THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Prediction of Oestrus in Dairy Cows: An Application of Machine Learning to Skewed Data

Adam D. Lynam

This thesis is submitted in partial fulfillment of the requirements for the Degree of Master of Science at the University of Waikato.

March 2009

Abstract

The Dairy industry requires accurate detection of oestrus(heat) in dairy cows to maximise output of the animals. Traditionally this is a process dependant on human observation and interpretation of the various signs of heat. Many areas of the dairy industry can be automated, however the detection of oestrus is an area that still requires human experts.

This thesis investigates the application of Machine Learning classification techniques, on dairy cow milking data provided by the Livestock Improvement Corporation, to predict oestrus. The usefulness of various ensemble learning algorithms such as Bagging and Boosting are explored as well as specific skewed data techniques.

An empirical study into the effectiveness of classifiers designed to target skewed data is included as a significant part of the investigation. Roughly Balanced Bagging and the novel Under Bagging classifiers are explored in considerable detail and found to perform quite favourably over the SMOTE technique for the datasets selected. This study uses non-dairy, commonplace, Machine Learning datasets; many of which are found in the UCI Machine Learning Repository.

Acknowledgements

I would like to thank my Masters supervisor, Bernhard Pfahringer, for putting up with me through the time it took to complete this work and for all the valuable help he provided throughout the thesis work.

I want to thank all the friends and family who supported me while I was working on the thesis. Most specifically, Helen Lynam, my Mum, and my two younger sisters: Sara and Emma Lynam. I really appreciated having such a great family to head home to when I needed to escape work.

I would also like to note two awesome flatmates I have lived with during most of my time at University and, most recently, while I was working on this thesis: Steven Loveridge and Scott Sutherland. I am glad to have had you guys around to keep me on track!

I feel its important to mention Russell Knutson, from LIC, for all his work in helping me get started and Tania Smith and Bevin Harris, also from LIC, for helping finalise everything on the LIC end of the work.

Contents

Chapter 1

Introduction

Machine Learning is an area of Artificial Intelligence which explores the application of algorithms that allow a computer to continuously refine its output parameters while observing incoming data, often in order to meet some target goal. Data Mining is the process of extracting patterns from data and the use of machine learning techniques allows the data extraction to be performed quickly by a computer and without direct human interaction throughout the process. Figure 1.1 shows the application of Machine Learning to the Data Mining process.

1.1 Research Goal

The goal of this investigation was to determine to what degree machine learning could be used to automate the detection of heat in dairy cows. Saitta and Neri[1] put forward a very convincing argument for conducting machine learning investigations in the "Real World", and, although it was not a direct inspiration, this investigation is certainly intended to be a step outside of the world of artificial data and into solving a real life problem.

Milking data was provided by the Livestock Improvement Corporation (LIC) at the beginning of the investigation. This data has had various manipulations applied to it over the time of the investigation, of particular note is the con-

Figure 1.1: A general overview of the application of Machine Learning to the Data Mining process.

version of the Firebird databases to MySQL databases. The specifics of these manipulations will be described as part of Chapter 5.

Having access to the expertise at LIC meant the investigation could be directed to a considerable degree, of particular importance was the selection of relevant attributes for learning a classifier model on. Full details on attributes selected will be provided in Chapter 5 as well.

It was known before this investigation began that a few problems were likely to cause trouble. The most easily foreseeable was that the data was coming from a real place; it was not going to be from an artificially created or complete source. Thus it was presumed that data would contain noise, noise being data which is not representative, not genuine. With this in mind a solution would have to be robust with regard to invalid or missing information. The issue of noise in the data is focused on in Section 5.4.

1.2 Explanation of Terms

In order to understand the chapters of this work it will be important to understand what is being referred to whenever any of the following terms are used.

Heat - The term **heat** is used to refer to a behaviour change in cows called Standing Heat. It is also commonly known as Oestrus or Estrus (an alternative spelling), and is the stage in the Oestrous/Estrous Cycle where a cow is

accepting of other animals to mount her and acts as a sign of sexual readiness.

Heat Event - A **heat event** is a recorded note in the LIC database where a cow has been noted to be in heat by some mechanism. These mechanisms are briefly outlined in Chapter 2.

Mating Event - A **mating event** is a recorded note, similar to that of a heat event, in the LIC database in which a cow has been inseminated as a result of identifying heat. Heat and mating events both represent indicators of heat, but through different systems.

1.3 Thesis Structure

This chapter was used to establish the context of the work. It included the goals of the investigation, a brief introduction to the LIC data the work is based around and a short glossary of terms important to the work.

Chapter 2 includes details of work that is related to this investigation. It begins with a general introduction to conventional heat detection in the dairy industry, moves on to explain, in some detail, the machine learning tool, WEKA, as well as the machine learning techniques relevant to the investigation. The related work chapter continues as it details some of the work performed by the Machine Learning discipline on other data from the agricultural sector. Finally Chapter 2 concludes with details of a very similar investigation completed in 1996.

Chapters 3 and 4 step back from the LIC data to explore the application of machine learning to skewed dataset problems in a general sense; the discoveries made will later be tied back to the main investigation during Chapters 6 and 7. The aim of this general investigation was twofold. The first aim was to explore the viability of the, novel, Under Bagging classification technique on skewed class problems using a number of publicly accessible datasets. The second aim

was to identify the most appropriate skewed data machine learning techniques, so that these techniques could be applied to the data in the main investigation; the detection of heat in cows.

Chapter 5 goes into considerable depth while explaining the pre-processing that was required to get the LIC data into a form where Data Mining techniques could be applied to it. These steps were: converting Firebird databases into MySQL databases by way of exporting and importing SQL statements, extracting values from the data that were not explicitly available, removing any obvious outlier values from the data, generating attributes as part of compressing multiple tables of data into a single flat file and, finally, generating ARFF files that could be loaded by WEKA.

Chapter 6 gives an overview of the experiments performed on the LIC data during the course of the investigation. The first six experiments show the evolution of the experimental process and detail when each step in the refinement process occurred. There are two additional experiments that were performed after the experimental process was refined. The first of these experiments was to quantify the usefulness of each attribute source used to generate attributes. The second experiment was the application of the most successful skewed data classifiers from the skewed data experiments in Chapter 4.

Chapter 7 contains the results of Experiments 6.1 - 6.8 from Chapter 6. Graphs of the classifiers with the highest overall performance and a table summarising the result values for all classifiers are given for each experiment. In addition to the experiments outlined in Chapter 6, Chapter 7 also includes a Section 7.9, which investigates the performance of one herd with particularly exceptional performance; this section attempts to find an explanation for the results achieved.

Chapter 8, the final chapter, summarises the entire investigation and mentions specific points relevant to anyone wishing to continue work in the area of predicting heat in cows.

Chapter 2

Related Work

2.1 The New Zealand Dairy Industry

The New Zealand dairy industry is New Zealand's largest revenue earner in exported goods; exports valued at $7.5 billon were sent overseas between July 2006 and July 2007[2]. According to New Zealand Dairy Statistics[3], in the 2007/2008 milking season, the New Zealand dairy industry was comprised of 4,012,867 cows; these cows were divided between 11,436 herds. 14,745 million (14.7 billion) litres of milk from these cows were processed by the industry over the season.

2.1.1 Daily Data Recording

In the modern New Zealand dairy industry cows are typically milked twice a day. In the data provided by LIC twice daily milking of animals allows for twice daily recording of data; there are limitations on what data can be recorded however. In many cases data cannot be recorded at a location because specialist equipment, required to conduct the measurement, is not available at the milking site. A basic example of this is requiring equipment capable of measuring a cow's weight in order to record a weight at each milking. It is also possible a measurement cannot be done at each milking due to time constraints; this would almost certainly be due to needing some significant

human input to measure or record the values.

When the equipment is available at a milking location, and there are no other limitations on the recording of data, values can be measured for each cow being milked. Once a value is measured it can be linked to a cow using their unique Radio-frequency Identification (RFID) tag number and recorded.

2.1.2 Detection of Heat Traditionally

Identifying heat in cows is currently done using visual observation by a skilled human and is supplemented by systems that tag notable cows that are behaving as if in heat for the skilled person to pay more attention to.

Identifying heat has never been an exact science, as such there are many specific methods a skilled person can use to identify heat. Foote[4] discusses some methods for identifying heat, however the most notable method for doing so is observing cows being mounted by other animals either directly (during milking time for example) or indirectly by methods such as observing ruffled hair at the cows rear end or noting a rubbing of **tail paint** some time after a mounting has happened.

2.2 Machine Learning Generally

2.2.1 WEKA

This investigation made heavy use of the WEKA Machine Learning suite explained in detail by Witten and Frank[5].

The WEKA implementations of the classifiers outlined below have been used throughout the investigation as part of various discussed as well as many additional practice experiments. In addition to the important use of classifiers, data was represented in WEKA's ARFF file format to allow manipulation using some of the powerful filters available in WEKA and create refined datasets.

The WEKA Explorer included in the suite was used extensively as part of trial runs of experiments, checking ARFF file integrity and running WEKA filters to refine datasets. The Explorer was indispensable in this regard as the ability to quickly perform a mock experiment or remove attributes from an ARFF file saved much time in coding or running an experiment with improper parameters specified.

2.2.2 ARFF File Format

An ARFF (Attribute-Relation File Format)[5] file is a plain text, human readable, file. The ARFF file format was created to supply instance data to the WEKA Machine Learning suite. An ARFF file always begins with an ordered declaration of attributes followed by the learning instances with their associated, comma separated, attribute values; each instance takes up a single line in the file.

The ARFF file format allows for the specification of missing values, using the question mark placeholder. This is important for this investigation as values are often unspecified in the database; meaning there are gaps in the data. Classifiers and filters still need to function when there are values missing; the question mark allows these values to be represented without disrupting functionality.

Classifiers will have their own way of dealing with missing values. These solutions can range from a crude approach, such as applying an average value for the attribute to any missing values, to a more sophisticated approach, like that of the J48 Decision Tree; Quinlan[6] explains how an instance with a missing value can travel down all branches of a tree splitting on the missing attribute but with its weight divided between the branches.

2.2.3 Classifiers

Within Data Mining there are a branch of problems known as classification problems, this investigation has chosen to approach the detection of heat as a classification problem and used the tools available in WEKA to achieve this goal.

Classification problems involve a defined set of classes or labels, usually at least two but sometimes just a single class is defined. Instances, another part of classification problems, are made up of a combination of attributes, these attributes are used to represent the current state of some system, and the specific nature of the attributes will depend on the problem. In this investigation the instances are made up of attributes which represent the milking of a specified dairy cow milking at a specified time on a specified day and the classes are *In Heat* or *Not in Heat*.

The act of assigning a class label to an instance of data is known as classification and this is performed by a set of computer code instructions called a classifier. In a general sense at least, all classifiers share the goal that they need to learn how to classify instances given what is known as a training set, a set of instances where the class labels are provided. There are many varied approaches to learning a model from provided data; even the way a classifier uses a model to assign a class labels can vary from very basic to very complex approaches.

2.2.3.1 J48 (C4.5)

J48 is the name for the implementation of the C4.5[6] decision tree generation algorithm in WEKA. C4.5 is a relatively basic machine learning classifier, but despite its simple methods it is very robust and can often compete well with other, more complex classifiers, especially when used as part of ensemble classifiers such as the later discussed bagging and boosting classifiers used in

this investigation.

At each node in the construction of the tree the C4.5 algorithm determines which of the attributes available provides the best option to split the training instances to maximise the information gain metric. Based on the splitting attribute chosen by the gain metric the instances are divided into new sets. Each set follows a different branch and will go on to create new tree branches with their own split points for the remaining instances in the set. Once a branch contains only instances of a single class a leaf node is created and branching ceases, alternatively if the information gained from a split is below a given threshold then a leaf is created for the majority class.

Once C4.5 has built a full decision tree using the above method there are options to prune leaf nodes back to avoid tailoring the tree too closely to the given training data, this is a powerful way to avoid becoming too influenced by noise in the training data.

2.2.3.2 Naïve Bayes

Naïve Bayes[7] is an extremely simple classifier based on Bayes Theorem[8]. The Naïve Bayes classifier is known in Machine Learning to give surprisingly accurate predictions[9, 10] despite being algorithmically simple and working on the, often false, assumption that all attributes in the data are statistically independent from each other.

The power of Naïve Bayes lies in the statistical independence assumption. When Bayes Theorem is applied to calculate the probability model for a prediction class, it would normally produce a complex equation; with consideration for all the relationships between attributes included. However, with the independence assumption, this can be expressed as a much simpler equation using a single term for each attribute. This simple equation can be computed very quickly.

Finally, to build a Naïve Bayes classifier out of the probability models for each class, an aggregation method is needed to combine the models into a single prediction. This is typically as simple as predicting the class for whichever probability model gives the greatest result for a given set of values for the attributes of the problem.

2.2.3.3 Averaged One-Dependence Estimators

Averaged One-Dependence Estimators (AODE)[11] is a classification technique which is very similar to Naïve Bayes; it is also based on the application of Bayes Theorem[8]. Essentially there are two differences to Naïve Bayes in AODE: the independence assumption is weakened and the aggregation technique for probability models is more complex.

The intended goal of the classifier is to get away from the independence assumption of Naïve Bayes. However, in order to take advantage of some the computational efficiency of Naïve Bayes, this assumption is simply weakened to allow one level of attribute dependence per probability model. There is a problem introduced by allowing one level of attribute dependence; there is more than one probability model for each class value.

In order to deal with more than one probability model AODE can aggregate any one-attribute dependence models it builds; combining them into one probability. With a probability for each class, selecting the class with the highest probability gives a classification.

2.2.3.4 Logistic Regression

Logistic Regression[12] is a classifier which applies a learning process to calculate a set of coefficients; a coefficient is generated for each attribute in a problem as well as an additional one that is not tied to an attribute. These coefficients are multiplied by their respective attribute values and summed to give the risk factor component of the Logistic Function[13]. The 0-1 value

produced by the Logistic Function, after thresholding, can be interpreted and used for classification purposes.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k \tag{2.2}$$

The Logistic Function Equation 2.1 and the Risk Factor Equation 2.2 are given. The risk factor component, z, is calculated by multiplying the β coefficients, by each of their respective x attribute values, for each of the k attributes from 1 to k; the β coefficients are learned by running the Logistic Regression classifier. The final step in calculating the risk factor is adding on the β_0 coefficient that is not associated to an attribute. Once the risk factor component is calculated the Logistic Function can be evaluated to produce a value between 0 and 1; this value can be used for prediction in machine learning.

2.2.3.5 Support Vector Machines

Support Vector Machines (SVM)[14] are machine learning classifiers which represent the separation of class space into areas divided by hyperplanes. Hyperplanes are simply the equivalent of lines on a two dimensional graph but for many more dimensions (one for each attribute being classified to be exact). This hyperplane is actually defined using the closest learning instances and their associated class; these instances are the support vectors which are needed to calculate the position of the hyperplane.

A support vector machine will attempt to identify what is known as a maximum-margin hyperplane, the single hyperplane that maximises the distance from every class in order to minimise the chance of misclassification. However in real world examples a compromise may have to be made as data may contain conflicting training instances. The Sequential Minimal Optimization (SMO)[15] classifier which is available in WEKA was the specific support vector machine implementation used in this investigation.

A transformation kernel can be added to a support vector machine to allow the linear support vector machine to represent its attributes in a higher dimensional or feature space. Feature space allows for more complex attributes; combinations of attributes or transformations to initial attributes are possible. Using attributes from feature space it may be possible to find a linear solution to a problem that corresponds to a non linear solution in normal attribute space. The Radial Basis Function (RBF)[16] kernel was used with SMO in this investigation.

2.2.3.6 Bagging

Bagging[17] is a type of ensemble machine learning classifier; it is called an ensemble because it works by constructing more than one model and combines the results to make a single classification based on an equally weighted vote between all the models built. Specifically bagging repeatedly uses a specified classifier for a given number of iterations with a different set of the training instances each time. For each iteration, a random sample of training instances is drawn with replacement.

Bagged Unpruned J48 Trees was the specific type of bagging used in this investigation. Using J48 trees with pruning options disabled means that this bagging classifier will be executing a process very similar to the Random Forest method. One major difference is that each J48 tree will consider all attributes to split on at each node in the tree instead of only using a subset like Random Forest.

2.2.3.7 Random Forests

Random Forest[18] is a machine learning classifier that works a little like the ensemble classification methods do, it works over many iterations of the same technique but with a different approach. The basic idea is that by repeatedly building decision trees with randomly generated subsets of learning instances

and limited options on attributes to build on at each point in tree construction, each tree has a greater chance of capturing something the others will not.

An implementation of the Random Forest algorithm involves sampling a limited number of instances from the set of learning instances with replacement; which means an instance can be drawn more than once. That set of drawn instances is used to build an unpruned decision tree. Each time a node needs to be built in the tree, another subset, this time of the available attributes, is drawn to compile a list of attributes over which an attribute information gain metric or similar is run to select the attribute to split on.

2.2.3.8 AdaBoostM1

AdaBoostM1[19] is a type of boosting ensemble classifier. Boosting is the process of combining many weak classifiers in some fashion in order to create an overall strong classifier when acting together. AdaBoost, specifically, works by building many iterations of a specified classifier. After each iteration is completed, the training instances are re-weighted according to the effectiveness of the classifiers built so far. If an instance is classified incorrectly by enough of the classifiers, it will get a higher weight to increase the chance subsequent classifier iterations will focus on that instance. The idea being that specific iterations of the base classifier will be given a set of learning instances to focus on learning an accurate model for instead of learning a less accurate model on the entire training set.

AdaBoostM1 combines all the classifier models together in a weighted vote in order to perform an overall classification; this means that some models will contribute more to the collective vote than others if they have a higher weight. The weight for a model is higher if it managed to accurately predict training instances that other models struggled to, thus the models that work best on instances that were difficult to classify contribute more to the final classification.

AdaBoosted J48 Trees and AdaBoosted Decision Stumps were used in this investigation. AdaBoosted J48 Trees use the AdaBoostM1 algorithm with the J48 classifier as the weak learner that is repeated. AdaBoosted Decision Stumps uses the AdaBoostM1 algorithm with a basic classifier that builds a very small tree called a decision stump. A Decision Stump is actually just a single split point on a single attribute, but the AdaBoost algorithm repeats this process many times which can produce an overall powerful classifier.

2.2.4 Precision-Recall and ROC Curves

In Machine Learning there are many metrics for measuring classifier performance over a set of data. Selecting an inferior metric for a particular problem can make results appear optimistically positive, where in reality the positive results simply mask a poor model.

Choosing a suitable metric was important for this investigation as it was an attempt to find information in real data, optimistic theoretical results would waste time should an implementation of the techniques used ever be attempted. What was needed was a metric which clearly captured the effectiveness of machine learning on this data.

Receiver Operator Characteristic Curves (ROC Curves), specifically the Area Under the Curve (AUC) for the ROC has been used as a comparison metric in Machine Learning since Spackman[20] first illustrated their use. ROC curves are created by plotting the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis while adjusting the threshold value for what is considered a positive classification. Results will compare ROC AUC values as part of evaluation. Further details on the effectiveness of ROC area as a comparison metric can be found in work by Bradley[21].

However, as Davis and Goadrich discuss[22], there are merits to comparing Precision-Recall values when dealing with skewed data; the Precision-Recall

AUC metric is likely the most suitable for this data for this reason. If ROC AUC was used alone, it would give optimistic results due to the negative class making up a significant majority of the data available. Precision-Recall is similar to ROC in that the True Positive Rate is still used, however on the x-axis this time, and is titled Recall. The difference lies in the y-axis representing Precision, a value calculated by the number of True Positives divided by the total number of positively classified instances.

2.3 Agricultural Industry and Machine Learning

McQueen, Garner, Nevill-Manning and Witten[23] explore the application of machine learning to real world agricultural data in the form of animal culling records. Their goal was to use machine learning techniques to predict which animals in a herd would be most sensible to cull, productively speaking. The work was conducted with specific focus on C4.5 and FOIL classifiers and showed that pre-processing of the data provided was a key factor in the positive result they obtained. Results achieved were positive, 95% classification accuracy was achieved with 30% of available instances being used for training, and the remaining 70% being used for testing the model constructed. The decision tree built to achieve these results was simple and easily human interpretable.

