01060421   4 PRE6
 0.00950188   8 PRE10
 0.00826647   7 PRE9
 0.00600333   9 PRE11
 0.00567997   5 PRE7
 0.00317208  12 PRE19
 0.00210082  13 PRE25
 0.00161183   6 PRE8
 0.00000876  11 PRE17
 0           15 PRE32
 0            2 PRE4
 0           16 AGE
```

Although only the three at the bottom were removed from this process, it is possible to remove more features. Three was chosen because it appeared to be a global minimum feature set before over-fitting occurred. This is when the learning curve appears to go back above its minimum due to not enough descriptive features.

II. Classifying Data

For the set of experiments following, multiple classifiers have been selected to find the best prediction possible. In a notable paper by D. Wolpert (The Lack of A Priori Distinctions Between Learning Algorithms) [7], this points out that there is not one classifier that will work for every problem in machine learning. Therefore, by comparing a range of classifiers, the

best traits will be found, and will contribute towards the final predictor provided in the appendices. This section takes a look at what classifiers have been selected, how each should work, and what fine tuning was done to each.

A. *Classifier selection*

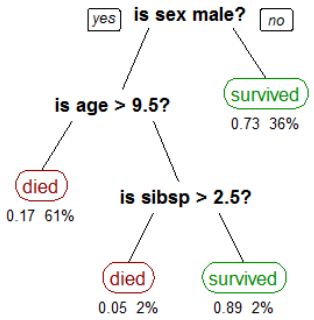
Three classifiers have been chosen for this part of the paper, and will be compared directly and indirectly in the results section of this report. The three classifiers selected are as follows:

- Decision tree - J48 for generating C4 trees
- Naive Bayes - A Naive Bayes classifier
- Neural network - Multilayer perception classifier using back-propagation

The first of which was selected because of both its practicality and popularity, which has been proved useful for data mining. Decision trees are used in a range of medical diagnoses, and because decision trees allow human readable rules of classification and are easy to interpret, they are very useful in this field of research.

With J48, similarly to most decision trees, the structure created appears like a flow chart, with each node denoting a test, each branch denoting a an outcome of a test, and each leaf node holding a class label. This classifier has two parts [8] to its use: the growth phase and the pruning phase. Trees split training sets based on the criteria (in this case mortality rate), and perform this until all records belonging to each split hold the same class label. Over-fitting is the outcome of creating trees because of this, therefore pruning is added to the scenario. Outliers and fuzzy data from the tree to ensure it holds only useful information. Construction of flow chart is quick, therefore the overall prediction rate in this paper should be quicker than other algorithms.

Fig. 5. Example of a decision tree, showing the attempt of using different ages to seperate cases.



The second classifier listed that will be compared in order to find out which algorithm will be most suited to this problem is a using a Naive Bayes approach. The simplest of the three, Weka offers an implementation of this which uses Bayes rule as shown in Figure 6. From first observation, it does appear that Bayes may work better with more evidence, though this may not be the case, as over-fitting may occur.

Fig. 6. Naive Bayes rule, where more evidence (attributes can be added after the first and multiplied).

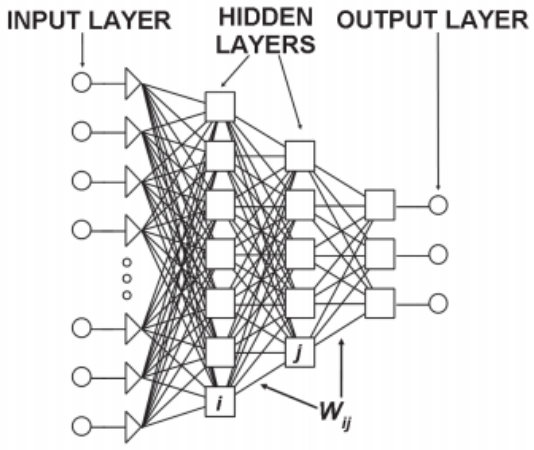
$$P(\text{outcome}|\text{evidence}) = \frac{P(\text{Likelihood of Evidence}) \times \text{Prior prob of outcome}}{P(\text{Evidence})}$$

Bayes has always been another popular machine learning approach to medical data, and many papers have justified the use of it. One such paper which looked at predicting heart disease [9], gives good reasons for the choice of the algorithm. When data amounts are high, this way of training and predicting does not exponentially increase in terms of time. Where attributes are independent of each other, Naive Bayes can handle it better than other algorithms (as shown in pre-processing, there are attributes in this paper's example for the need of this), and output is more efficient compared to other methods.