Additionally as part of continued work in the area of agriculture in machine learning at Waikato University Garner, Cunningham, Holmes, Nevill-Manning and Witten[24] have compiled further notes on their work with the dairy cow culling database mentioned above. This work reinforces the importance of pre-processing data and working closely with experts in the field. They also stress the importance of having a powerful organisational system for mapping the many attribute combinations to experiments. When working with agricultural data an investigator will undoubtedly run many experiments with potential

improvements to be gained after each step in their process.

2.4 Detection of Heat in Machine Learning

There is one investigation of considerable relevance that was published in 1996 by Mitchell, Sherlock and Smith[25]. Their investigation was into the application of classifiers with human comprehensible output, namely C4.5 (J48 in WEKA) and FOIL classifiers, to the domain of detecting oestrus (heat) in dairy cows.

Their investigation contained four experiments, the first three of which are directly relevant to this study:

Experiment 1 contained instances created with a 3 day sliding window. The attributes measured over each window were the milk volume deviation from the herd milk volume mean, the order the cows were milked in, recorded as a percentage of the way through the milking, and the number of days since the last heat event. A generated instance was positive if a heat event was recorded on the last day of the sliding window. Notably, any instances that were generated within 24 hours of a known heat event were removed in an attempt to remove noise from the training set. The best results on the test data were unquestionably with C4.5 rules with 41.0% of the heat events correctly classified and 34.7% non-heat events incorrectly classified as heat events.

Experiment 2 was mostly the same as the first experiment, but this time the instances close to heat events that were removed were now included in the training set. If the milk volume deviation was within one standard deviation of the running herd mean then it was considered a non-heat event. If there was more than one standard deviation difference, the instance was treated as a heat event. Additionally, in the testing phase, if a non-heat event was classified as a heat event but lay within 24 hours of a heat event then it was

considered correct. The best result on the test data was obtained with C4.5 rules again; 44.1% of heat events classified accurately with only 20.1% of non-heat events misclassified as heat events. This was an improvement over the first experiments results.

Experiment 3, the final relevant experiment, was different from the other two experiments in that it used a 5 day sliding window and calculated normalised differences from herd means for both milk volume and cow milking order. The treatment of instances within 24 hours of a heat event was mostly similar to experiment 2. The difference was, that in order to be labelled as a heat event, the volume deviation now had to be over two standard deviations from the mean instead of just one. Results for the, arguably, best performing classifier, C4.5 rules, varied from the first two experiments. Classifying 68.7% of heat events accurately was certainly a positive result, but the accompanying 73.7% of non-heat events that were incorrectly classified puts this in perspective.

The results for Experiment 3 are debatably the best they achieved and while there is a large percentage of failed classifications, their investigation explains that a failed insemination based on a false classification would incur a relatively low cost compared to the cost of missing an instance of heat altogether.

Chapter 3

Skewed Class Problems

A significant number of problems in Machine Learning can be seen to have skewed distributions of learning examples. Skewed distribution problems are characterised by an unequal split between classes for examples. This split is almost always present in both the training and any test datasets. The skew of this split could range from a very small skew in a two class problem, where one class contains 40% of the examples and another contains the remaining 60%, to a very large skew in a two-class problem, where one class contains 0.3% of the examples and the other contains the far more significant 99.7% of remaining examples. Though the skew in a problem can be small, most problems would not be called skewed unless there was at least a 1 to 10 split between two classes. Many datasets based on real world data end up with skewed class distributions.

Japkowicz[26] justifies alternative approaches to skewed class problems by showing that it is certainly possible to get improved results with techniques designed specifically to deal with class imbalance. Random over- and under-sampling are mentioned specifically and it is shown that more sophisticated techniques, such as selection of non-outlier examples during the re-sampling process, are unnecessary for improved results.

There are two distinct approaches in Machine Learning for dealing with skewed data.

The first chooses to assign costs to each class (or example directly), effectively weighting each example prior to classification. This approach can be quite intuitive if there is an easy to calculate cost for each class or example, but it can often be tough to assign a cost to at least one of the classes.

Pazzani et al.[27] offer a good introduction to this way of dealing with skewed class distribution problems and they explore a few implementations of this approach. However, they got disappointing results when their attempts to reduce the costs of misclassification errors actually resulted in higher costs. Positive results with cost sensitive classification can be found in Domingos[28] work. His MetaCost algorithm can reassign class labels to training examples, based on the cost metric, so that a classifier applied to it will be encouraged to better model the majority examples. In this way it blurs the line between the two approaches to classifying skewed problems.

The second approach, and the one explored in this investigation, is that of using controlled selection of examples to remove the effects of skew. This could range from anything as basic as selecting fewer majority class examples, so skew is reduced, to a literally generating new, synthetic, minority class examples to boost their presence.

Kubat and Matwin[29] explore the idea of under-sampling of the majority class over under-sampling of the entire set of data. They call it One-sided Selection because they are selective about which class to under-sample. Their results show that classifiers that were normally sensitive to skewed data could better deal with this type of data once their One-sided Selection algorithm was applied.

Algorithm 3.1 Pseudocode for Unskewed Sampling

Require: Set of all minority class training instances $MinorityInstances$;
Set of all majority class training instances $MajorityInstances$;
The number of minority class examples to draw $MinorityFraction$ (represented as a fraction of the number of minority class examples, $MinorityFraction \geq 0$);
The number of majority class examples to draw $MajorityFraction$ (represented as a fraction of the number of majority class examples, $MajorityFraction \geq 0$)

Ensure: A set of re-sampled training instances

1: $MinorityClassGoal \Leftarrow size(MinorityInstances) \times MinorityFraction$
2: $MajorityClassGoal \Leftarrow size(MajorityInstances) \times MajorityFraction$
3: **for** $i \Leftarrow 0$; $i < MinorityClassGoal$; $i \Leftarrow i + 1$ **do**
4: Add a random instance from $MinorityInstances$ to $UnskewedInstances$
5: **end for**
6: **for** $j \Leftarrow 0$; $j < MajorityClassGoal$; $j \Leftarrow j + 1$ **do**
7: Add a random instance from $MajorityInstances$ to $UnskewedInstances$
8: **end for**
9: **return** $UnskewedInstances$

3.1 Classifiers Designed to Target Skewed Data

As skewed problems are common in Machine Learning there has been significant work in creating techniques which can deal with skewed data appropriately. In WEKA these techniques take the form of meta classifiers which work on the skewed data, before handing it off to a standard base classifier(s).

This investigation explored the application of four of these meta style classifiers specifically designed to work on two class skewed distribution problems: Unskewed Sampling, SMOTE, Roughly Balanced Bagging and Under Bagging.

3.1.1 Unskewed Sampling

The first, and most basic, classifier allows for only the two most basic transformations of a skewed dataset. These transformations are under-sampling of the majority class, the same as One-sided Selection discussed by Kubat and Matwin[29] and over-sampling of the minority class. Collectively these two

will be known as Unskewed Sampling. With this basic meta classifier it is possible to reduce the number of majority class examples and/or duplicate minority class examples in order to get a more even class distribution in two class problems.

3.1.2 SMOTE

SMOTE (Synthetic Minority Over-sampling TEchnique), is a method of dealing with class distribution skew in datasets designed by Chawla, Bowyer, Hall and Kegelmeyer[30]. They show SMOTE to be more successful than under-sampling alone and, in many of their experiments, better than modified Naïve Bayes and Ripper implementations (altered to perform better on skewed data). In order to increase the presence of minority class examples in the problem the SMOTE process generates brand new minority class examples using the set of minority training examples as a base. It is somewhat similar to over-sampling of the minority class, except that entirely new, synthetic, examples are generated instead of only cloning existing ones.

Each SMOTE step works by selecting an existing minority example as a base and then selects, by way of a user defined parameter, a number of the example's nearest neighbors. From here, the original algorithm selects a neighbor at random. Finally for each attribute pair in the neighbor and base example a new synthetic attribute value is generated which falls between the two attribute values. One approach to generating each new attribute value is to calculate the difference in attribute values between the two examples, multiply it by a random number between 0 and 1, then add that value to the lower of the two existing attribute values.

This investigation has added two additional options to the WEKA implementation of SMOTE:

1. The first is an optional check that any synthetic examples generated are

Algorithm 3.2 Pseudocode for SMOTE

Require: Set of all minority class training instances $MinorityInstances$;

Set of all majority class training instances $MajorityFraction$;

The amount of SMOTE to perform $SMOTEAmount$ (represented as a fraction of the number of minority class examples, $SMOTEAmount \geq 0$);

The number of nearest neighbors to consider in the SMOTE step k ($k > 0$)

Ensure: A set of training instances with SMOTE generated minority examples added

1: Add all $MinorityInstances$ to $SMOTEInstances$

2: Add all $MajorityFraction$ to $SMOTEInstances$

3: $MinorityGenerationGoal \Leftarrow size(MinorityInstances) \times SMOTEAmount$

4: **for** $i \Leftarrow 0$; $i < MinorityGenerationGoal$; $i \Leftarrow i + 1$ **do**

5: $BaseInstance \Leftarrow$ A random instance from $MinorityInstances$

6: $InstanceNeighbors \Leftarrow$ The k nearest neighbors for $BaseInstance$

7: Add the result of $SmoteExampleGenerator(BaseInstance, InstanceNeighbors)$ to $SMOTEInstances$

8: **end for**

9: **return** $SMOTEInstances$

{$SmoteExampleGenerator$ - Performs the SMOTE instance generation}

Require: The first instance $BaseInstance$;

The k nearest neighbors for the passed instance $InstanceNeighbors$

Ensure: A brand new generated instance

10: $RandomNeighbor \Leftarrow$ A random neighbor from $InstanceNeighbors$

11: $NumberAttributes \Leftarrow$ The number of attributes present in $BaseInstance$

12: **for** $i \Leftarrow 0$; $i < NumberAttributes$; $i \Leftarrow i + 1$ **do**

13: $BaseValue \Leftarrow$ The current attribute value for $BaseInstance$

14: $NeighborValue \Leftarrow$ The current attribute value for $RandomNeighbor$

15: Generate a random number matching the condition ($0 \geq RandomNumber < 1$)

16: $NewAttribute \Leftarrow NeighborValue - BaseValue \times RandomNumber$

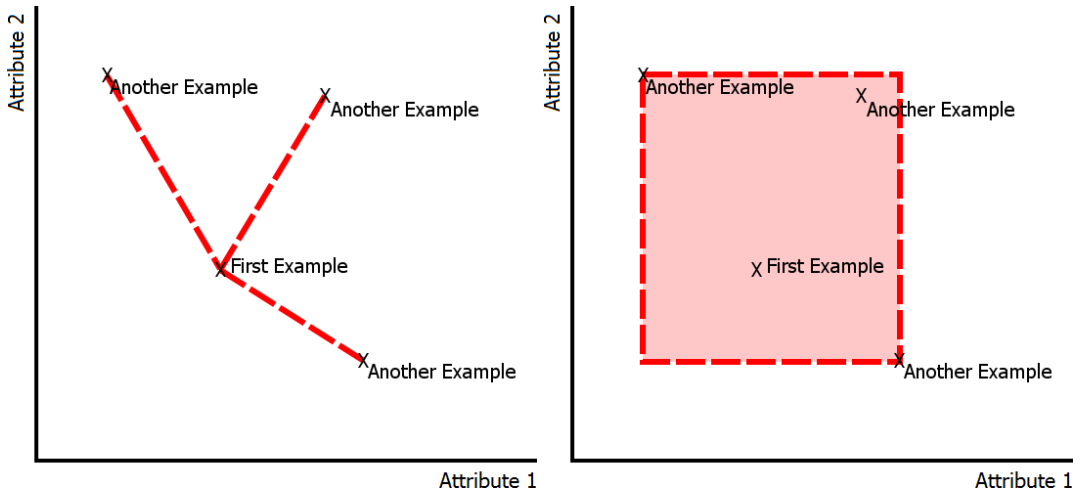
17: Add $NewAttribute$ to $GeneratedInstance$

18: **end for**

19: **return** $GeneratedInstance$

actually still representative of the minority class. This is confirmed by checking that the nearest neighbors for the new example do not include any majority class examples. This option is known as Synthetic Example Protection.

2. The second option is called the Neighbor per Attribute approach. The purpose of this option is to allow for more unique examples to be generated in the SMOTE step. This is accomplished by allowing the SMOTE step to select a different random neighbor for each attribute instead of simply selecting a single neighbor to generate all attributes from. This can be conceptualised as the difference between picking a random point inside a multi-dimensional hyper sphere and picking a random point along a line between two points. Figure 3.1 shows a comparison between the Neighbor per Attribute approach and the single attribute generation approach used in the original SMOTE.



(a) Unmodified SMOTE generation space

(b) Generation space for SMOTE with the Neighbor per Attribute approach applied

Figure 3.1: Using simplified instances with two attributes the difference achieved through using the Neighbor per Attribute approach can be observed. Figure (a) shows where unmodified SMOTE could generate an example from. Figure (b) shows where SMOTE using the Neighbor per Attribute approach could generate an example from.

3.1.3 Roughly Balanced Bagging

Roughly Balanced Bagging is another technique designed to deal with skewed class distributions. Hido and Kashima[31] are the creators of the Roughly Balanced Bagging classifier where the idea is to apply under-sampling of the majority class in a controlled bagging setting. The results achieved in their investigation indicated their Roughly Balanced Bagging classifier was capable of outperforming AdaBoost and RIPPER techniques on the nine datasets they applied classifiers to.

Using a Bagging framework, Roughly Balanced Bagging builds a specified number of instance sets for a number of base classifier iterations to learn from. In order to work better on skewed datasets, Roughly Balanced Bagging uses under-sampling of the majority class to get a new, less skewed, distribution of classes. However, it does not simply apply the method used by Unskewed Sampling, instead, it uses a negative binomial sampling technique and probability threshold to control how many examples of each class make it into each set of instances. The result is that each of the Bagging iterations will contain almost the same number of instances from each class, but it will vary a little bit with each one. Hido and Kashima[31] argue their technique better mimics the intentions of the basic Bagging classifier.

Roughly Balanced Bagging really only has one parameter over the standard Bagging classifier; the minority example threshold. This value, between 0 and 1, is set to give the chance of drawing a minority class example on each step of the drawing process. The drawing stops when the size of the set of minority class examples drawn reaches the number of the minority class training examples available (this does not mean all the minority examples are present as selection is performed with replacement), a lower value will mean more majority class examples make it in, whereas a higher value will mean less majority class examples make it into each set of instances. The default

Algorithm 3.3 Pseudocode for Roughly Balanced Bagging

Require: Set of all minority class training instances $MinorityInstances$;

Set of all majority class training instances $MajorityInstances$;

The chance of selecting a minority example $MinorityChance$ (represented as a probability, $0 < MinorityChance \leq 1$)

Ensure: A set of re-sampled training instances for an iteration of the base classifier

$MinorityClassGoal \Leftarrow size(MinorityInstances)$

$MajorityClassGoal \Leftarrow$

$RoughlyBalancedBaggingMajorityGoalCalculator(MinorityClassGoal, MinorityChance)$

for $i \Leftarrow 0$; $i < MinorityClassGoal$; $i \Leftarrow i + 1$ **do**

    Add a random instance from $MinorityInstances$ to $BalancedInstances$

end for

for $j \Leftarrow 0$; $j < MajorityClassGoal$; $j \Leftarrow j + 1$ **do**

    Add a random instance from $MajorityInstances$ to $BalancedInstances$

end for

return $BalancedInstances$

{$RoughlyBalancedBaggingMajorityGoalCalculator$ - Replicates the negative binomial sampling part of Roughly Balanced Bagging}

Require: The number of minority class examples being drawn $MinorityClassGoal$; The chance of a minority class example being selected $MinorityChance$

Ensure: The number of majority class example to draw

$MinorityCount \Leftarrow 0$

$MajorityCount \Leftarrow 0$

while $MinorityCount < MinorityClassGoal$ **do**

    Generate a random number matching the condition ($0 \geq RandomNumber < 1$)

    if $RandomNumber < MinorityChance$ **then**

        $MinorityCount \Leftarrow MinorityCount + 1$

    else

        $MajorityCount \Leftarrow MajorityCount + 1$

    end if

end while

return $MajorityCount$

value of 0.5 will produce instance sets with almost exactly the same number of majority examples as minority examples.

3.1.4 Under Bagging

Under Bagging is essentially a simplified version of the Roughly Balanced Bagging technique. It was developed during this investigation to deal with skewed class distribution problems specifically. This is accomplished in much the same way as Roughly Balanced Bagging, but without the focus on maintaining the Bagging style of instance selection. The result is a more predictable classification scheme than Roughly Balanced Bagging, but with the remaining advantages of the Bagging ensemble method.

Under Bagging, like Roughly Balanced Bagging, uses the Bagging process inherently. This means that any specified number of under bagged iterations can be performed using the Under Bagging instance selection method. However, unlike Roughly Balanced Bagging, Under Bagging uses a method far more comparable to Unskewed Sampling to select instances for each of its iterations. This method involves sampling with replacement from the minority and majority class to create a user-specified ratio of minority to majority class examples.

Like Roughly Balanced Bagging, Under Bagging adds only a single extra parameter addition over a standard Bagging implementation. However this parameter works differently due to Under Bagging always using a fixed minority class example size; the size of the original set of all minority class examples. The bag size factor parameter allows for selection of the number of majority class examples in each iteration of bagging as a factor of the number of minority class examples. A bag size factor value of 1.0 would mean each iteration of the base classifier would have an equal number of majority and minority class examples. The size of the training set in each run of the classifier would be twice the number of minority examples available. By raising (more majority

Algorithm 3.4 Pseudocode for Under Bagging

Require: Set of all minority class training instances $MinorityInstances$;

Set of all majority class training instances $MajorityInstances$;

The number of majority class examples to draw $MajorityFraction$ (represented as a fraction of the number of minority class examples, $MajorityFraction \geq 0$)

Ensure: A set of re-sampled training instances for an iteration of the base classifier

1: $MinorityClassGoal \Leftarrow size(MinorityInstances)$
2: $MajorityClassGoal \Leftarrow size(MinorityInstances) \times MajorityFraction$
3: **for** $i \Leftarrow 0$; $i < MinorityClassGoal$; $i \Leftarrow i + 1$ **do**
4: Add a random instance from $MinorityInstances$ to $UnderBaggedInstances$
5: **end for**
6: **for** $j \Leftarrow 0$; $j < MajorityClassGoal$; $j \Leftarrow j + 1$ **do**
7: Add a random instance from $MajorityInstances$ to $UnderBaggedInstances$
8: **end for**
9: **return** $UnderBaggedInstances$

class) or lowering (less majority class) the ratio of minority to majority class examples can be altered.

Chapter 4

Experiments on Skewed Class Problems

For comparing the simple Under Bagging technique that has been developed during this investigation to the existing classification techniques described in Chapter 3, experiments were performed on a number of commonly used datasets in Machine Learning.

4.1 Two-Class Skewed Datasets Used

These datasets either have a skewed class distribution by default or have been constructed to have one for the purposes of exploring skewed data classification.

It is important to note that the datasets used contain only two classes, known as the minority (positive) class or the majority (negative) class. Some of these datasets have more than two classes in their original form. For some of these, two classes have been selected to act as the minority and majority class in the new two class dataset, the smaller obviously being the minority class. In other cases a single class has been selected as the minority class and all other classes make up the majority class collectively. The latter can be easily identified in that they explicitly mention predicting one class versus *Others*. Table 4.1 contains a breakdown of class representation in the datasets used.

Dataset	Positive	Negative	Ratio
Abalone (Oldest Vs Others) [32]	36	4141	1 : 99
Abalone (Youngest Vs Others) [32]	74	4103	2 : 98
Adult (>$50K Vs <$50K) [32]	7841	24720	24 : 76
Anneal (Class#5 Vs Others) [32]	60	738	8 : 92
Car (VeryGood Vs Others) [32]	65	1663	4 : 96
Glass (Class#7 Vs Others) [32]	29	185	14 : 86
Haberman (Died Vs Survived) [32]	81	225	26 : 74
Hepatitis (Die Vs Live) [32]	32	123	15 : 85
Hypothyroid (Primary Hypothyroid Vs Others) [32]	95	3677	3 : 97
ICDM08 Full (Base Vs Explosion) [33]	623	8072	7 : 93
ICDM08 StationV (Base Vs Explosion) [33]	115	1589	7 : 93
ICDM08 StationW (Base Vs Explosion) [33]	14	763	2 : 98
ICDM08 StationX (Base Vs Explosion) [33]	210	2090	9 : 91
ICDM08 StationY (Base Vs Explosion) [33]	40	1169	3 : 97
ICDM08 StationZ (Base Vs Explosion) [33]	244	2461	9 : 91
Ionosphere (Bad Vs Good) [32]	126	225	36 : 64
Magic (Hadron Vs Gamma) [32]	6688	12332	35 : 65
Phoneme (Oral Vs Nasal) [32]	1586	3818	29 : 71
Pima Indians Diabetes (Positive Vs Negative) [32]	268	500	35 : 65
Satimage (Cotton Crop Vs Others) [32]	479	3956	11 : 89
Satimage (Damp Grey Vs Others) [32]	415	4020	9 : 91
Satimage (Vegetation Stubble Vs Others) [32]	470	3965	10 : 90
Shuttle (High Vs Flow) [32]	6748	34108	16 : 84
Sick (Sick Vs Negative) [32]	231	3541	6 : 94

Table 4.1: Class representation and split ratios for each dataset used

4.2 Classifier Comparison Experiment

The first experiment performed was designed to establish the general effectiveness of Under Bagging, Roughly Balanced Bagging, SMOTE and just plain Unskewed Sampling. This initial experiment varied parameters on the individual classifiers to get an idea of parameter sensitivity for the algorithms. All classifiers used J48 as their base classifier. The classifiers and parameters used are outlined in Table 4.2.

The experiments were performed using the WEKA experimenter, a 10x10-fold cross validation was performed using the above classifiers and the selected datasets. Tables 4.3 to 4.8 show the mean ROC AUC values along with the the standard deviation in square brackets.

It can be seen from these ROC area results that Unskewed Sampling and SMOTE almost never perform better than standard Bagging; the single exception being on the ICDM08 location Y data. Conversely, the phoneme and adult datasets are the only datasets for which Under Bagging and Roughly Balanced Bagging are beaten by Bagging, with Roughly Balanced Bagging using a 3:1 ratio performing particularly well overall.

In order to fully appreciate the advantages of Under Bagging and Roughly Balanced Bagging it is important to compare the training time for these classifiers to that of standard Bagging. The training times are given in Tables 4.9 to 4.14

It can be observed from the training times that both the 3:1 ratio Under Bagging and 3:1 Roughly Balanced Bagging, while running acceptably on the smaller datasets, run into time issues on datasets with a large number of minority examples; shuttle, magic, adult and to a lesser degree phoneme all take much longer to train than their 1:1 ratio counterparts.

Classifier [Short name]	Parameters
Bagging [Bagging]	100 iterations of standard Bagging.
Unskewed Sampling [USS 1x]	Under-sampling the majority class to 50% of its original size. Minority class example set size unchanged.
Unskewed Sampling [USS 3x]	Under-sampling the majority class to 50% of its original size. Minority class example set size oversampled to 300% of its original size.
SMOTE [SMT 1]	One synthetic minority class example generated for each existing minority example.
SMOTE [SMT 1se]	One synthetic minority class example generated for each existing minority example. Uses Synthetic Example Protection (SEP) and the Neighbor per Attribute Approach (NAA).
SMOTE [SMT 3]	Three synthetic minority class examples generated for each existing minority example.
SMOTE [SMT 3se]	Three synthetic minority class examples generated for each existing minority example. Uses SEP and NAA.
Roughly Balanced Bagging [RBB 1:1]	One to one ratio of majority to minority examples. 100 iterations.
Roughly Balanced Bagging [RBB 3:1]	Three to one ratio of majority to minority examples. 100 iterations.
Under Bagging [UB 1:1]	One to one ratio of majority to minority examples. 100 iterations.
Under Bagging [UB 3:1]	Three to one ratio of majority to minority examples. 100 iterations.

Table 4.2: The classifiers and their parameters for the Classifier Comparison Experiment (4.2)

Classifiers	Abalone(Oldest)	Abalone(Youngest)	Adult	Anneal
Bagging	0.8544[0.08]	0.9846[0.02]	0.9073[0.01]	**1.0000[0.00]**
USS 1x	0.5984[0.18]	0.8386[0.13]	0.8537[0.01]	0.9967[0.02]
USS 3x	0.5442[0.18]	0.8755[0.10]	0.8370[0.01]	**1.0000[0.00]**
SMT 1se	0.6209[0.18]	0.8836[0.10]	0.8720[0.01]	**1.0000[0.00]**
SMT 1se	0.6590[0.16]	0.8934[0.10]	0.8751[0.01]	**1.0000[0.00]**
SMT 3	0.6372[0.16]	0.9071[0.09]	0.8605[0.01]	**1.0000[0.00]**
SMT 3se	0.6518[0.17]	0.9161[0.09]	0.8635[0.01]	**1.0000[0.00]**
RBB 1:1	0.8827[0.07]	**0.9880[0.01]**	**0.9126[0.01]**	**1.0000[0.00]**
RBB 3:1	0.8822[0.07]	0.9848[0.02]	0.9090[0.01]	**1.0000[0.00]**
UB 1:1	0.8822[0.07]	0.9874[0.02]	**0.9126[0.01]**	**1.0000[0.00]**
UB 3:1	**0.8842[0.07]**	0.9849[0.02]	0.9090[0.01]	**1.0000[0.00]**

Table 4.3: ROC area results for the Classifier Comparison Experiment (4.2) - Abalone to Anneal Datasets

Classifiers	Car	Glass	Haberman	Hepatitis	Hypothyroid
Bagging	0.9973[0.00]	0.9556[0.08]	0.6926[0.10]	0.8410[0.13]	**0.9997[0.00]**
USS 1x	0.9650[0.05]	0.9016[0.12]	0.6162[0.12]	0.7462[0.16]	0.9755[0.04]
USS 3x	0.9753[0.03]	0.9152[0.09]	0.6217[0.11]	0.7393[0.15]	0.9879[0.03]
SMT 1se	0.9989[0.00]	0.9176[0.12]	0.6449[0.10]	0.6989[0.18]	0.9721[0.05]
SMT 1se	0.9988[0.00]	0.9133[0.11]	0.6465[0.09]	0.6630[0.18]	0.9787[0.05]
SMT 3	0.9986[0.00]	0.9023[0.13]	0.6493[0.11]	0.6887[0.19]	0.9772[0.05]
SMT 3se	0.9985[0.00]	0.9104[0.12]	0.6452[0.11]	0.7024[0.19]	0.9796[0.05]
RBB 1:1	0.9996[0.00]	0.9584[0.08]	0.7097[0.10]	0.8591[0.12]	0.9993[0.00]
RBB 3:1	0.9995[0.00]	0.9538[0.09]	0.6927[0.10]	0.8525[0.12]	0.9995[0.00]
UB 1:1	**0.9997[0.00]**	**0.9599[0.08]**	**0.7112[0.10]**	**0.8612[0.12]**	0.9993[0.00]
UB 3:1	**0.9997[0.00]**	0.9517[0.09]	0.6937[0.10]	0.8540[0.12]	0.9994[0.00]

Table 4.4: ROC area results for the Classifier Comparison Experiment (4.2) - Car to Hypothyroid Datasets

Classifiers	ICDM08-Full	ICDM08-V	ICDM08-W	ICDM08-X	ICDM08-Y
Bagging	0.7165[0.03]	0.7856[0.05]	0.7616[0.16]	0.7061[0.04]	0.5872[0.13]
USS 1x	0.5053[0.02]	0.5375[0.09]	0.5031[0.03]	0.5463[0.07]	0.4994[0.01]
USS 3x	0.6613[0.05]	0.7151[0.08]	0.6071[0.16]	0.6543[0.05]	0.5603[0.11]
SMT 1se	0.5439[0.05]	0.5610[0.11]	0.5000[0.00]	0.5503[0.06]	0.5005[0.01]
SMT 1se	0.5518[0.05]	0.5551[0.10]	0.5000[0.00]	0.5495[0.06]	0.5015[0.02]
SMT 3	0.6464[0.04]	0.6731[0.11]	0.5253[0.10]	0.6419[0.06]	0.5063[0.04]
SMT 3se	0.6422[0.04]	0.6748[0.12]	0.5193[0.08]	0.6358[0.06]	0.5113[0.05]
RBB 1:1	**0.7222[0.03]**	**0.8040[0.05]**	**0.8344[0.13]**	**0.7074[0.04]**	**0.7749[0.07]**
RBB 3:1	0.7201[0.03]	0.8011[0.05]	0.8187[0.14]	0.7033[0.04]	0.7728[0.08]
UB 1:1	0.7219[0.03]	0.8038[0.05]	0.8339[0.10]	0.7070[0.04]	0.7700[0.07]
UB 3:1	0.7198[0.02]	0.8007[0.05]	0.8190[0.14]	0.7045[0.04]	0.7748[0.08]

Table 4.5: ROC area results for the Classifier Comparison Experiment (4.2) - ICDM08-Full to ICDM08-Y

Classifiers	ICDM08-Z	Ionosphere	Magic	Phoneme
Bagging	**0.5644[0.05]**	0.9764[0.02]	0.9331[0.01]	0.9563[0.01]
USS 1x	0.5017[0.02]	0.8611[0.08]	0.8088[0.01]	0.8475[0.02]
USS 3x	0.5143[0.06]	0.8649[0.06]	0.7899[0.01]	0.8509[0.02]
SMT 1se	0.5116[0.04]	0.8766[0.08]	0.8559[0.01]	0.8851[0.02]
SMT 1se	0.5038[0.03]	0.8754[0.07]	0.8602[0.01]	0.8892[0.02]
SMT 3	0.5268[0.05]	0.8777[0.07]	0.8419[0.01]	0.8855[0.02]
SMT 3se	0.5271[0.05]	0.8682[0.07]	0.8504[0.01]	0.8891[0.02]
RBB 1:1	0.5539[0.06]	0.9736[0.03]	0.9338[0.01]	0.9510[0.01]
RBB 3:1	0.5474[0.06]	**0.9784[0.02]**	**0.9348[0.01]**	**0.9595[0.01]**
UB 1:1	0.5567[0.06]	0.9749[0.02]	0.9337[0.01]	0.9510[0.01]
UB 3:1	0.5439[0.06]	0.9775[0.02]	**0.9348[0.01]**	0.9592[0.01]

Table 4.6: ROC area results for the Classifier Comparison Experiment (4.2) - ICDM08-Z to Phoneme

Classifiers	Pima Indians Diabetes	Satimage(Cotton Crop)	Satimage(Damp Grey)
Bagging	0.8261[0.05]	0.9985[0.00]	0.9509[0.01]
USS 1x	0.6920[0.06]	0.9622[0.02]	0.7645[0.06]
USS 3x	0.6869[0.06]	0.9746[0.02]	0.7726[0.06]
SMT 1se	0.7494[0.05]	0.9720[0.02]	0.7527[0.07]
SMT 1se	0.7430[0.06]	0.9705[0.02]	0.7565[0.06]
SMT 3	0.7377[0.06]	0.9687[0.02]	0.7533[0.06]
SMT 3se	0.7436[0.06]	0.9749[0.02]	0.7602[0.06]
RBB 1:1	0.8316[0.05]	**0.9989[0.00]**	0.9511[0.01]
RBB 3:1	0.8225[0.05]	0.9988[0.00]	0.9514[0.01]
UB 1:1	**0.8318[0.05]**	0.9986[0.00]	0.9513[0.01]
UB 3:1	0.8237[0.05]	0.9985[0.00]	**0.9517[0.01]**

Table 4.7: ROC area results for the Classifier Comparison Experiment (4.2) - Pima Indians Diabetes to Satimage(Damp Grey) Datasets

Classifiers	Satimage(Vegetation Stubble)	Shuttle	Sick
Bagging	**0.9930[0.00]**	0.9999[0.00]	0.9954[0.01]
USS 1x	0.8889[0.04]	0.9995[0.00]	0.9399[0.05]
USS 3x	0.9106[0.03]	0.9997[0.00]	0.9637[0.03]
SMT 1se	0.9046[0.03]	0.9996[0.00]	0.9708[0.04]
SMT 1se	0.9081[0.03]	0.9996[0.00]	0.9621[0.04]
SMT 3	0.9078[0.03]	0.9996[0.00]	0.9681[0.03]
SMT 3se	0.9106[0.03]	0.9997[0.00]	0.9646[0.04]
RBB 1:1	0.9917[0.00]	**1.0000[0.00]**	0.9957[0.00]
RBB 3:1	0.9927[0.00]	**1.0000[0.00]**	0.9948[0.01]
UB 1:1	0.9916[0.00]	**1.0000[0.00]**	**0.9958[0.00]**
UB 3:1	0.9927[0.00]	**1.0000[0.00]**	0.9954[0.01]

Table 4.8: ROC area results for the Classifier Comparison Experiment (4.2) - Satimage(Vegetation Stubble) to Sick Datasets

Classifiers	Abalone(Oldest)	Abalone(Youngest)	Adult	Anneal
Bagging	2.31[0.07]	1.69[0.07]	229.56[3.76]	0.46[0.02]
USS 1x	**0.01[0.00]**	**0.01[0.00]**	**1.06[0.04]**	**0.00[0.00]**
USS 3x	0.02[0.00]	**0.01[0.00]**	2.42[0.10]	**0.00[0.00]**
SMT 1se	0.04[0.01]	0.03[0.00]	68.53[2.73]	0.02[0.00]
SMT 1se	0.05[0.01]	0.05[0.00]	255.66[3.43]	0.05[0.00]
SMT 3	0.04[0.00]	0.03[0.00]	168.03[4.29]	0.03[0.00]
SMT 3se	0.09[0.01]	0.10[0.00]	729.07[9.57]	0.12[0.00]
RBB 1:1	0.05[0.00]	0.04[0.00]	49.97[0.69]	0.02[0.01]
RBB 3:1	0.10[0.01]	0.11[0.00]	199.65[1.83]	0.05[0.03]
UB 1:1	0.05[0.00]	0.05[0.00]	50.45[0.66]	0.02[0.00]
UB 3:1	0.10[0.00]	0.10[0.01]	197.39[1.76]	0.04[0.01]

Table 4.9: Training times(seconds) for the Classifier Comparison Experiment (4.2) - Abalone to Anneal Datasets

Classifiers	Car	Glass	Haberman	Hepatitis	Hypothyroid
Bagging	0.09[0.01]	0.10[0.01]	0.08[0.01]	0.13[0.01]	2.13[0.14]
USS 1x	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.01[0.00]**
USS 3x	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.01[0.00]**
SMT 1se	0.02[0.00]	**0.00[0.00]**	**0.00[0.00]**	0.01[0.00]	0.15[0.02]
SMT 1se	0.04[0.00]	**0.00[0.00]**	**0.00[0.00]**	0.01[0.00]	0.37[0.02]
SMT 3	0.02[0.00]	**0.00[0.00]**	**0.00[0.00]**	0.01[0.00]	0.18[0.02]
SMT 3se	0.09[0.00]	0.01[0.00]	0.01[0.00]	0.02[0.00]	0.83[0.03]
RBB 1:1	0.01[0.00]	0.03[0.00]	0.04[0.00]	0.05[0.00]	0.10[0.00]
RBB 3:1	0.02[0.00]	0.06[0.00]	0.09[0.00]	0.10[0.00]	0.20[0.01]
UB 1:1	0.01[0.00]	0.03[0.00]	0.04[0.00]	0.05[0.00]	0.10[0.00]
UB 3:1	0.02[0.00]	0.06[0.01]	0.09[0.01]	0.10[0.01]	0.19[0.01]

Table 4.10: Training times(seconds) for the Classifier Comparison Experiment (4.2) - Car to Hypothyroid Datasets

Classifiers	ICDM08-Full	ICDM08-V	ICDM08-W	ICDM08-X	ICDM08-Y
Bagging	3.22[0.07]	0.38[0.01]	0.09[0.00]	0.67[0.02]	0.19[0.01]
USS 1x	**0.02[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**	**0.00[0.00]**
USS 3x	0.04[0.01]	**0.00[0.00]**	**0.00[0.00]**	0.01[0.00]	**0.00[0.00]**
SMT 1se	0.19[0.02]	0.01[0.00]	0.00[0.00]	0.01[0.00]	0.00[0.00]
SMT 1se	0.90[0.02]	0.02[0.00]	0.00[0.00]	0.05[0.00]	0.01[0.00]
SMT 3	0.36[0.02]	0.01[0.00]	0.00[0.00]	0.03[0.00]	0.00[0.00]
SMT 3se	2.46[0.05]	0.05[0.00]	0.00[0.00]	0.13[0.00]	0.01[0.00]
RBB 1:1	0.48[0.01]	0.05[0.00]	0.01[0.00]	0.12[0.00]	0.02[0.00]
RBB 3:1	1.00[0.03]	0.11[0.00]	0.01[0.00]	0.29[0.01]	0.03[0.00]
UB 1:1	0.45[0.01]	0.05[0.00]	0.01[0.00]	0.12[0.00]	0.02[0.00]
UB 3:1	0.98[0.03]	0.10[0.00]	0.01[0.00]	0.26[0.01]	0.03[0.00]

Table 4.11: Training times(seconds) for the Classifier Comparison Experiment (4.2) - ICDM08-Full to ICDM08-Y Datasets

Classifiers	ICDM08-Z	Ionosphere	Magic	Phoneme
Bagging	0.59[0.05]	1.55[0.03]	96.69[3.97]	8.19[0.07]
USS 1x	**0.00[0.00]**	**0.01[0.00]**	**0.68[0.03]**	**0.06[0.00]**
USS 3x	0.01[0.00]	**0.01[0.00]**	1.38[0.05]	0.12[0.01]
SMT 1se	0.02[0.00]	0.03[0.00]	8.01[0.32]	0.38[0.01]
SMT 1se	0.06[0.00]	0.05[0.00]	34.99[3.48]	1.13[0.01]
SMT 3	0.04[0.01]	0.06[0.00]	22.03[0.81]	0.97[0.02]
SMT 3se	0.18[0.01]	0.10[0.00]	96.96[12.46]	3.20[0.03]
RBB 1:1	0.11[0.01]	0.91[0.03]	59.98[0.47]	4.33[0.03]
RBB 3:1	0.27[0.02]	1.95[0.05]	154.54[1.96]	9.45[0.05]
UB 1:1	0.10[0.01]	0.90[0.03]	59.12[0.60]	4.29[0.03]
UB 3:1	0.24[0.02]	1.87[0.05]	144.65[2.62]	9.23[0.05]

Table 4.12: Training times(seconds) for the Classifier Comparison Experiment (4.2) - ICDM08-Z to Phoneme Datasets

Classifiers	Pima Indians Diabetes	Satimage(Cotton Crop)	Satimage(Damp Grey)
Bagging	0.84[0.01]	19.91[0.44]	16.00[0.18]
USS 1x	**0.01[0.00]**	**0.09[0.01]**	**0.10[0.01]**
USS 3x	**0.01[0.00]**	0.12[0.01]	0.18[0.01]
SMT 1se	0.02[0.00]	0.38[0.03]	0.37[0.01]
SMT 1se	0.05[0.00]	0.89[0.03]	0.84[0.02]
SMT 3	0.06[0.00]	0.60[0.03]	0.72[0.03]
SMT 3se	0.12[0.00]	2.14[0.06]	2.15[0.05]
RBB 1:1	0.61[0.01]	2.40[0.04]	3.68[0.05]
RBB 3:1	1.27[0.02]	6.65[0.15]	6.80[0.06]
UB 1:1	0.57[0.01]	2.23[0.04]	3.54[0.04]
UB 3:1	1.16[0.02]	6.37[0.14]	6.56[0.07]

Table 4.13: Training times(seconds) for the Classifier Comparison Experiment (4.2) - Pima Indians Diabetes to Satimage(Damp Grey) Datasets

Classifiers	Satimage(Vegetation Stubble)	Shuttle	Sick
Bagging	18.24[0.30]	74.59[3.25]	3.64[0.32]
USS 1x	**0.10[0.01]**	**0.28[0.04]**	**0.02[0.01]**
USS 3x	0.14[0.01]	0.41[0.04]	0.03[0.00]
SMT 1se	0.37[0.02]	5.35[0.35]	0.25[0.04]
SMT 1se	0.92[0.02]	55.73[7.42]	0.75[0.04]
SMT 3	0.67[0.02]	12.87[0.74]	0.36[0.04]
SMT 3se	2.29[0.05]	144.52[17.50]	1.86[0.05]
RBB 1:1	3.32[0.03]	7.89[0.19]	0.43[0.01]
RBB 3:1	6.95[0.08]	31.81[3.26]	1.03[0.05]
UB 1:1	3.12[0.03]	7.68[0.14]	0.41[0.01]
UB 3:1	6.80[0.09]	30.92[1.03]	0.89[0.04]

Table 4.14: Training times(seconds) for the Classifier Comparison Experiment (4.2) - Satimage(Vegetation Stubble) to Sick Datasets

In the worst case, with the magic dataset, the 3:1 ratio Under Bagging and Roughly Balanced Bagging classifiers actually take longer to train than standard Bagging! This is easily explained however, as it is due to the magic dataset having one of the least skewed splits (35% minority class); this means the dataset started with a 2:1 split and the Under Bagging step is actually sampling more majority examples per minority example than the original dataset contained. Thus Under Bagging and Roughly Balanced Bagging have effectively become forms of over-sampling.