The final classifier that will be used in the prediction of thoracic surgery mortality risk will be with neural networks. In Weka, by default one neural network library comes pre-installed, Multilayer perception. Artificial neural networks (ANN) are a more recent development in medical diagnosis, and come from more of an artificial intelligence background. These are used widely in science, and are popular where attribute relationships may be unknown or very complex [10].

Looking at neural networks from a statistical point of view, these in essence are formed by an input layer, hidden layers and an output layer. The more complex the system studied, the more layers are added. Neurons in the input layer receive data to be trained on, and move them to the first hidden layer through weighted links. Once all data is passed through hidden layers, data is mathematically processed and reach the networks output. Figure 7 illustrates this.

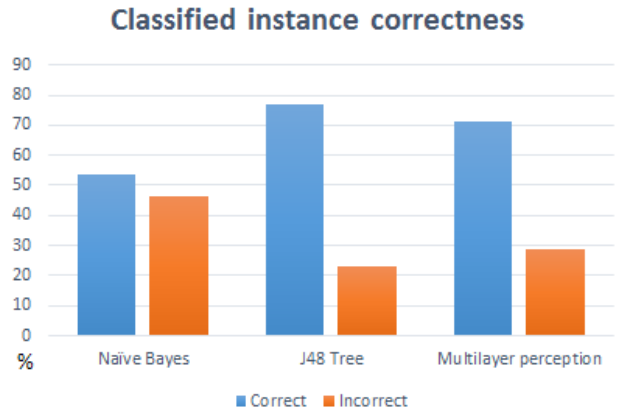
Fig. 7. Neural network layout. W is the weight of connection between j and i nodes.



B. Configuration

Each classifier has a set of hyper-parameters within Weka that can be adjusted according to the problem they face. By default, these are set in a way to be useful to the widest range of data sets, therefore may need to be changed to increase the effectiveness for the data tested in this paper. For each configuration change following, the configuration string can be found in Appendix C. Before each classifier is configured, the classifiers were tested using default options, and the statistic of how correct they were at predicting can be found in Figure 8.

Fig. 8. Initial run of classifiers using training data, with default parameters applied.



As shown in the Figure 8, Naive Bayes is the classifier with the weakest result as of yet. In Weka, without the use of filters or boosting, this classifier is limited in the number of configuration options available. Of the parameters that were tested, one known as supervised discretization had some positive effect to the amount of correctly classed instances [11]. This is a method of variable selection, and transforms continuous values of variables to discrete ones. This option has been shown to improve classifiers (in this case Naive Bayes) that are sensitive to high dimensional data. Table 1 shows the difference in the performance of this classifier with the addition of this method.

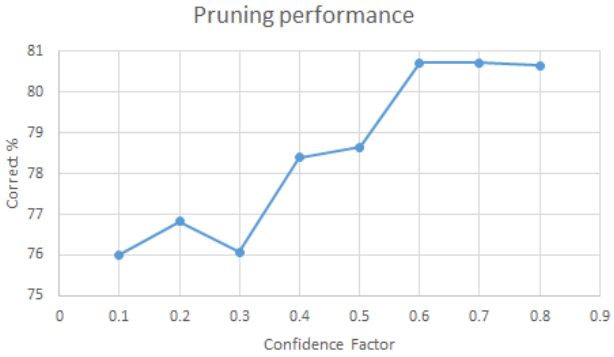
TABLE I. Comparison of predictiveness with discretization enabled or disbaled with the Naive Bayes classifier.

	Correctly classified	ROC curve
Without discretization	53.65%	0.745
With discretization	66.15%	0.663

The second of the classifiers, J48 in Weka carries more configuration options than that of Bayes, with most relating to the pruning of the tree. As mentioned earlier, pruning is performed after the tree is generated in order to remove outliers, and helps prevent under-fitting. Initially, the decision tree classifier was quite successful in classifying correctly, but with changes to the way it pruned, this could be refined.

Starting with the confidence factor, this was increased, which would mean that instances of predictions that had less confidence would be pruned less often. Although this meant the tree may then be more biased, the accuracy did in fact increase. Figure 9 shows the prediction correctness based on the confidence factors tested.