This indicates that it is important to consider lowering the ratio for datasets where the number of minority examples is large as high ratios will slow the skewed data classifiers significantly.

The most interesting conclusion, once reviewing the training times in addition to the AUC results, is that Under Bagging and Roughly Balanced Bagging are both capable of getting good AUC results when compared standard Bagging as well as keeping their training times lower and in most cases significantly lower than the Bagging baseline. It is also interesting that there does not appear to be any parameter combination for the skewed data classifiers that performs the best over all the datasets; the Parameter Tuning Experiment (4.3) explores this further.

SMOTE appears, in these experiments, to have quite a bad results versus training time trade-off. Although it is not iteratively applying its base classifier, it is still has quite high training times, sometimes similar to the 1:1 Under Bagging or Roughly Balance Bagging results which are building 100 models. Also, using the novel Synthetic Example Protection and the Neighbor per Attribute approaches seems to make the situation even worse; the adult dataset in particular takes over five times as long to train with these options enabled. As these additional settings appear to improve results on some datasets it is possible they could be quite effective on some niche problems where the longer

training times are less of a hindrance.

4.3 Parameter Tuning Experiment

Having identified the potential of Under Bagging and confirming that Roughly Balanced Bagging was performing well, the next step was to find out how sensitive the results were to varying the classifier. For both Under Bagging and Roughly Balanced Bagging there is one parameter to select. This parameter will decide on the class ratio for Under Bagging and the rough class ratio for Roughly Balanced Bagging.

Classifier [Short name]	Parameter	Ratio (majority:minority)
Bagging [Bagging]	None	Default (baseline)
Under Bagging [UB 1:3]	0.33	1:3
Roughly Balanced Bagging [RBB 1:3]	0.75	1:3
Under Bagging [UB 1:2]	0.5	1:2
Roughly Balanced Bagging [RBB 1:2]	0.66	1:2
Under Bagging [UB 1:1]	1.0	1:1
Roughly Balanced Bagging [RBB 1:1]	0.5	1:1
Under Bagging [UB 2:1]	2.0	2:1
Roughly Balanced Bagging [RBB 2:1]	0.33	2:1
Under Bagging [UB 3:1]	3.0	3:1
Roughly Balanced Bagging [RBB 3:1]	0.25	3:1

Table 4.15: The classifiers, their parameters and resulting ratios for the Parameter Tuning Experiment (4.3)

Some of the parameters chosen would create ratios where there were more majority class examples to each minority example. For both classifiers there were parameters set to produce a 1 to 1 ratio. Finally there were a few classifiers running parameters that would leave more minority class examples than majority. Table 4.15 lists the parameters and classifiers used.

Classifiers	Abalone(Oldest)	Abalone(Youngest)	Adult	Anneal
Bagging	0.8544[0.08]	0.9846[0.02]	0.9073[0.01]	**1.0000[0.00]**
RBB 3:1	0.8822[0.07]	0.9848[0.02]	0.9090[0.01]	**1.0000[0.00]**
RBB 2:1	0.8830[0.07]	0.9845[0.02]	0.9108[0.01]	**1.0000[0.00]**
RBB 1:1	0.8827[0.07]	0.9880[0.01]	0.9126[0.01]	**1.0000[0.00]**
RBB 1:2	0.8757[0.07]	**0.9896[0.01]**	0.9131[0.01]	**1.0000[0.00]**
RBB 1:3	0.8673[0.07]	0.9881[0.01]	**0.9137[0.01]**	**1.0000[0.00]**
UB 3:1	0.8842[0.07]	0.9849[0.02]	0.9090[0.01]	**1.0000[0.00]**
UB 2:1	**0.8853[0.07]**	0.9848[0.02]	0.9109[0.01]	**1.0000[0.00]**
UB 1:1	0.8822[0.07]	0.9874[0.02]	0.9126[0.01]	**1.0000[0.00]**
UB 1:2	0.8751[0.07]	**0.9896[0.01]**	0.9131[0.01]	**1.0000[0.00]**
UB 1:3	0.8640[0.07]	0.9888[0.01]	0.9136[0.01]	**1.0000[0.00]**

Table 4.16: ROC area results for the Parameter Tuning Experiment (4.3) - Abalone to Anneal Datasets

Tables 4.16 to 4.21 show the AUC results of the 10x10-fold cross validation experiment run on these classifiers. All classifiers in this experiment were set to 100 iterations of bagging and used unpruned J48 trees as the base classifier.

The first observation that can be made from this data is that Under Bagging and Roughly Balanced Bagging continue to be very viable alternatives to standard Bagging. An Under Bagging or Roughly Balanced Bagging classifier produces the top ROC area on all but four of the datasets; in these cases Bagging beats the other classifiers by a very slim margin. The performance of the classifiers seems to be very consistent over all of the parameter combinations tried.

If a count of significant wins, losses and ties is taken, it turns out, based on AUC alone, that Under Bagging 1:2, Under Bagging 1:3 and Roughly Balanced Bagging 1:2 get very similar results overall and are in a tier above the other classifiers. Somewhat close, but a little further behind, are the other classifiers with ratios of 1:1 or greater (in favour of the majority class). The classifiers showing up last include the less than 1:1 classifiers and standard Bagging.

Classifiers	Car	Glass	Haberman	Hepatitis	Hypothyroid
Bagging	0.9973[0.00]	0.9556[0.08]	0.6926[0.10]	0.8410[0.13]	**0.9997[0.00]**
RBB 3:1	0.9995[0.00]	0.9538[0.09]	0.6927[0.10]	0.8525[0.12]	0.9995[0.00]
RBB 2:1	**0.9997[0.00]**	**0.9654[0.07]**	0.7011[0.10]	0.8560[0.12]	0.9994[0.00]
RBB 1:1	0.9996[0.00]	0.9584[0.08]	0.7097[0.10]	0.8591[0.12]	0.9993[0.00]
RBB 1:2	0.9995[0.00]	0.9628[0.07]	**0.7210[0.10]**	0.8600[0.12]	0.9992[0.00]
RBB 1:3	0.9994[0.00]	0.9629[0.07]	0.7180[0.09]	**0.8628[0.12]**	0.9988[0.00]
UB 3:1	**0.9997[0.00]**	0.9517[0.09]	0.6937[0.10]	0.8540[0.12]	0.9994[0.00]
UB 2:1	0.9996[0.00]	0.9648[0.07]	0.7046[0.10]	0.8603[0.12]	0.9994[0.00]
UB 1:1	**0.9997[0.00]**	0.9599[0.08]	0.7112[0.10]	0.8612[0.12]	0.9993[0.00]
UB 1:2	0.9995[0.00]	0.9626[0.07]	0.7176[0.10]	0.8569[0.12]	0.9990[0.00]
UB 1:3	0.9993[0.00]	0.9635[0.07]	0.7201[0.09]	**0.8628[0.12]**	0.9988[0.00]

Table 4.17: ROC area results for the Parameter Tuning Experiment (4.3) - Car to Hypothyroid Datasets

Classifiers	ICDM08-Full	ICDM08-V	ICDM08-W	ICDM08-X	ICDM08-Y
Bagging	0.7165[0.03]	0.7856[0.05]	0.7616[0.16]	0.7061[0.04]	0.5872[0.13]
RBB 3:1	0.7201[0.03]	0.8011[0.05]	0.8187[0.14]	0.7033[0.04]	0.7728[0.08]
RBB 2:1	0.7212[0.03]	0.8016[0.05]	0.8233[0.14]	0.7034[0.04]	0.7761[0.08]
RBB 1:1	0.7222[0.03]	0.8040[0.05]	0.8344[0.13]	0.7074[0.04]	0.7749[0.07]
RBB 1:2	0.7216[0.03]	0.8056[0.05]	0.8519[0.10]	0.7097[0.04]	0.7705[0.07]
RBB 1:3	0.7226[0.03]	**0.8077[0.05]**	**0.8559[0.11]**	**0.7111[0.04]**	**0.7779[0.07]**
UB 3:1	0.7198[0.02]	0.8007[0.05]	0.8190[0.14]	0.7045[0.04]	0.7748[0.08]
UB 2:1	0.7215[0.02]	0.8027[0.05]	0.8253[0.14]	0.7040[0.04]	0.7777[0.07]
UB 1:1	0.7219[0.03]	0.8038[0.05]	0.8339[0.10]	0.7070[0.04]	0.7700[0.07]
UB 1:2	0.7228[0.02]	0.8074[0.05]	0.8543[0.10]	0.7106[0.04]	0.7760[0.07]
UB 1:3	**0.7232[0.03]**	0.8049[0.05]	0.8528[0.11]	0.7089[0.04]	0.7704[0.07]

Table 4.18: ROC area results for the Parameter Tuning Experiment (4.3) - ICDM08-Full to ICDM08-Y Datasets

Classifiers	ICDM08-Z	Ionosphere	Magic	Phoneme
Bagging	**0.5644[0.05]**	0.9764[0.02]	0.9331[0.01]	0.9563[0.01]
RBB 3:1	0.5474[0.06]	**0.9784[0.02]**	**0.9348[0.01]**	**0.9595[0.01]**
RBB 2:1	0.5512[0.06]	0.9773[0.02]	**0.9348[0.01]**	0.9568[0.01]
RBB 1:1	0.5539[0.06]	0.9736[0.03]	0.9338[0.01]	0.9510[0.01]
RBB 1:2	0.5571[0.06]	0.9687[0.03]	0.9320[0.01]	0.9442[0.01]
RBB 1:3	0.5579[0.06]	0.9636[0.03]	0.9299[0.01]	0.9387[0.01]
UB 3:1	0.5439[0.06]	0.9775[0.02]	**0.9348[0.01]**	0.9592[0.01]
UB 2:1	0.5510[0.06]	0.9762[0.02]	0.9347[0.01]	0.9564[0.01]
UB 1:1	0.5567[0.06]	0.9749[0.02]	0.9337[0.01]	0.9510[0.01]
UB 1:2	0.5569[0.06]	0.9683[0.03]	0.9318[0.01]	0.9435[0.01]
UB 1:3	0.5530[0.06]	0.9622[0.03]	0.9298[0.01]	0.9386[0.01]

Table 4.19: ROC area results for the Parameter Tuning Experiment (4.3) - ICDM08-Z to Phoneme Datasets

Classifiers	Pima Indians Diabetes	Satimage(Cotton Crop)	Satimage(Damp Grey)
Bagging	0.8261[0.05]	0.9985[0.00]	0.9509[0.01]
RBB 3:1	0.8225[0.05]	0.9988[0.00]	0.9514[0.01]
RBB 2:1	0.8259[0.05]	0.9985[0.00]	0.9517[0.01]
RBB 1:1	0.8316[0.05]	**0.9989[0.00]**	0.9511[0.01]
RBB 1:2	**0.8334[0.05]**	0.9986[0.00]	0.9494[0.02]
RBB 1:3	0.8289[0.05]	0.9985[0.00]	0.9481[0.02]
UB 3:1	0.8237[0.05]	0.9985[0.00]	0.9517[0.01]
UB 2:1	0.8276[0.05]	0.9983[0.00]	**0.9518[0.01]**
UB 1:1	0.8318[0.05]	0.9986[0.00]	0.9513[0.01]
UB 1:2	0.8324[0.05]	0.9985[0.00]	0.9487[0.02]
UB 1:3	0.8296[0.05]	0.9984[0.00]	0.9474[0.02]

Table 4.20: ROC area results for the Parameter Tuning Experiment (4.3) - Pima Indians Diabetes to Satimage(Damp Grey) Datasets

Classifiers	Satimage(Vegetation Stubble)	Shuttle	Sick
Bagging	**0.9930[0.00]**	0.9999[0.00]	0.9954[0.01]
RBB 3:1	0.9927[0.00]	**1.0000[0.00]**	0.9948[0.01]
RBB 2:1	0.9923[0.00]	**1.0000[0.00]**	**0.9960[0.01]**
RBB 1:1	0.9917[0.00]	**1.0000[0.00]**	0.9957[0.00]
RBB 1:2	0.9918[0.00]	**1.0000[0.00]**	0.9951[0.00]
RBB 1:3	0.9914[0.00]	**1.0000[0.00]**	0.9939[0.00]
UB 3:1	0.9927[0.00]	**1.0000[0.00]**	0.9954[0.01]
UB 2:1	0.9921[0.00]	**1.0000[0.00]**	0.9951[0.01]
UB 1:1	0.9916[0.00]	**1.0000[0.00]**	0.9958[0.00]
UB 1:2	0.9918[0.00]	**1.0000[0.00]**	0.9949[0.00]
UB 1:3	0.9910[0.00]	**1.0000[0.00]**	0.9940[0.00]

Table 4.21: ROC area results for the Parameter Tuning Experiment (4.3) - Satimage(Vegetation Stubble) to Sick Datasets

However, just as in the Classifier Comparison Experiment (4.2), it is important to take into account the time to train the models. Tables 4.22 to 4.27 show these training times.

Perhaps it did not stand out in the Classifier Comparison Experiment (4.2), but it is now quite clear that adding more majority class examples really does have a notable impact on training times. The adult and magic datasets show this impact most obviously. However, generally speaking, Under Bagging and Roughly Balanced Bagging classifiers keep faster training times than standard Bagging.

It seems as if there is a trade-off between training time and potential AUC. However, while it is true that training with a set of about 2 or 3 majority class examples per minority example seems to give the best AUC, trading off a small amount of AUC by selecting a 1 to 1 example ratio yields a significantly faster training time. Thus, if maximising AUC was not of paramount importance, an option that produces a model in less time could be adopted and still be competing well with standard Bagging.

Classifiers	Abalone(Oldest)	Abalone(Youngest)	Adult	Anneal
Bagging	2.31[0.07]	1.69[0.07]	229.56[3.76]	0.46[0.02]
RBB 3:1	0.11[0.02]	0.10[0.00]	197.00[5.69]	0.05[0.03]
RBB 2:1	0.08[0.00]	0.08[0.00]	123.42[0.89]	0.03[0.00]
RBB 1:1	0.05[0.00]	0.04[0.00]	52.86[0.64]	**0.02[0.00]**
RBB 1:2	**0.03[0.00]**	0.03[0.00]	30.64[0.27]	**0.02[0.02]**
RBB 1:3	**0.03[0.00]**	**0.02[0.00]**	23.98[0.19]	**0.02[0.00]**
UB 3:1	0.10[0.02]	0.10[0.00]	181.77[3.65]	0.04[0.01]
UB 2:1	0.07[0.00]	0.07[0.00]	104.41[2.69]	0.03[0.01]
UB 1:1	0.05[0.00]	0.05[0.00]	48.22[0.56]	**0.02[0.00]**
UB 1:2	**0.03[0.00]**	0.03[0.00]	28.50[0.23]	**0.02[0.00]**
UB 1:3	**0.03[0.00]**	0.03[0.00]	**22.88[0.17]**	**0.02[0.00]**

Table 4.22: Training times(seconds) for the Parameter Tuning Experiment (4.3) - Abalone to Anneal Datasets

Classifiers	Car	Glass	Haberman	Hepatitis	Hypothyroid
Bagging	0.09[0.01]	0.10[0.01]	0.08[0.01]	0.13[0.01]	2.13[0.14]
RBB 3:1	0.02[0.00]	0.06[0.00]	0.09[0.00]	0.10[0.00]	0.19[0.00]
RBB 2:1	0.02[0.00]	0.04[0.00]	0.07[0.00]	0.08[0.00]	0.14[0.00]
RBB 1:1	**0.01[0.00]**	0.03[0.00]	0.04[0.00]	0.05[0.00]	0.10[0.00]
RBB 1:2	**0.01[0.00]**	**0.02[0.00]**	0.03[0.00]	0.04[0.00]	0.07[0.00]
RBB 1:3	**0.01[0.00]**	**0.02[0.00]**	**0.02[0.00]**	**0.03[0.00]**	**0.06[0.00]**
UB 3:1	0.02[0.00]	0.06[0.00]	0.09[0.00]	0.10[0.01]	0.19[0.01]
UB 2:1	0.02[0.00]	0.04[0.00]	0.07[0.00]	0.08[0.00]	0.15[0.00]
UB 1:1	**0.01[0.00]**	0.03[0.00]	0.04[0.00]	0.05[0.00]	0.10[0.00]
UB 1:2	**0.01[0.00]**	**0.02[0.00]**	0.03[0.00]	0.04[0.00]	0.07[0.00]
UB 1:3	**0.01[0.00]**	**0.02[0.00]**	**0.02[0.00]**	**0.03[0.00]**	**0.06[0.00]**

Table 4.23: Training times(seconds) for the Parameter Tuning Experiment (4.3) - Car to Hypothyroid Datasets

Classifiers	ICDM08-Full	ICDM08-V	ICDM08-W	ICDM08-X	ICDM08-Y
Bagging	3.22[0.07]	0.38[0.01]	0.09[0.00]	0.67[0.02]	0.19[0.01]
RBB 3:1	1.00[0.02]	0.11[0.00]	0.01[0.00]	0.29[0.01]	0.03[0.00]
RBB 2:1	0.74[0.02]	0.08[0.00]	0.01[0.00]	0.21[0.01]	0.02[0.00]
RBB 1:1	0.48[0.01]	0.05[0.00]	0.01[0.00]	0.12[0.00]	0.02[0.00]
RBB 1:2	0.33[0.01]	0.04[0.00]	0.01[0.00]	0.09[0.00]	**0.01[0.00]**
RBB 1:3	0.27[0.01]	**0.03[0.00]**	**0.00[0.00]**	**0.07[0.00]**	**0.01[0.00]**
UB 3:1	0.97[0.03]	0.10[0.01]	0.01[0.00]	0.25[0.01]	0.03[0.00]
UB 2:1	0.71[0.02]	0.08[0.00]	0.01[0.00]	0.19[0.00]	0.02[0.00]
UB 1:1	0.45[0.01]	0.05[0.00]	0.01[0.00]	0.12[0.00]	0.02[0.00]
UB 1:2	0.30[0.01]	0.04[0.00]	0.01[0.00]	0.08[0.00]	**0.01[0.00]**
UB 1:3	**0.25[0.01]**	**0.03[0.00]**	**0.00[0.00]**	**0.07[0.00]**	**0.01[0.00]**

Table 4.24: Training times(seconds) for the Parameter Tuning Experiment (4.3) - ICDM08-Full to ICDM08-Y Datasets

Classifiers	ICDM08-Z	Ionosphere	Magic	Phoneme
Bagging	0.59[0.05]	1.55[0.03]	96.69[3.97]	8.19[0.07]
RBB 3:1	0.26[0.02]	1.93[0.05]	155.50[3.11]	9.46[0.06]
RBB 2:1	0.20[0.01]	1.50[0.04]	107.81[0.95]	7.08[0.06]
RBB 1:1	0.11[0.01]	0.89[0.03]	60.16[0.44]	4.33[0.03]
RBB 1:2	0.07[0.00]	0.52[0.02]	37.89[0.22]	2.97[0.02]
RBB 1:3	**0.05[0.00]**	**0.38[0.01]**	28.95[0.17]	2.38[0.03]
UB 3:1	0.24[0.02]	1.86[0.05]	141.50[4.39]	9.19[0.06]
UB 2:1	0.17[0.01]	1.48[0.03]	103.66[1.97]	6.90[0.05]
UB 1:1	0.10[0.01]	0.90[0.03]	59.00[0.56]	4.29[0.04]
UB 1:2	0.07[0.01]	0.52[0.02]	36.48[0.20]	2.93[0.03]
UB 1:3	**0.05[0.00]**	**0.38[0.01]**	**28.69[0.15]**	**2.36[0.02]**

Table 4.25: Training times(seconds) for the Parameter Tuning Experiment (4.3) - ICDM08-Z to Phoneme Datasets

Classifiers	Pima Indians Diabetes	Satimage(Cotton Crop)	Satimage(Damp Grey)
Bagging	0.84[0.01]	19.91[0.44]	16.00[0.18]
RBB 3:1	1.26[0.02]	6.58[0.14]	6.67[0.07]
RBB 2:1	1.02[0.01]	4.60[0.10]	5.42[0.06]
RBB 1:1	0.60[0.01]	2.38[0.04]	3.66[0.04]
RBB 1:2	0.41[0.01]	1.57[0.03]	2.81[0.04]
RBB 1:3	**0.30[0.01]**	1.26[0.02]	2.28[0.03]
UB 3:1	1.15[0.02]	6.19[0.13]	6.41[0.10]
UB 2:1	0.88[0.01]	4.09[0.08]	4.99[0.06]
UB 1:1	0.56[0.01]	2.20[0.04]	3.44[0.07]
UB 1:2	0.37[0.01]	1.43[0.03]	2.55[0.03]
UB 1:3	**0.30[0.01]**	**1.14[0.02]**	**2.11[0.03]**

Table 4.26: Training times(seconds) for the Parameter Tuning Experiment (4.3) - Pima Indians Diabetes to Satimage(Damp Grey) Datasets

Classifiers	Satimage(Vegetation Stubble)	Shuttle	Sick
Bagging	18.24[0.30]	74.59[3.25]	3.64[0.32]
RBB 3:1	6.89[0.08]	30.78[2.89]	1.01[0.05]
RBB 2:1	5.32[0.06]	18.13[1.12]	0.73[0.03]
RBB 1:1	3.28[0.04]	8.13[0.14]	0.42[0.01]
RBB 1:2	2.42[0.03]	4.71[0.06]	0.29[0.01]
RBB 1:3	2.08[0.04]	3.68[0.03]	**0.23[0.01]**
UB 3:1	6.62[0.10]	29.93[2.96]	0.87[0.04]
UB 2:1	4.82[0.05]	17.18[0.82]	0.65[0.02]
UB 1:1	3.06[0.03]	7.61[0.26]	0.40[0.01]
UB 1:2	2.24[0.04]	4.50[0.12]	0.27[0.01]
UB 1:3	**1.95[0.04]**	**3.57[0.05]**	**0.23[0.01]**

Table 4.27: Training times(seconds) for the Parameter Tuning Experiment (4.3) - Satimage(Vegetation Stubble) to Sick Datasets

After considering all the results available it seems that, in the absence of any additional information, it makes sense to approach a skewed problem with a specific classifier designed to deal with the skew. When selecting parameters it does not make sense to select a ratio which gets more minority class examples than majority. Starting with two majority class examples for every minority class example seems like a good balance between training time and AUC result.

Chapter 5

Pre-processing Raw Information

It is no exaggeration to say that pre-processing of information has been responsible for the majority of the time consumed in this investigation. This is due, most significantly, to the continuous improvements made to the code responsible for the various pre-processing stages and the necessity of running the modified results of any changes made to a stage through the entire remaining pre-processing stages again. To further complicate changes, the nature of pre-processing is that very little can be parallelised; this is because each stage in pre-processing is performed one after another for any single herd.

The various stages to the pre-processing of the given data can be summarised into five areas: converting the databases, extracting implied information, outlier detection and removal, attribute generation and generating ARFF files. Conversion of the Firebird database files into MySQL databases is performed first, for primarily logistical reasons. Next, processing was done on milking times in order to add an explicit label specifying if data belonged to a morning or afternoon milking period. Then, in order to remove some of the obviously present outlier data, an outlier removal process was performed in an attempt to clean up the errors in the data. Various attributes are then generated from the cleaned data after a database flattening query is executed. Finally the attributes generated from the flattened database are collected as learning instances and added to an ARFF file in preparation for feeding into a classifier.

5.1 Available Information

Database Name	Season	Herd Size	Has Daily		Heat Events	Mating Events
			Weights	Milk Yields		
A-06	2006	506 cows	yes	yes	468	535
B-06	2006	582 cows	yes	yes	240	344
B-07	2007	591 cows	yes	yes	536	539
C-07	2007	904 cows	no	yes	1	741
D-07	2007	1005 cows	yes	no	2	880

Table 5.1: Information about the data available for each herd

The data provided by LIC for the purposes of this investigation was in the form of Firebird databases. Data for five herds was provided; some with data from the 2006 milking season and some from the 2007 season with one providing both. Worth noting is that the C-07 herd does not contain daily weight recordings while the D-07 herd does not contain a set of daily milk yield values. Therefore these herds do not have the attributes generated from this information present in experiments run on them. A summary of the relevant details for each herd can be found in Table 5.1

5.2 Converting Firebird Databases

The first step in pre-processing the data was to change the provided Firebird databases to MySQL databases. The advantages of MySQL database files over Firebird files are twofold. The first and most significant advantage is that MySQL was supported on the computers available at the University of Waikato where the investigation was being completed. The second advantage was that MySQL is considerably better supported in the public domain, popular tools such as phpMyAdmin were available and Java and PHP access of MySQL was much more commonly used by the development community; therefore a solution to any problems encountered could be found in more reasonable time.

5.2.1 Export Firebird to SQL Statements

The first step in the conversion is to export an existing Firebird database table structure and its rows of data to SQL CREATE TABLE and INSERT statements through a piece of software capable of loading a Firebird database and with the functionality to export it to SQL statements. The software used to accomplish this goal in this investigation was EMS SQL Manager for Interbase/Firebird Databases[34].

5.2.2 Find and Replace Commands

The next step was to perform some limited alterations to the generated SQL file to make it suitable for the MySQL import process. This is not a foolproof process however as there are sometimes characters or single SQL statements with a strange syntax that causes the import process to stop even after performing the general modifications. The answer in this case is that human interaction is required to find a solution that lets the import continue without compromising the values of the data preventing the import and without modifying any other data as a side effect. There is no universal solution and whenever something stopped the import process it was assessed on a per case basis and altered to allow the import to continue.

The general alterations are simply a list of find and replace commands executed using the 'sed'[35] program, which is available on most UNIX installations. The find–>replace statements are shown below followed by an explanation of why the replace statement needs to occur.

```
REPLACE:
  /*
WITH:
  #

REPLACE:
```

```
    */
  WITH:
    #
```

The '/*' and '*/' comment syntax used in a Firebird SQL file needs to be replaced with the '#' style comment syntax of MySQL to allow import.

```
  REPLACE:
    CHARACTER SET NONE
  WITH:
    <blank>

  REPLACE:
    COLLATE NONE
  WITH:
    <blank>
```

The 'CHARACTER SET NONE' and 'COLLATE NONE' options on a column are not supported in MySQL and thus must be removed, as this does not compromise data values it is safe to simply remove (as is represented by <blank >).

```
  REPLACE:
    IBBOOL
  WITH:
    ENUM ('T', 'F')
```

MySQL does not have a Boolean data type and thus the IBBOOL type from Firebird needs to be replaced. This replace statement turns Boolean types into MySQL enumerated types with true and false values.

```
  REPLACE:
    BLOB SUB_TYPE 1
  WITH:
```

```
  TEXT
```

MySQL does not have a BLOB type; the replacement to a TEXT type allows values to be unaltered.

```
REPLACE:
  current_time
WITH:
  CURRENT_TIMESTAMP
```

The MySQL equivalent of current_time is the CURRENT_TIMESTAMP defined variable, a simple switch to this produces the correct result in MySQL.

```
REPLACE:
  "INIT"
WITH:
  INITIAL

REPLACE:
  INIT
WITH:
  INITIAL

REPLACE:
  "SQL"
WITH:
  SEQUEL

REPLACE:
  SQL
WITH:
  SEQUEL
```

In MySQL INIT and SQL are keywords, however the original Firebird databases used INIT and SQL as column names, renaming these columns to longer names allows import while maintaining column name meanings.

```
REPLACE:
  "NAME"
WITH:
  NAME
```

The opposite problem to the above, NAME is a keyword in Firebird databases and thus the export to SQL process escapes the column name using quote marks, however, in MySQL a column name with quote marks is not the same as the column name without the quotes, so in order to make all INSERT statements work the quote marks must be removed.

```
REPLACE:
  INSERT
WITH:
  REPLACE
```

Firebird databases appears to be case sensitive, where MySQL is not, unfortunately there was not enough time for a person to fix every case where this was an issue and the choice to simply overwrite conflicts was made, all INSERT statements become REPLACE statements which will overwrite a data row with the same values.

```
REPLACE:
  '\'
WITH:
  '\\'
```

Finally, in a few places a slash will be present in the exported SQL file that needs to be escaped to remain a literal slash when imported.

5.2.3 Importing SQL Statements to MySQL

The last step in the conversion process is to import the altered export files into an empty MySQL database with an appropriate name. The basic command line client provided with an installation of MySQL is capable of performing an import, in the form of SQL statements, without the need to use additional software. This stage is very simple assuming the provided files contain MySQL compliant statements. Failing that, there will be an error during the import and the database will not be completely recreated.

5.3 Extracting Implicit Information

Within the raw data provided there were a few pieces of information which were not explicitly specified but would be important to know. Cow milking typically occurs twice daily with effectively a morning and an afternoon milking every day. It was necessary to know what time separated the morning and afternoon milking periods, and related to this was which milk period each recorded milking event belonged to. Once all milking dates could be assigned to a milking period it would then be possible to calculate the order the cows were milked in.

In order to associate a row of milking data to its relevant milking period it was necessary to come up with a time threshold to separate the morning and afternoon milking periods. This choice was made by manually inspecting the milking times for a suitable dividing time based on the distances between milking times, the value chosen based on this analysis was midday. It was considered that a dynamic time might have to be chosen for each herd or even each day but midday appeared to be accurate for most examples. Thus milking times before midday were attributed to the morning milking period and those after midday were considered part of the afternoon milking period. This process was used to assign milking period information to cow weight and

milk yield data as well, this was an important step for the database flattening process.

Having a way to associate a milking time and date to a specific milking period meant it was possible to generate the order the cows were milked in during any single milking period. This step was performed for all milking periods in preparation for the creation of milking order attributes during the attribute generation process described later. Thus the explicit milking date and time information was transformed; making the milking order and milk period information explicit.

5.4 Outlier Removal

This outlier removal section will actually cover two related transformations made to the raw data. The first is concerned with the identification and removal of outliers, however, as the second transformation shows, the outliers are not actually removed from the data; they are simply marked for the second step in the overall process. This second step is actually an attempt to clean up the data before the attribute generation by filling in gaps where values are missing in the data, and also the marked outlier data identified in the first step, with suitable replacement values.

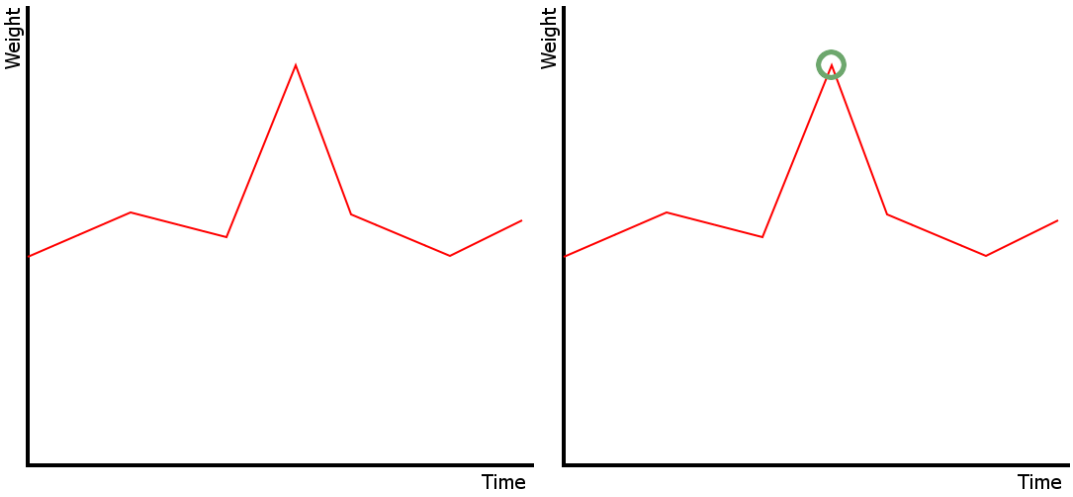
The step that is the real outlier detection phase is performed on both the weight and yield data provided by the unprocessed database files. The basic reasoning behind performing this outlier removal phase is that upon inspection by an informed human the available weight and yield information is obviously flawed. Reasoning may include an unrealistic change in cows weight between daily measurements or impossibly large amounts of milk production recorded; explanations for the errors may vary from human recording error to faulty calibration of measurement tools. Unfortunately, although an informed human could identify faults in the recorded data, it is not realistic for a human to sift

through tens of thousands of records finding each offending value.

The automated technique used in this investigation to identify outliers involved moving a sliding time window of 11 days over the available weight or yield data and finding each median over this window. The 6^{th} value in the window was the one being assessed relative to the five values before and after it in time. Each time a value was known for a milking period it was compared to the median of the 11 day window; if the value was found to be more than 10% different from the median it was marked as an outlier. 10% was picked as a threshold by having a human view data graphs with outliers marked on while the threshold was varied.

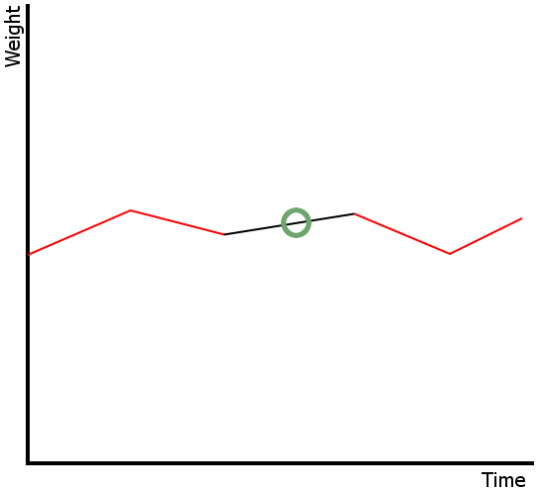
The second step in the overall outlier removal process was intended to make use of the fact that the data was time series data and had a logical progression over time; this knowledge meant data values between known values in time could be extrapolated and filled in. The process involved starting from the earliest known value in time, for both weight and yield information, and, one day at a time, progressed towards the last known value estimating missing values and replacing outlier values as it progressed.

Whenever a value for a day was known it was left unmodified. However, if a value was either not available or was marked as an outlier, then the value of the previous day was stored and a sub process began counting the number of days until another value was available. Once the sub process had identified the next known value it now knew how many days lay between the last known value and the value it had finally arrived at. In order to extrapolate the values in between the sub process calculated the difference between the two known values and divided it by the number of days it had counted. The final step in the sub process was to go back to the first unknown day, add the divided difference to the day previous and assign it to the unknown day. The sub process would continue adding the divided difference until all of the

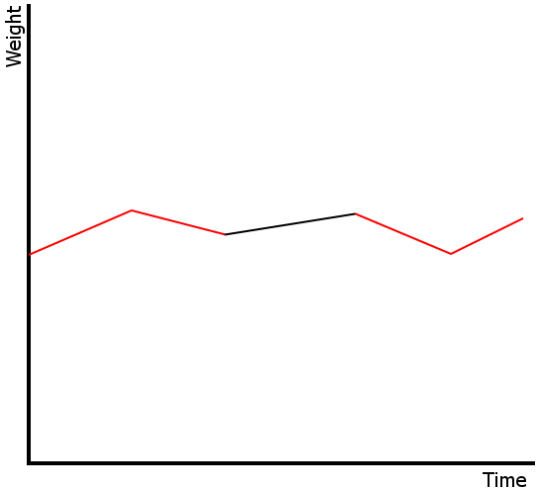


(a) A set of values containing an outlier

(b) Identifying the outlier



(c) Removing the outlier



(d) The values after the outlier is removed

Figure 5.1: An example of identifying, removing, and replacing an outlier with extrapolated values. Figures (a), (b), (c), (d) show the steps in this process

unknown days were assigned a value. Then the main process would resume until it reached the final day; initiating the sub process whenever a day had no associated value. A simplified example of this linear interpolation process can be seen in Figure 5.1.

5.5 Attribute Generation/Database Flattening

The final major transformation step in the pre-processing of the data was the act of generating attributes and as a result flattening the database tables. The process known as flattening a database is the process of generating a single table of data from a series of tables with some common link. To flatten this database a single table is created with each row in the table representing a single milking period for a single cow with its relevant milking order, weight measurement and yield measurement associated, assuming they are all available for the period. Once this flattening has been performed the attribute generation can begin.

Flattening this database was done using a single large JOIN SQL statement executed on the MySQL database. This JOIN statement was executed for each animal in the herd, over all the milking dates available and gathered the milking order, weight and yield information for both the morning and afternoon milking periods each day. Once the statement completed, the flattened data for that animal could be processed by the attribute generation steps.

With the exception of the actual daily values for a given milking period, all attributes are calculated over a time window. These time windows represent values over the days before the milking period the attributes are being generated for. In all cases the last day in a time window is the day the window is generated for. The size of the time windows are 7 days, 14 days, 21 days and, although not present in early experiments, 28 days.

In this investigation there are four sources of attributes: the milking order, the animal weight, the milk yield and the milking speed. When speaking of generated attributes they are known as the rank, weight, yield and speed attributes. Each of these sources is used to generate the attributes described below. The attributes used in each experiment will be clearly shown in the experiment setup chapter as not all attributes were used in every experiment. As more information was available, additional attributes were added and some existing attributes were modified. A description of these attributes and how they were generated follow.

Morning and Afternoon Values - These attributes are the only ones to exist outside of a time window and there are two per attribute source. These attributes represent the rank, weight, yield or speed values for the morning and afternoon milking periods for the day attributes are being generated for. In later experiments, in an attempt to remove daily herd variance, these attributes were modified to instead represent how the values of the current cow compared to those of the entire herd. This was achieved originally by subtracting the mean value of the entire herd from the unmodified value for the current animal and then dividing by the herd standard deviation. Later this method was refined to only subtracting the mean and not diving by the standard deviation, the reason being that dividing by the standard deviation would create very small numbers which could lose precision during the later ARFF processing stages.

Mean Value over a Window - For each window size and each attribute source there is an attribute generated which represents the mean value of all the days in the entire time window. This value is calculated by taking the mean of all the morning values present in the time window, the use of the morning value is due to information provided by LIC that the morning milking values are less likely to be affected by day to day changes such as weather conditions or the amount of food consumed by the herd.