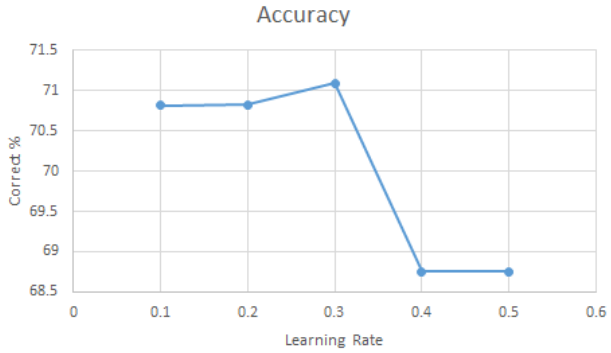
Fig. 9. Confidence factor increases showing change in accuracy of correct predictions.



As shown in the previous figure, a confidence factor of 0.6 was where the accuracy peaked. Whilst testing the different confidence factor values, it was noted that the time taken to run the classifier increased as confidence factors reached above 0.5. As for further configurations, none were found that could increase both the accuracy and the ROC curve values.

The final classifier (Multilayer perception) also had a strong selection of parameters available, but in this case none could help increase the accuracy further than the default state. By increasing the learning rate of the neural network, it was hoped that the classifier would learn faster, and therefore be more accurate towards the end of training. As shown in Figure 10 though, the increase in learning rate actually decreased the accuracy, and the same happened when decreasing the learning rate.

Fig. 10. Learning rate plotted against accuracy of correct predictions.



For this reason, this classifier remains unchanged, including the learning rate of 0.3, which already seems to be at its peak for accuracy. Although the neural network classifier could not be

configured directly, further adaptions to the algorithm could be performed either through boosting or further filtering. To test this and the other classifiers, an AdaBoost filter was wrapped around each and compared with the previous results (Shown in Table 2). Note: AdaBoost has been applied to all current configuration changes mentioned previously in an attempt to increase accuracy further.

TABLE II. Predictive correct percentages with and without AdaBoost. Computing time is shown after AdaBoost was applied.

	Without AdaBoost	With AdaBoost	Time to compute
Nave Bayes	66.15	67.71	<5 Seconds
J48 Tree	80.73	81.51	>2 Minutes
Multilayer perception	71.09	70.05	>1 Minute

As shown in the above Table 2, using AdaBoost has increased the effectiveness of two of the classifiers, though two of the classifiers also suffered heavily in terms of speed. With this taken into account, it was decided that only Nave Bayes benefitted from boosting, so was the only one to keep this as part of the algorithm. The configuratiomn can be found for this in Appenix D.

The final configuration change made was to try each classifier with a filter applied. Because the data had been pre-processed already, using a filtered classifier would be beneficial. In Weka, applying a filter is simple, and any classifier can be wrapped by it. The tests using this with each classifier (again, any configuration completed up to this stage has been taken into account) can be found below in Table 3.

TABLE III. Predictive correct percentages with and without filtering. Compute power remains unchanged.

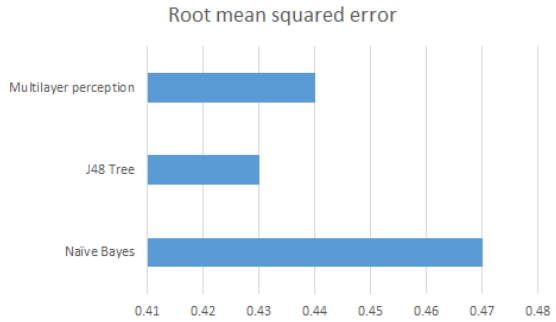
	Un-Filtered	Filtered	Time to compute
Nave Bayes	67.71	67.18	<5 Seconds
J48 Tree	80.73	69.51	<10 Seconds
Multilayer perception	71.09	71.88	<10 Seconds

The table shows that only the neural network classifier benefits from the filtered wrapper. Therefore, in Appendix E, the configuration for this algorithm is available. The other classifier algorithms will remain the same due to no need to add the filter. It can also be noted that unlike AdaBoost, the filter wrapper did not add any significant amount of time to compute predictions.

III. Results Discussion

Following pre-processing, classification configuration, and additions of filters/ boosting, the three can now be compared and evaluated. When using Weka, there are many statistics available following a test run. One of these is the root mean squared error. This statistic is a good general purpose error metric and amplifies large errors. Therefore, this is a good starting point to see how each of the tuned classifiers perform (Figure 11). It should be noted that the results shown in this section of the report were run in the Weka experimenter for efficiency of running multiple tests at once. As seen in the figure below, both the decision tree and Bayes methods seem most efficient.

Fig. 11. Squared error for all three classifiers.