Standard Deviation over a Window - Another window attribute, the standard deviation, is a simple statistic measured over each window size to capture the variance of an attribute source during the period the window covers.

Distance/Deviations between Morning and Afternoon Values - This attribute underwent a name change when it had its calculation altered. Originally, just like the base morning and afternoon values with the herd mean, it was calculated by subtracting the afternoon value from the morning value and dividing by the standard deviation of the window. However, the attribute simply represented the distance between the two values once the division by the standard deviation was later removed. Because the Morning and Afternoon values are outside of the time windows, when this change occurred, this attribute no longer varied between the windows; this meant it was duplicated in each window.

Distance/Deviations from Mean - This attribute is an adjustment, similar to the daily herd adjustment performed on the base morning and afternoon values, on a per cow basis. To calculate the attribute, the mean of the window is subtracted from the base morning value. Just as the deviations between morning and afternoon attribute began as the distance divided by the window standard deviation, so too did the distance from mean attribute. Similarly, it was also renamed when the calculation was altered.

Min and Max Values over a Window - These two attributes are simple, they simply hold the smallest and largest, respectively, morning values over the time window for each of the four attribute sources, rank, weight, yield and speed. Aside from changing implicitly when the base morning attribute was adjusted for the daily herd variances these attributes have remained the same in all experiments.

Range of Values over a Window - This attribute is present in addition to the min and max attributes to show how wide the gap in values is. Although this information is available implicitly, through the min and max attributes, some classifiers must be explicitly informed of this distance to be able to use it in learning a model.

Is Heat/Mating - Two potential class attributes are generated in the same way, the first being if the cow was recorded as being in heat on the current day, the second being if the cow was recorded as being mated on the current day. These class attributes are generated as the target for the classifier to learn; if more than one is generated only one should be present after the ARFF file generation step.

5.6 Creating ARFF Files

This is the final step before actually using a generated ARFF file to run experiments on various classifiers. Although a trivial step once all the pre-processing steps above are completed, the actual generation of an ARFF file is not completely straight forward. Until this step all pre-processing attempted to keep as much data as possible, however it is sensible at this point to use the knowledge we have about dairy farming to make the dataset of machine learning instances as specific as possible.

The first key to the dataset generation is to use the knowledge gained from introspection of the database and confirmed by LIC experts. That knowledge is that, with only an insignificant number of exceptions, it can be assumed that all dairy cows in a herd will be in heat within a three month window for the data provided. This was between October 1st and December 31st for both available seasons. With this information the data used to create each ARFF file for a herd can be narrowed to simply be anything falling between these months. This can save considerable amounts of time when running

experiments. Additionally, by limiting the number of non-heat instances in the dataset the strongly skewed distribution of heat to non-heat events is, at least slightly, reduced.

Secondly, if there are any values which were not replaced as part of the outlier removal process and remain without values after the attribute generation process then these values must be given the question mark denotation for a missing value in the ARFF file. This might occur if the weight or yield attribute sources are simply not available for a herd or if they are not available for the entire four month heat period discussed above. With the missing values represented in the ARFF file it is possible to use a replace missing values filter inside WEKA to replace all missing values with the mean value for the given attribute and generate a new dataset with values for all attributes.

The third and final part of the ARFF file generation step is to select which of the generated class attributes to use. Either heat or mating events can be added as the class attribute to be learned by the machine learning classifiers; both cannot be added simultaneously as they are directly correlated and would give very optimistic results to the prediction of the other. Each experiment will mention which of the class variable it was predicting.

Chapter 6

Experiment Setup

Over the time this investigation was conducted, many experiments were performed. These experiments started from experiments on various classifiers using a single 7 day time window of attributes. Eventually experiments became larger and focused on the most successful classifiers with attributes generated over four lengths of time: 7 day, 14 day, 21 day and 28 day windows were used for generating attributes in the most robust experiments.

In addition to the attribute pool evolving as the investigation progressed, there was also a point at which it became known that using mating events as the prediction goal was an option and that potentially training examples would be more accurate using mating events instead of the originally used heat events. The basic justification for the improvement in training examples was that mating events represent a service, a cow insemination attempt, to the farmer who was recording the data. It was suspected that data might be better recorded when this service was performed as it represented an event of financial significance to the farmer. This is in obvious contrast to the heat events, which are meant to be recorded by a farmer when he is made aware of a sign of heat in a cow through some means. It is certainly believable an event could be recorded inaccurately, or not at all, for a sign of heat.

Finally, as more time passed, there was opportunity for more data to be used in experiments. Initially experiments could only be conducted on the two 2006 herds available from the start of the investigation. However, as time progressed, three new herds were introduced allowing for 2007 herd data to be experimented on as well.

More specific details of the refinement steps in experiment parameters are outlined in the experiment summary sections.

6.1 Heat Prediction using a Single Time Window of 7 Days

Although there were plenty of informal experiments performed before any of the experiments described here, the first notable round of experiments performed were those intended to predict heat events using attributes generated over a time window of 7 days.

The choice to use attributes generated over a time window was strongly inspired by the work by Mitchell, Sherlock and Smith[25] (described in detail in Section 2.4). A slightly wider window, based over a week, was chosen in an attempt to capture longer term trends than the 5 day windows used in their investigation. For every day with data available, a 7 day window representing the current day and the previous 6 days is used to generate attributes for the instance to represent the state of the current day. This is important as it ensures experiment instances do not represent any future data at any given time.

In these early experiments the basic morning and afternoon measurement attributes did not have the daily herd adjustment, discussed in Section 5.5, applied to them. As the other attributes contrast these daily milking values to the aggregated values over the windows all attributes were affected as a result.

As this was the first set of official experiments, heat events were used as the experiment target class for all experiments at this time. As explained generally, in Section 5.5, an instance in an experiment is only considered a heat event when the final day in a time window has been recorded as a heat event in the data. These experiments were performed on the 2006 herd data as the 2007 data was not yet available.

While the number of time windows to use was being explored, during the first three experiments, the only classifier used was Random Forest.

6.2 Heat Prediction using Multiple Time Windows of 7/14/21 Days

The next set of formal experiments increased the number of attributes available to classifiers in an attempt to expand representation of each day of milking. The information added came in the form of additional time windows; 14 day and 21 day windows. By including data that contrasted the daily measurement to those further in the past it was hoped that any longer term trends might be captured. Again, it was important that these new time windows only looked at the past 14 and 21 days, respectively, as it would not be meaningful to predict heat using future information.

As these experiments were among the earliest conducted they were predicting heat events and were conducted on the 2006 herds, A-06 and B-06, only. As with the previous experiment the morning and afternoon values were not yet being adjusted to remove the daily herd variance.

As this was the second experiment conducted and the number of time windows to use was still being explored at this stage the Random Forest classifier was the only one applied to the data.

6.3 Heat Prediction after Adding a 28 Day Window

After reporting on the attributes used in the previous experiments to some experts at LIC it was made known that a cow's oestrus cycle can be longer than 21 days for some animals. This meant there could be details lost outside the largest 21 day windows that were currently being used. It seemed important to expand time windows by another level, this came in the form of a 28 day window to generate attributes over. This new window represented the previous 27 days and the current day for a day attributes were being generated for.

The first experiments conducted after adding the 28 day window didn't include the daily herd adjustments and they were only conducted on the 2006 herd data. All experiments were still predicting heat at this stage; however these were the last few experiments to do so.

This was the last of the initial three experiments to explore the correct number of time windows to use; making this the last experiment to only apply the Random Forest classifier to its data.

6.4 Addition of Daily Herd Adjustments

Possibly the most significant alteration to the experiment parameters was the addition of the daily herd adjustment to the morning and afternoon values for each day. This adjustment is discussed in greater detail as part of Section 5.5. The basic idea is to remove the variance common to the entire herd for each daily measurement, on a day by day basis. The intended result is that any trends discovered for a specific cow will better apply to the entire herd. Additionally, and perhaps more importantly, this adjustment should help prevent a model finding trends which are basically changes observed in the entire herd, perhaps as a result of a change in paddock or food.

It was at the time of adding the daily herd adjustment to the data values that access to the first 2007 herd became available. The first 2007 herd was actually the 2007 data for the existing B-06 herd. This was very good because it provided a chance to see how the data for a herd varied between years.

The widest array of classifiers of any experiment was applied when the daily herd adjustment was introduced. The original Random Forest classifier remains from the previous three experiments and is joined by the following additional classifiers: AdaBoostM1 with J48 as a base classifier, two Bayesian based classifiers: AODE and Naïve Bayes, Bagging with unpruned J48 trees as the base classifier, Logistic Regression, SMO with the RBF Kernel and, finally, just straight J48 running without an ensemble classifier.

6.5 Mating Prediction Using Multiple Time Windows

The next significant change to the experimental procedure, second only to the addition of the daily herd adjustments, was the choice to use mating events as the prediction class. A more detailed explanation for the change is described in the attribute generation step, but basically the advantage is that mating events are considered to be better recorded by farmers. This means that a classifier predicting mating events over heat events should be able to produce a more concise model. These experiments were performed on the three sets of data available: the A-06, the B-06 and the B-07 herd.

The number of classifiers applied in this experiment was cut down; only classifiers which had preformed well in the Addition of Daily Herd Adjustments Experiment (6.4) were used. These classifiers were: AdaBoostM1 with J48 trees, Bagging with unpruned J48 trees, Random Forest and SMO with the RBF Kernel. Additionally, as it was suspected it might perform well, Ad-

aBoostM1 using Decision Stumps as a base classifier was applied to the data as well.

6.6 Removal of Standard Deviation Differences

The final minor attribute parameter adjustment was to stop representing the distance from window mean and distance between morning and afternoon values in terms of standard deviations and simply record the values as straight differences. The reasoning behind this was that it was suspected that some of the very small numbers produced by dividing by the standard deviations were being rounded when being written to disk as an ARFF file. As it wasn't important to represent the difference attributes in terms of the number of standard deviations it was decided that this process be dropped to avoid losing any attribute precision during classification.

Around the time of this change access to the C-07 and D-07 had be gained, this gave two more 2007 herds to apply classification to as part of this round of experiments. The advantage of having these extra herds was that results should better reflect the effectiveness of the process in predicting heat on other unseen herds. In its simplest form, it allowed for better comparison between results by having more herds to compare against.

As classifier performance had been well established by this point there were only four classifiers applied. The four classifiers were: Bagged unpruned J48 trees, Random Forest, SMO with the RBF Kernel and AdaBoostM1 using Decision Stumps as a base classifier. Due to the performance of the Decision Stumps over J48 trees the AdaBoostM1 with J48 trees was dropped from the list of classifiers to apply.

6.7 Comparing Effectiveness of Attributes

The experiment process had now become refined well enough to start reaching some real conclusions about the data and techniques being applied. It was suspected that the effectiveness of the attributes that were being given to classifiers might have been quite mixed. This was based on the split nodes of J48 trees and the weights applied to attributes by the support vector machines. It seemed the attributes were not all contributing the same predictive power and perhaps some were contributing very little.

In order to test the effective predictive power of each of the attribute sources it was necessary to look at each attribute source in the absence of the others. Thus, the first step was to generate three new ARFF dataset files for each of the three attribute sources, the rank (milking order), the weight and the milk yield. The yield also included the milking speed as it seemed incorrect to deal with the two separately. Once these new, single attribute source, datasets were generated there were a total of 15 data files; a rank, weight and yield dataset for each herd that data was available for. However, due to the data not being present, the weight dataset for the C-07 herd and the yield dataset for the D-07 herd contained no instances. The result was 13 usable datasets, a complete 5 sets for rank, 4 for weight and 4 for yield.

The experiments performed to explore the effectiveness of each attribute on their own used all the modifications made to the experiment process thus far. This meant that mating events were being predicted, the daily herd adjustment had been applied, attributes were being generated over all four window sizes, right up to the 28 day window, and access to data for all the herds was available.

As this experiment was about comparing the effectiveness of attributes it was important to only pick one classifier to compare with. Bagging with unpruned J48 trees as the base classifier was selected to be applied to each set of data.

6.8 Application of Classifiers Designed to Target Skewed Data

Chapter 3 mentions the use of classifiers designed specifically to target skewed datasets. As the data provided by LIC contains a skewed class distribution it made sense to apply the general techniques for skewed problems to this data. The actual skew present in each herd varied a little, but all were close to 1% mating event days versus 99% non-mating event days.

As Under Bagging and Roughly Balanced Bagging were the classifiers which seemed to provide the best results on the datasets used in Chapter 4 they were selected to be used in this experiment. It was expected that, as in the results of Chapter 4, a ratio of two or three majority examples for each minority example would give the best results. There was also a good chance the AUC for one or both of these techniques would compete with standard Bagging on the LIC herd data as well.

This experiment was conducted with the same settings as the experiments in Section 6.6 and Section 6.7. Mating events were predicted, the daily herd adjustment and 28 days of windowed attributes were generated and, as there was access to all five herds, all herds were used in the experiment.

This experiment applied the Under Bagging and Roughly Balanced Bagging skewed data classifiers to the data as well as the Bagging with unpruned J48 regular classifier for comparison purposes. The parameters to use with Under Bagging and Roughly Balanced Bagging were selected so that classifiers were running with ratios of: 1 majority:1 minority, 2 majority:1 minority and 3 majority:1 minority. These selections were based on the success of these ratios in Chapter 4.

Chapter 7

Experiment Results

This chapter will advance through the results as the experimental process and attribute pool was refined over time. Results follow the same structure as Chapter 6, thus each section of results refers to a single experiment setup section. A brief explanation of each experiment will be given for each section. However, for the full experiment setup, please refer to the relevant section in Chapter 6.

Each set of results has been graphed to show ROC curves as well as calculating the AUC for each ROC curve. ROC has been selected as a comparison measure as it is a well respected metric in the Machine Learning field.

Additionally, and probably far more relevant, Precision-Recall graphs have been generated using the minority class (heat or mating events) as the positive class. Along with the graphs comes an estimate of the AUC for these curves. AUC is not as simple to calculate in the case of Precision-Recall, hence the use of an approximation. The reasoning for showing Precision-Recall curves is that ROC curves can often appear optimistic on highly skewed problems.

A table of summarised results will be included to aid in comparison of the various AUC values between classifiers. This should help comparing the values from a single experiment and, by comparing these tables between sections,

show the change in graph AUC as the experimental process was refined.

Finally, a few paragraphs about observations that can be seen in the results will be given. Whenever possible, and relevant, the implications of these observations will be explained in this area as well.

7.1 Results of Heat Prediction using a Single Time Window of 7 Days

This was the first serious experiment undertaken during this investigation. As the first it was executed with the least number of attributes, being just the attributes generated over a 7 day window. There was no daily herd adjustment used at this stage and classifiers were still building models to predict heat events. This experiment only had the 2006 herds available; the B-06 and A-06 herds.

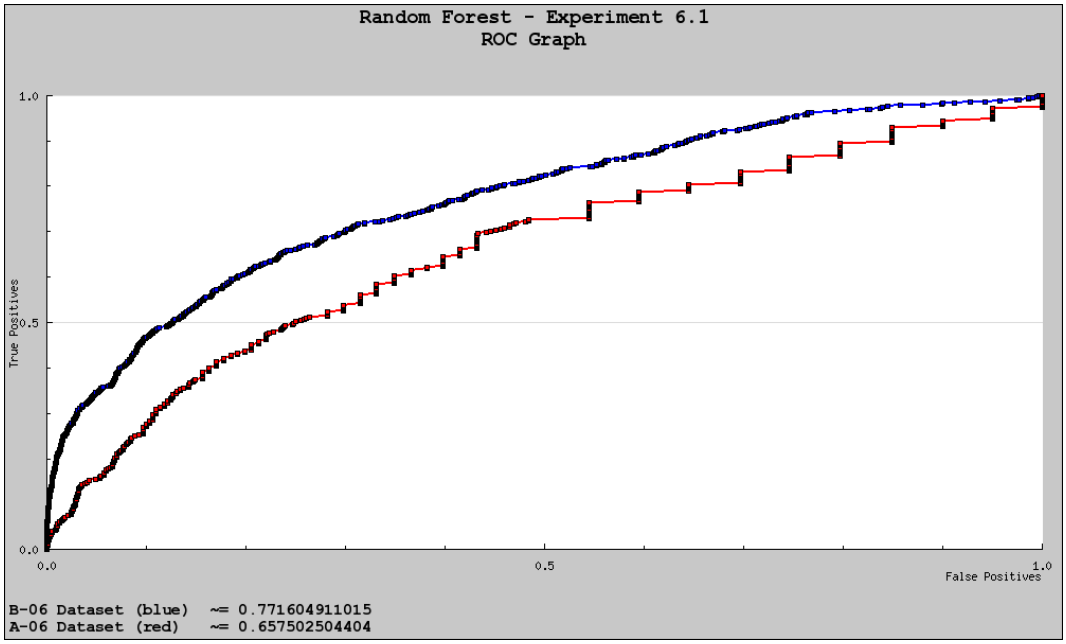


Figure 7.1: Random Forest ROC graph for Heat Prediction using a Single Time Window of 7 Days Experiment (6.1)

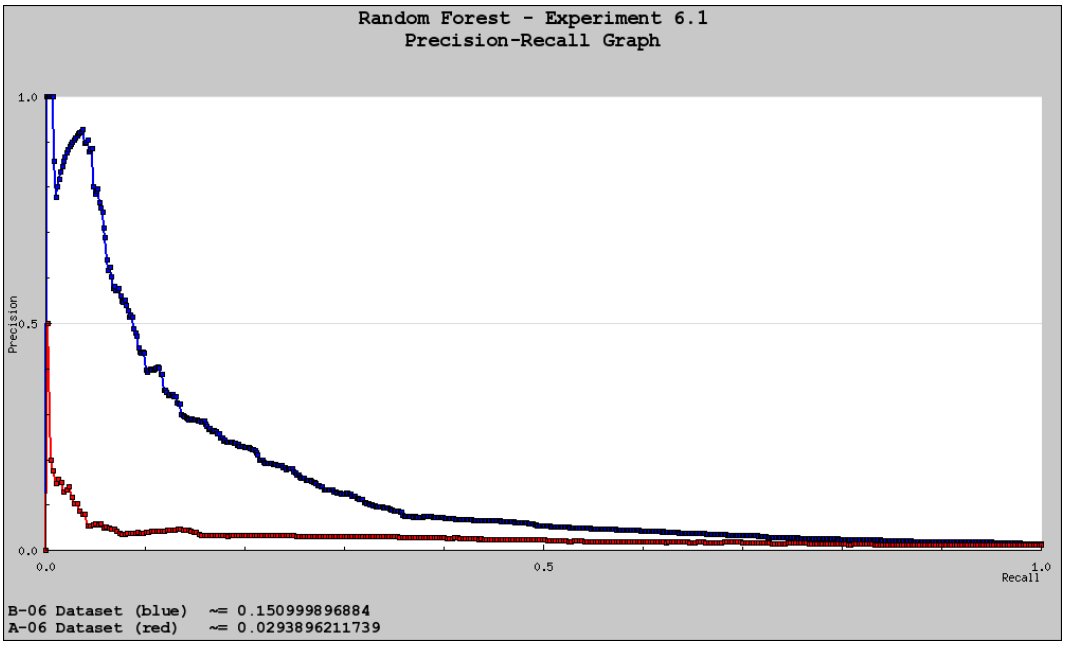


Figure 7.2: Random Forest Precision-Recall graph for Heat Prediction using a Single Time Window of 7 Days Experiment (6.1)

Classifier	B-06		A-06	
	ROC	PR	ROC	PR
Random Forest	0.7716	0.1510	0.6575	0.0294

Table 7.1: Summarised Results for Heat Prediction using a Single Time Window of 7 Days Experiment (6.1)

The ROC and Precision-Recall graphs for Random Forest, the only classifier used in this experiment, are Figures 7.1 and 7.2 respectively.

As this is the first experiment, there is little to conclude aside from that Precision-Recall results for both herds are unacceptable. Results for the A-06 ROC area show little improvement over simply guessing if a cow is in heat or not; this may indicate that for this herd predicting heat is difficult.

These results showed that there would need to be some serious refinement of the experimental procedure in order to get reasonable values. This experiment was the first of three to explore the number of time windows to use.