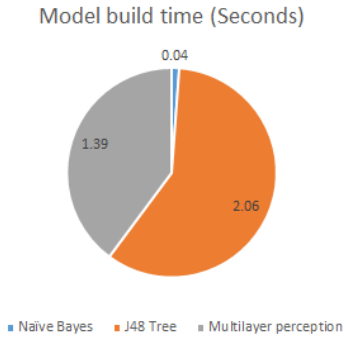


Whilst performing data pre-processing, it was decided that balancing should be performed, using the SMOTE filter. In the Figure below, the same algorithms have been used, but with the data firstly unbalanced, and then balanced. Surprisingly, balancing the data decreased the accuracy of prediction in all test cases for the classifiers in this paper. Therefore, moving forward, no filter will be applied at a pre-processing stage.

Fig. 12. Comparison with and without SMOTE filter for pre-processing.

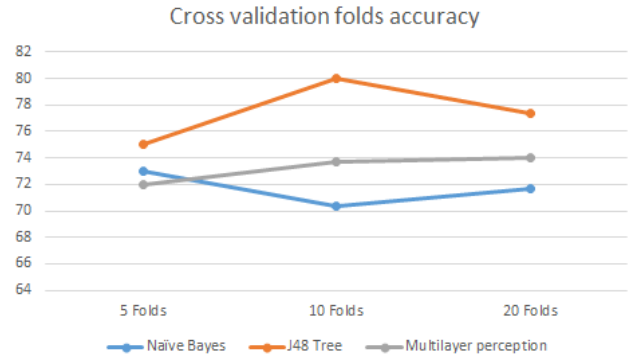
From a speed perspective, although this has been looked at during configuration, the time taken for each classifier has some importance when handling medical data. Many patients may be present, and the time would increase exponentially, so it is important a good classifier is also fast. Figure 13 shows the time taken to build a model of the data with each classifier. It can be seen that the fastest of the three is the Nave Bayes with AdaBoost. All three though are still relatively quick.

Fig. 13. Build time for all classifiers.



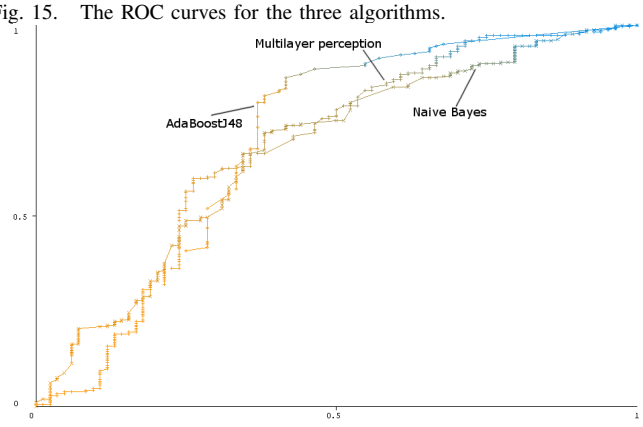
Whilst testing all configurations during experimenting, the amount of folds in cross validation has been kept at 10 to maintain fair testing. However, tests performed (Figure 14) with each classifier with an increase in folds (20) or a decrease in folds (5) showed that the local maximum in accuracy remained with 10 folds.

Fig. 14. Depiction of effect of accuracy with different amounts of cross-validation folds.



In order to help provide a clearer result of which classifier and algorithm is the best for this papers purpose, the Weka knowledge flow application was used to plot ROC curve areas, and also compare a range of statistics about the learning performed. Alike the experimenter application, this tool allowed multiple classifiers to be tested together, but Knowledge flow takes this further by allowing charts to show multiple classifiers at once. In Figure 15, the ROC curves are shown for each classifier.

Fig. 15. The ROC curves for the three algorithms.



From the above three, the boosted Nave Bayes classifier starts off the best when it comes to learning, but does drop slightly when more and more samples are evaluated. This classifier finishes on a ROC area of 0.669. Following this, the Filtered Multilayered perception classifier performs the weakest of the three (but with a slightly higher ROC) with a value ROC area value of 0.682. Finally, we can see in the diagram and from the ROC area value of 0.779.

Following the testing of all three classifiers, and with each being subject to different filters, dimensionality reduction and pre-processing, that the most appropriate classifier for the prediction of death rates a year after surgery is the J48 decision tree algorithm.

With the results and thus classifier selected, the suggested predictor that will be used on the test data can now be defined. The following rules should apply, and further instructions are provided in Appendix F. This algorithm should be:

- InfoGainAttributeEval should be used to rank attributes, and this found that attributes PRE4, PRE8 and PRE32 can be removed from the data set.
- No balancing is applied.
- J48 Decision tree classifier is selected, with 10 cross-validation folds.
- The only parameter to change is the confidence Factor, which should be increased to 0.6.