7.2 Results of Heat Prediction using Multiple Time Windows of 7/14/21 Days

A first attempt at improving results tried adding more time windows. As a result there were more attributes for each instance which would hopefully improve results. The Random Forest classifiers in this experiment were still predicting heat events and the daily herd adjustment was not being used yet. As with the Heat Prediction using a Single Time Window of 7 Days Experiment (6.1), this experiment only had the 2006 herds available.

Classifier	B-06		A-06	
	ROC	PR	ROC	PR
Random Forest	0.8112	0.1518	0.6943	0.0371

Table 7.2: Summarised Results for Heat Prediction using Multiple Time Windows of 7/14/21 Days Experiment (6.2)

Random Forest was, again, the only classifier used in these early experiments. The ROC and Precision-Recall graphs are Figures 7.3 and 7.4.

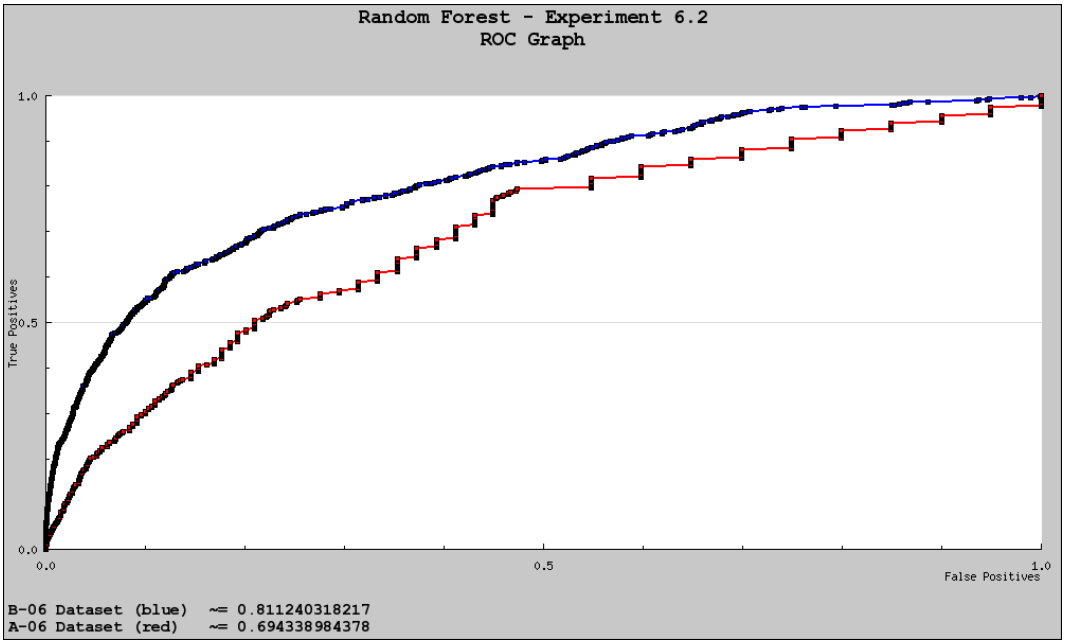


Figure 7.3: Random Forest ROC graph for Heat Prediction using Multiple Time Windows of 7/14/21 Days Experiment (6.2)

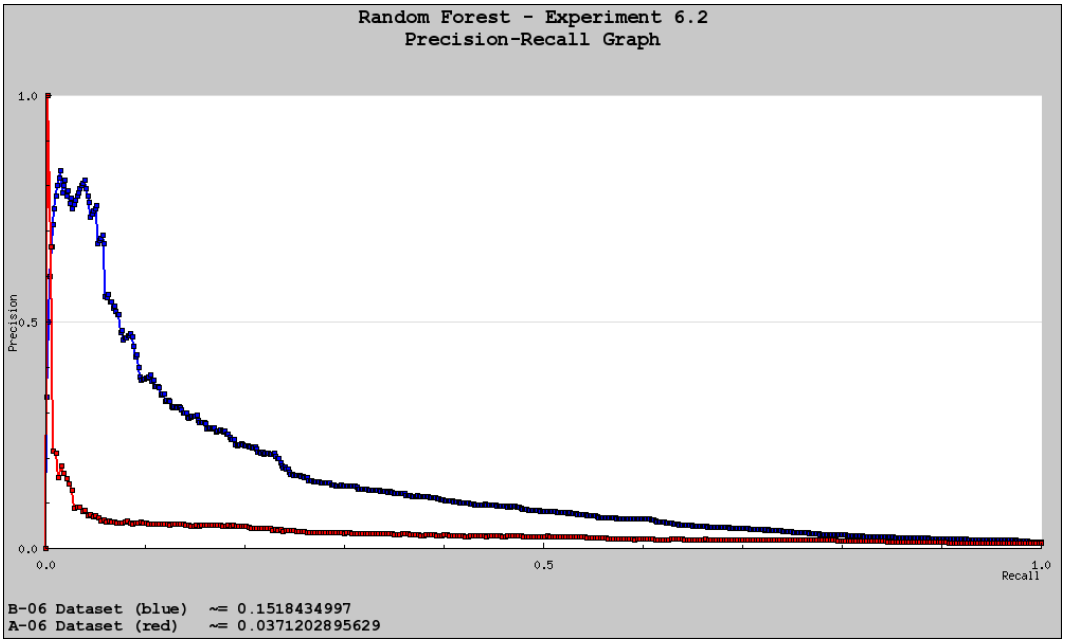


Figure 7.4: Random Forest Precision-Recall graph for Heat Prediction using Multiple Time Windows of 7/14/21 Days Experiment (6.2)

It is reassuring to observe that adding additional time windows has increased the Precision-Recall and ROC areas for the B-06 herd by a reasonable amount. ROC area for the B-06 herd has improved; raising from 0.7716 to 0.8112. The A-06 herd has moved towards a more acceptable ROC area but Precision-Recall area still remains very poor.

These results suggest there is much to be gained from refining the experiment process. As this was only the second step in the refinement of the experimental process it was still possible for the results to improve even further.

7.3 Results of Heat Prediction after Adding a 28 Day Window

It was originally thought that the 21 day window would be getting all the necessary long term information. However, it became known, through consulting with LIC experts, that cows could experience an oestrus cycle that lasted longer than 21 days. Therefore, in case there was some information being missed, a 28 day window was added.

The full set of window sizes was now present but the daily herd adjustment had not yet been applied. The only herds available at this stage were the two 2006 herds. Classifiers predicted heat events for this experiment.

Classifier	B-06		A-06	
	ROC	PR	ROC	PR
Random Forest	0.8052	0.1435	0.6590	0.0370

Table 7.3: Summarised Results for Heat Prediction after Adding a 28 Day Window Experiment (6.3)

This is the final experiment where the only classifier used was Random Forest. The ROC and Precision-Recall graphs are Figures 7.5 and 7.6.

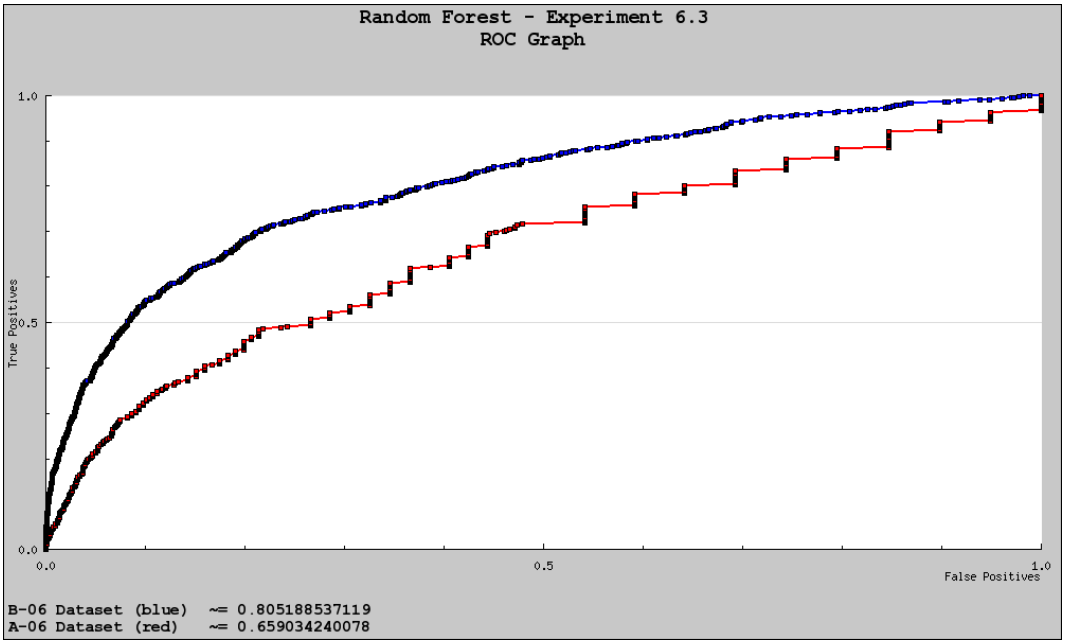


Figure 7.5: Random Forest ROC graph for Heat Prediction after Adding a 28 Day Window Experiment (6.3)

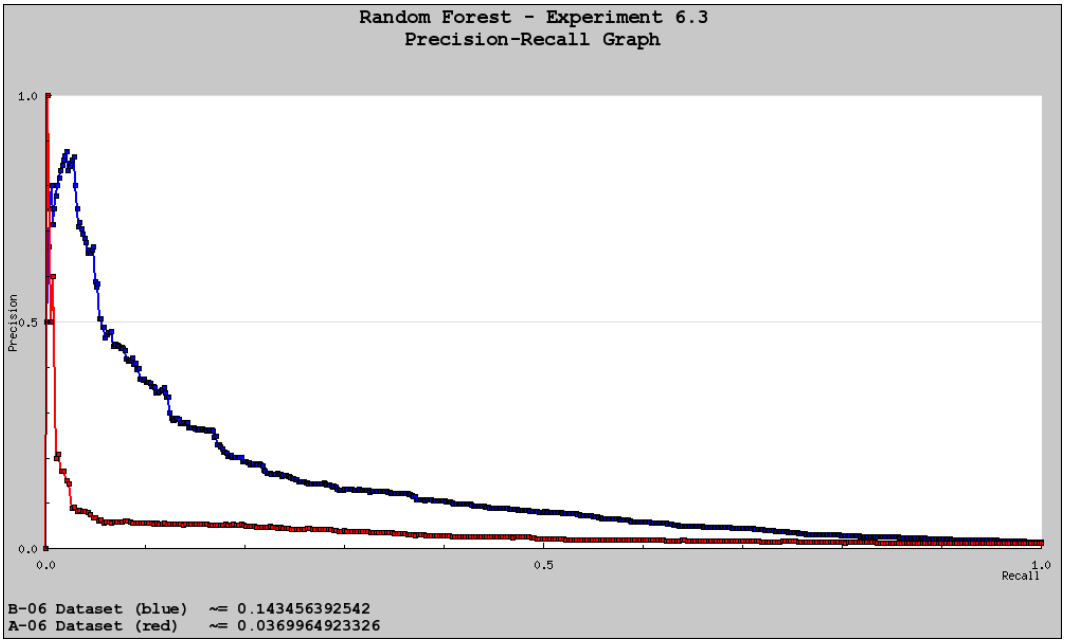


Figure 7.6: Random Forest Precision-Recall graph for Heat Prediction after Adding a 28 Day Window Experiment (6.3)

It can be observed in Table 7.3 that the results for both the herds have generally deteriorated, as a result of adding the 28 day window. In the case of the A-06 herd the ROC area has dropped back to the 7 day window value. However, the other values have not lowered so drastically.

As the values have shifted around a little having added the 28 day time window, it is possible that that not all the time windows are contributing to predicting heat. However, the logical next step is applying more classifiers to the experiment to get an idea of the general effectiveness of classifiers, rather just looking at results from Random Forest.

7.4 Results of Addition of Daily Herd Adjustments

This is the first experiment to include daily herd adjusted attributes. This means the attributes for each cow were adjusted to remove variance present in the entire herd. The goal was to help classifiers build more generalised models; models that would more accurately model the entire herd.

This experiment was the last to predict heat events and the first to include the B-07 herd. This brought the number of herds up to 3; the B-06, A-06 and now the B-07.

As this experiment introduced multiple classifiers, and showing them all would rather clutter the display, only the highest performing classifiers, Bagged J48 and Random Forests, have been selected to show graphs for. Essentially the shape of the curves is the same for the other graphs, just with less area beneath them. One could imagine them being shifted down towards the x-axis.

This experiment was the first to introduce the B-07 herd. This allowed for comparison between the previously more successful B-06 herd and its 2007

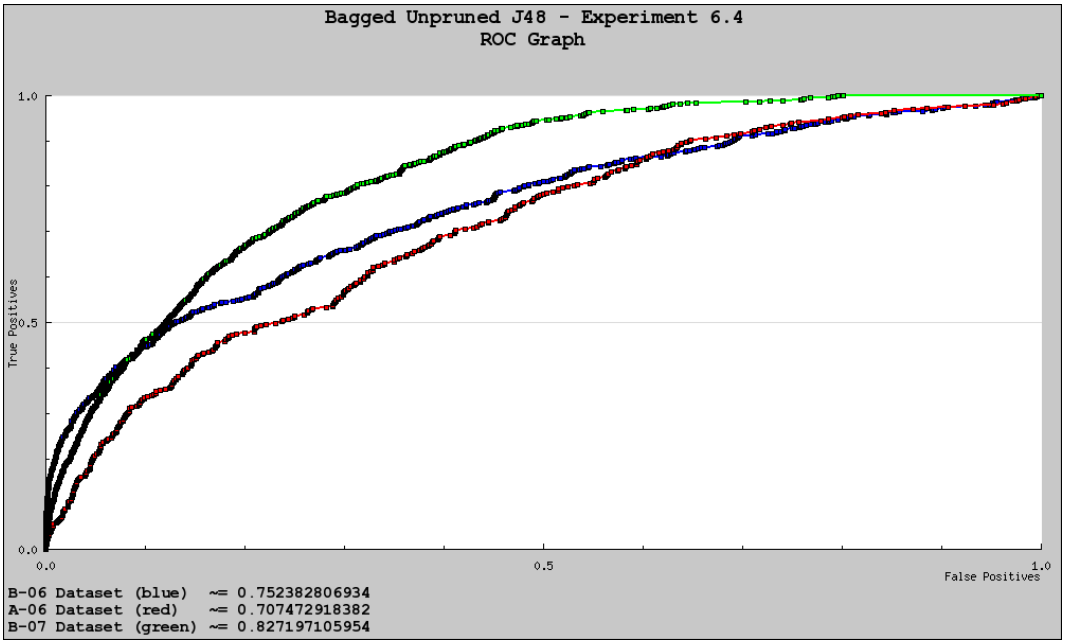


Figure 7.7: Bagged Unpruned J48 ROC graph for Addition of Daily Herd Adjustments Experiment (6.4)

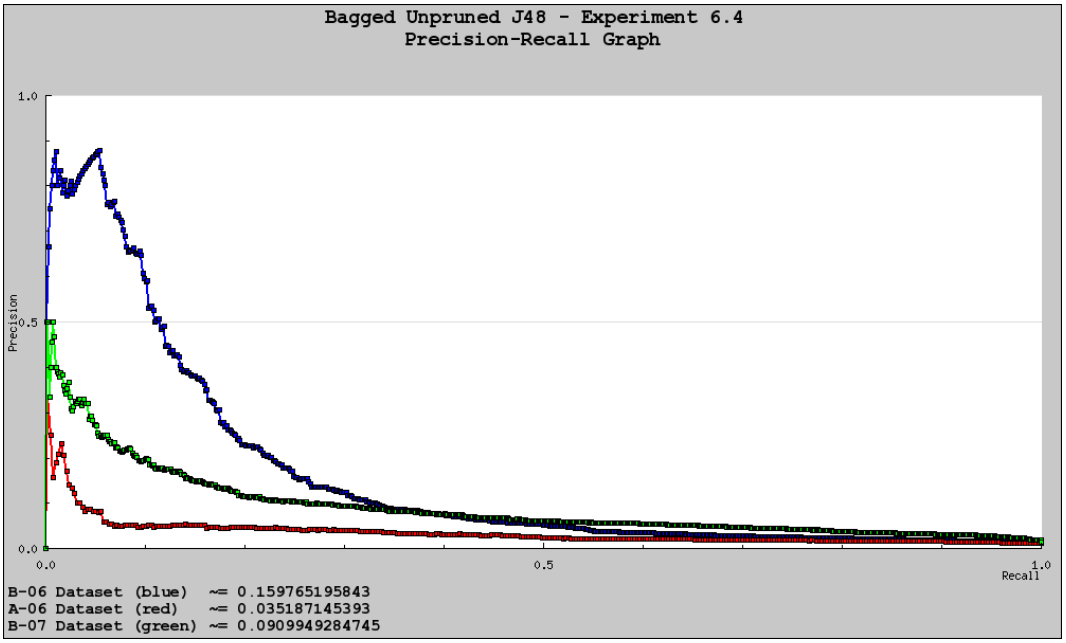


Figure 7.8: Bagged Unpruned J48 Precision-Recall graph for Addition of Daily Herd Adjustments Experiment (6.4)

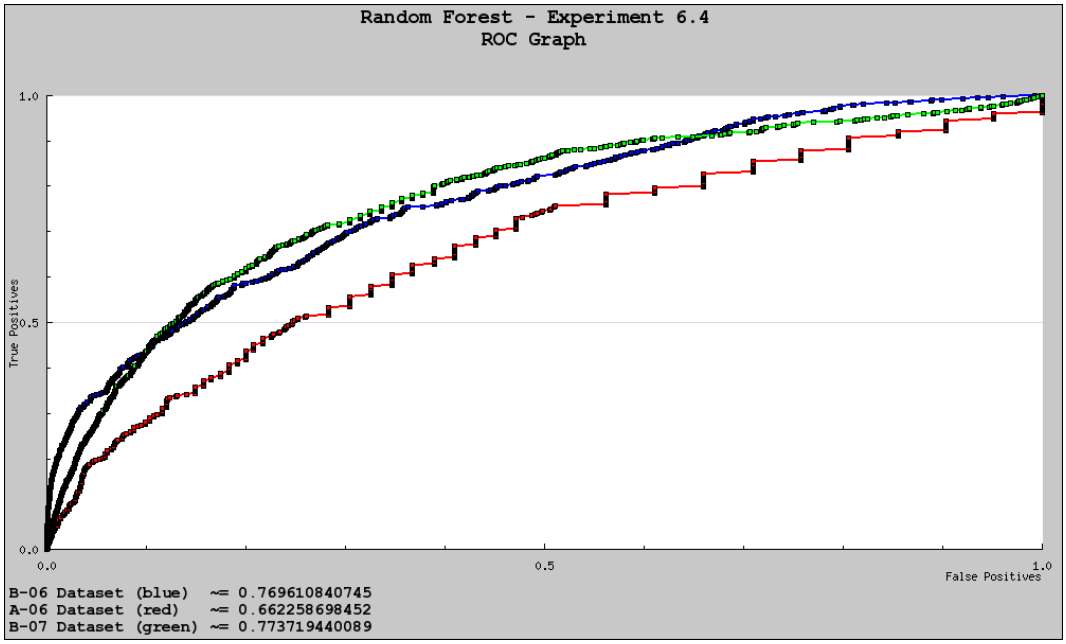


Figure 7.9: Random Forest ROC graph for Addition of Daily Herd Adjustments Experiment (6.4)

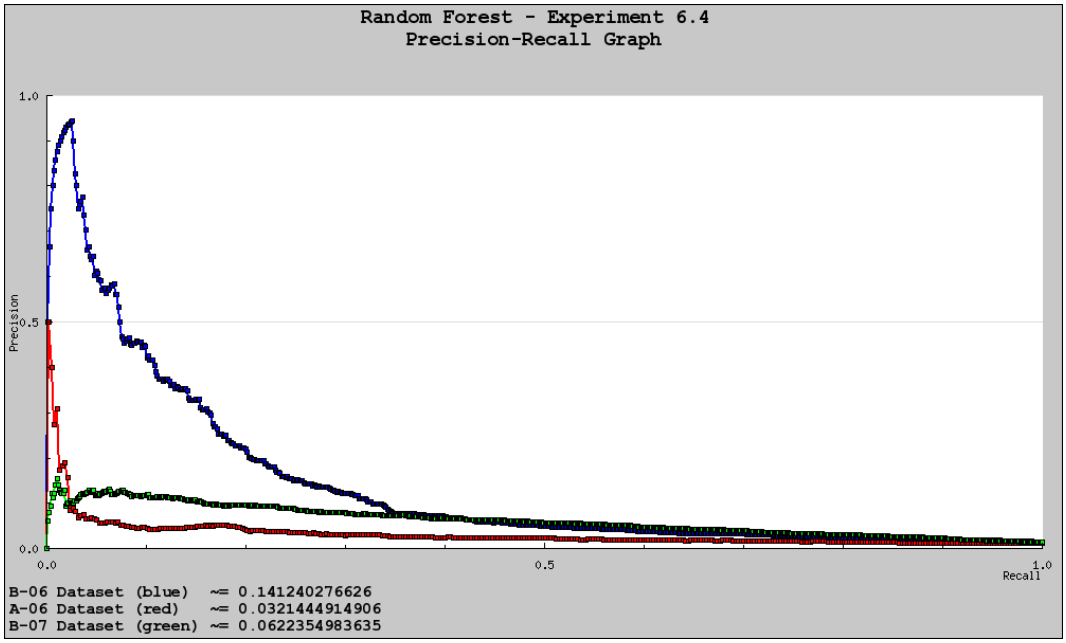


Figure 7.10: Random Forest Precision-Recall graph for Addition of Daily Herd Adjustments Experiment (6.4)

Classifier	B-06		A-06		B-07	
	ROC	PR	ROC	PR	ROC	PR
AdaBoosted J48	0.7248	0.1331	0.6380	0.0277	0.7672	0.0605
AODE	0.7474	0.0815	0.6500	0.0192	0.8001	0.0613
Bagged Unpruned J48	0.7524	**0.1598**	**0.7075**	**0.0352**	**0.8272**	**0.0910**
J48	0.5768	0.0788	0.4485	0.0099	0.6337	0.0360
Logistic Regression	0.7097	0.0695	0.6583	0.0229	0.7632	0.0646
Naïve Bayes	0.5529	0.0195	0.6130	0.0169	0.6663	0.0281
Random Forest	**0.7696**	0.1412	0.6623	0.0321	0.7737	0.0622
SMO with RBF Kernel	0.7200	0.0976	0.6128	0.0190	0.7043	0.0495

Table 7.4: Summarised Results for Addition of Daily Herd Adjustments Experiment (6.4)

counterpart. The results were unexpected. While the B-07 herd performs very well in terms of ROC area, surpassing its 2006 counterpart with all but the SMO classifier, it achieves lower areas for Precision-Recall on all but the Naïve Bayes classifier.