IV. Conclusion

Upon completion of training and predicting the risk of death based on data in the training and testing sets provided, a summary of the task can now be given. With no prior knowledge of Weka, statistics, or medical research, this paper gives a good insight to the uses of machine learning for efficiently predicting binary classes (though multiple classes will be applicable too) of information based on several attributes of data.

Although the final produced algorithm proved effective and quick at creating the predictions (which can be found in the predictions.csv file in this assignment folder), it can be reflected that no algorithms are multi-purpose. The J48 decision tree may be the best at solving this problem, but when it comes to more intensive data mining with very large amounts of data or more attributes, the tree may slow down and become more biased. This paper has however shown how to combat large attribute amounts with the use of dimensionality reduction and filtering.

Looking at the topic from a none-technical viewpoint, although predictions may be based from the data, this may not always tell a full story of a patient, and there is always some unpredictability when it comes to medical scenarios. To say something is 100% true is a very hard task, as external factors (for example, an unknown spread of the lung cancer prior to operation) can seriously affect the outcome of a test.

In summary, machine learning, with the use of tools such as Weka have proven to be a greater and greater force in recent years, and with the improvement of algorithms such as explored in this paper will increase alertness to many medical issues and risks in the future.

APPENDIX

All configurations have been run in Weka version 3.7.13 Appendices must be followed in order to be able to recreate results in configuration tuning or in result stage. Use 10 fold cross validation when classifying.

A. SMOTE data balancing

weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 100.0 -S 1

B. Data pre-processing/ attribute selection

weka.attributeSelection.InfoGainAttributeEval
weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1

Remove attributes from data after running: 2,6,15

C. Classifier configuration

All following configurations are after all changes have been applied to individual classifiers.

1) Naive Bayes: weka.classifiers.bayes.NaiveBayes -D

2) J48 Decision tree: weka.classifiers.trees.J48 -C 0.6 -M 2

3) Multilayer perception: weka.classifiers.functions.
MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

D. Naive Bayes with AdaBoost

weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.bayes.NaiveBayes – -D

E. Multilayer perception filtered

weka.classifiers.meta.FilteredClassifier -F
"weka.filters.supervised.attribute.Discretize
-R first-last -precision 6"
-W weka.classifiers.functions.MultilayerPerceptron – -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

F. Proposed solution

Follow these instructions to create the algorithm for predicting the results for the test set:

1) With the training set applied, apply an attribute evaluator:
   weka.attributeSelection.InfoGainAttributeEval
2) Remove attributes labelled 'PRE4', PRE8' and 'PRE32'
3) 10 Cross validation folds
4) Use the following classifier configuration:
   weka.classifiers.trees.J48 -C 0.6 -M 2

REFERENCES

[1] University of Waikato, *Weka 3: Data Mining Software in Java*, Machine Learning Group at the University of Waikato, *http://www.cs.waikato.ac.nz/ml/weka/*, 2013.

[2] Boston Medical Center, *Center for Thoracic Oncology*, Boston University school of medicine, *http://www.bmc.org/thoraciconcology/treatments/lung-resection.htm*, 2014.

[3] M. Rahman, *Machine learning based data pre-processing for the purpose of medical data mining and decision support*, Hull University, 2014.

[4] Microsoft, *SMOTE*, Microsoft Aure Modules, *https://msdn.microsoft.com/en-us/library/azure/dn913076.aspx*, 2016.

[5] N. Chawla et al, *SMOTE: Synthetic Minority Over-sampling Technique*, Journal of Artificial Intelligence Research 16 (2002) 321357. 2001.

[6] S. Reid, *Dimensionality Reduction Techniques*, Turing Finance, *http://www.turingfinance.com/*, 2014.

[7] DH. Wolpert, *The Lack of A Priori Distinctions Between Learning Algorithms*, Neural Computation. 1996;8:1341-1390. 1995.

[8] D. Lavanya, *Performance Evaluation of Decision Tree Classifiers on Medical Datasets*, International Journal of Computer Applications (0975 8887). 2011.

[9] R. R.Patil, *Heart Disease Prediction System using Naive Bayes and Jelinek-mercer smoothing*, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 5. 2014.

[10] F. Amato et al, *Artificial neural networks in medical diagnosis*, Journal of applied biomedicine 11: 4758. 2013.

[11] J. Lustgarten et al, *Improving Classification Performance with Discretization on Biomedical Datasets*, AMIA Annu Symp Proc. 2008; 2008: 445449. 2008.