In addition to the unexpected results seen for the new data, the B-07 herd, there was still much to gain from this experiment as it was the first to start using multiple classifiers. It can be seen by viewing Table 7.4 how each classifier performed. Bagged J48 trees have come out on top for all herds in terms of Precision-Recall area, followed quite closely by Random Forest; the only classifier to beat Bagged J48 for one of the ROC areas. AODE does very well in terms of ROC area but falls far short when it comes to Precision-Recall. AdaBoosted J48, Logistic Regression and SMO using an RBF Kernel are all very close in performance. The performance of classifiers in this experiment dictated which classifiers were used in future experiments.

7.5 Results of Mating Prediction using Multiple Time Windows

This experiment is the first to use mating events instead of heat events. The reason for predicting mating events instead of heat events is that mating events should be better recorded ensuring more accurate models are built. See Section 5.5 where the choice to use mating events is explained in full detail.

The three herds mentioned already, the B-06, A-06 and B-07 herds, were used in this experiment. The daily herd adjustment was used for this experiment.

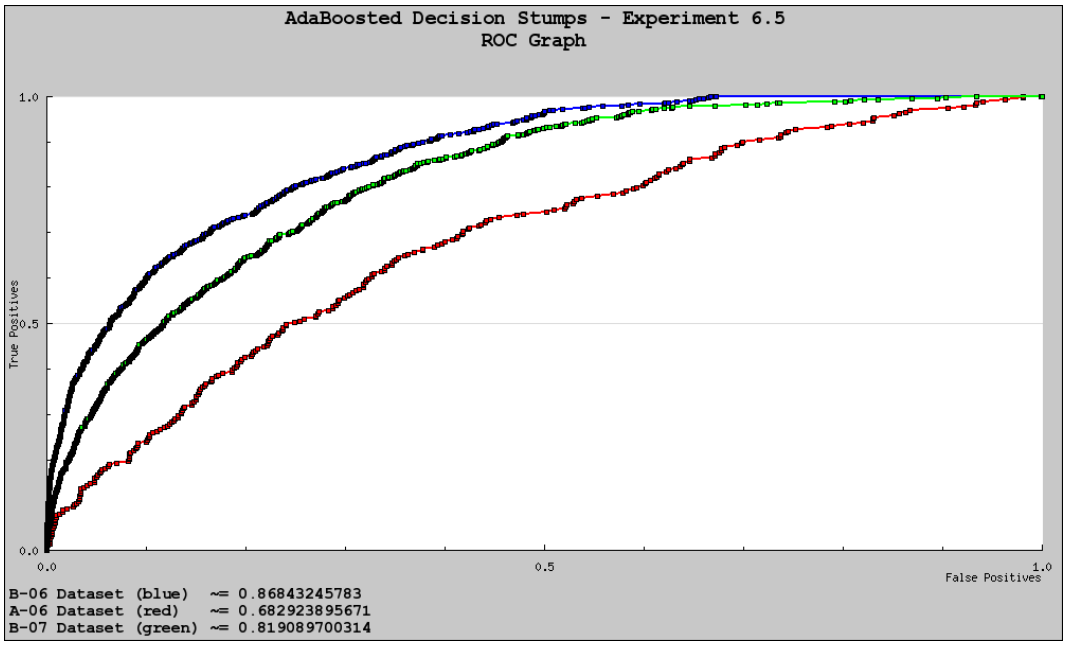


Figure 7.11: AdaBoosted Decision Stumps ROC graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

AdaBoosted Decision Stumps, Bagged J48 and Random Forests were the highest performing classifiers in this experiment, thus, in the interest of avoiding so many similar graphs, these are the ones shown. The graphs for the other classifiers are, again, very similar in shape.

The effects of using mating events are immediately noticeable on the B-06 herd; showing a dramatic increase in ROC and Precision-Recall areas. Unfor-

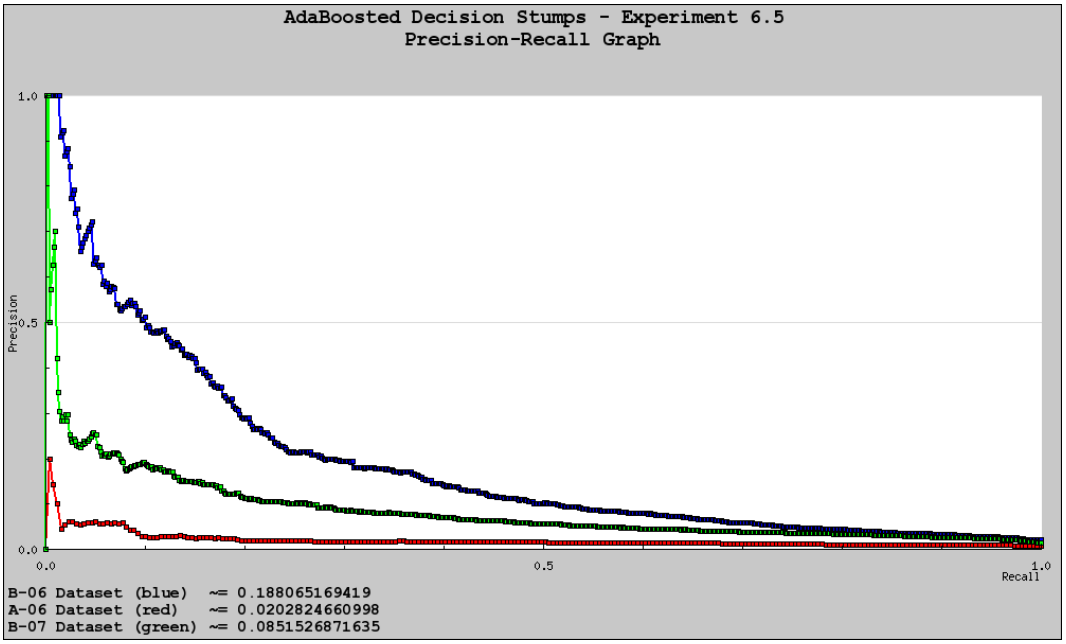


Figure 7.12: AdaBoosted Decision Stumps Precision-Recall graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

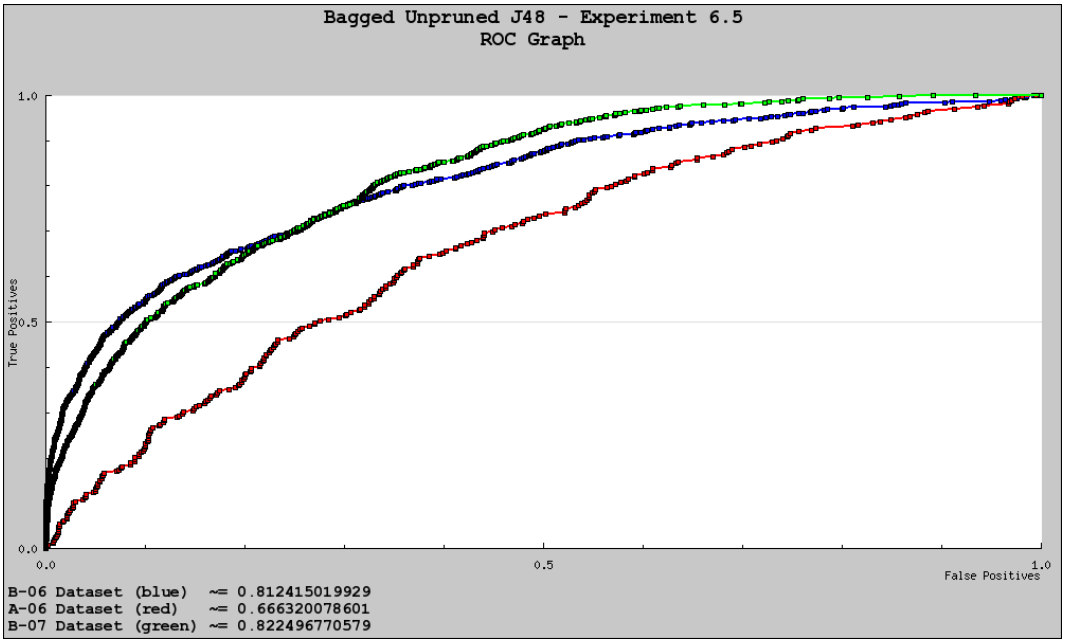


Figure 7.13: Bagged Unpruned J48 ROC graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

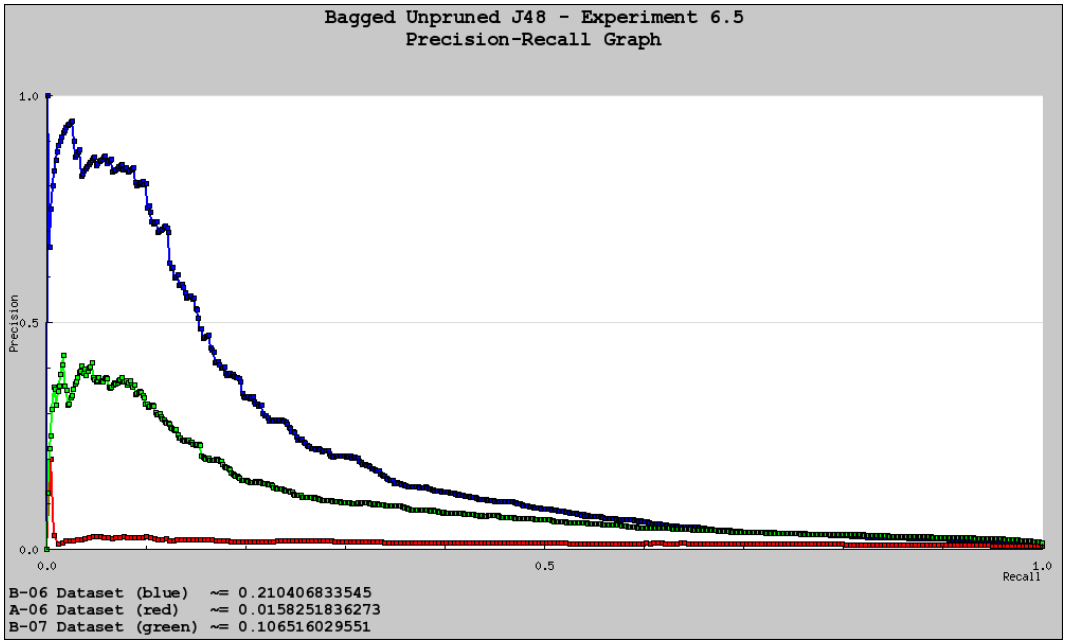


Figure 7.14: Bagged Unpruned J48 Precision-Recall graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

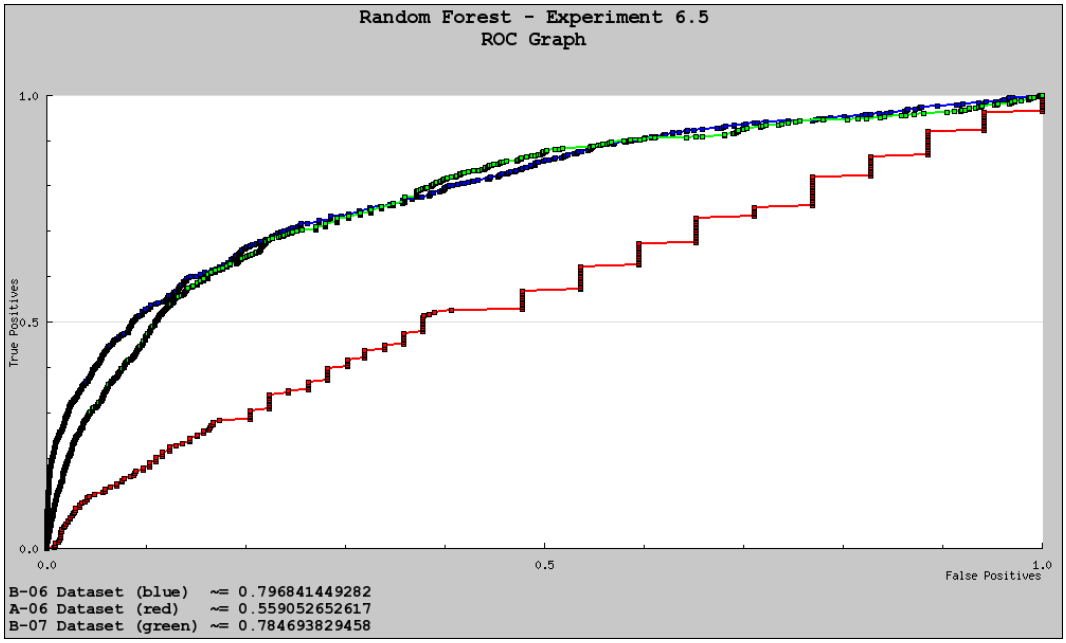


Figure 7.15: Random Forest ROC graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

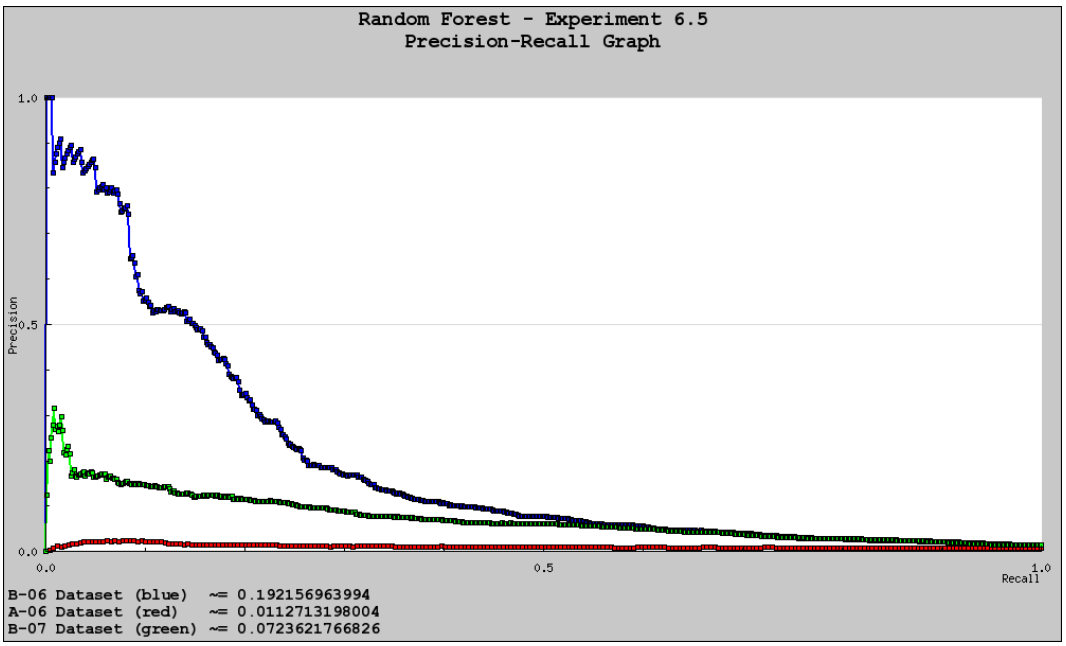


Figure 7.16: Random Forest Precision-Recall graph for Mating Prediction using Multiple Time Windows Experiment (6.5)

Classifier	B-06		A-06		B-07	
	ROC	PR	ROC	PR	ROC	PR
AdaBoosted J48	0.7874	0.1890	0.5908	0.0120	0.7784	0.0825
AdaBoosted Stumps	**0.8684**	0.1881	**0.6829**	**0.0203**	0.8191	0.0852
Bagged Unpruned J48	0.8124	**0.2104**	0.6663	0.0158	**0.8225**	**0.1065**
Random Forest	0.7968	0.1922	0.5591	0.0113	0.7847	0.0724
SMO with RBF Kernel	0.7856	0.1559	0.5863	0.0106	0.7226	0.0697

Table 7.5: Summarised Results for Mating Prediction using Multiple Time Windows Experiment (6.5)

tunately, the A-06 herd seems just as resistant to the changes in experimental procedure in this experiment as in previous ones, in this case it has even lost ROC and Precision-Recall area where the other herds made some gains at least.

In terms of classifier performance, Bagged J48 has done well, but has not beaten the other classifiers on every item this time. The newly introduced AdaBoosted Decision Stumps classifier has taken some top places. There is no clearly best classifier between these two and both have quite close area values on all three herds. Additionally, because the AdaBoost with Decision Stumps classifier appeared to be doing better than AdaBoost with J48, the J48 based AdaBoost was dropped from later experiments.

7.6 Results of Removal of Standard Deviation Differences

This experiment introduced the remaining two herds; the C-07 and D-07 herds. These herds are less complete than the three herds that had already been in use. The C-07 herd contained no weight information, while the D-07 herd contained no milk yield information. This meant attributes for these particular sources could not be added to learning instances.

Additionally this experiment changed the way some attributes were represented. Instead of dividing distances between values by the standard deviations (to get the number of standard deviations difference) the numbers were simply given without modification. This was done because it was suspected these distance values might be losing precision when being saved to a file as a result of being very small fractions of standard deviations.

The daily herd adjustment was applied to the values in this experiment and the classifiers built models to predict mating events.

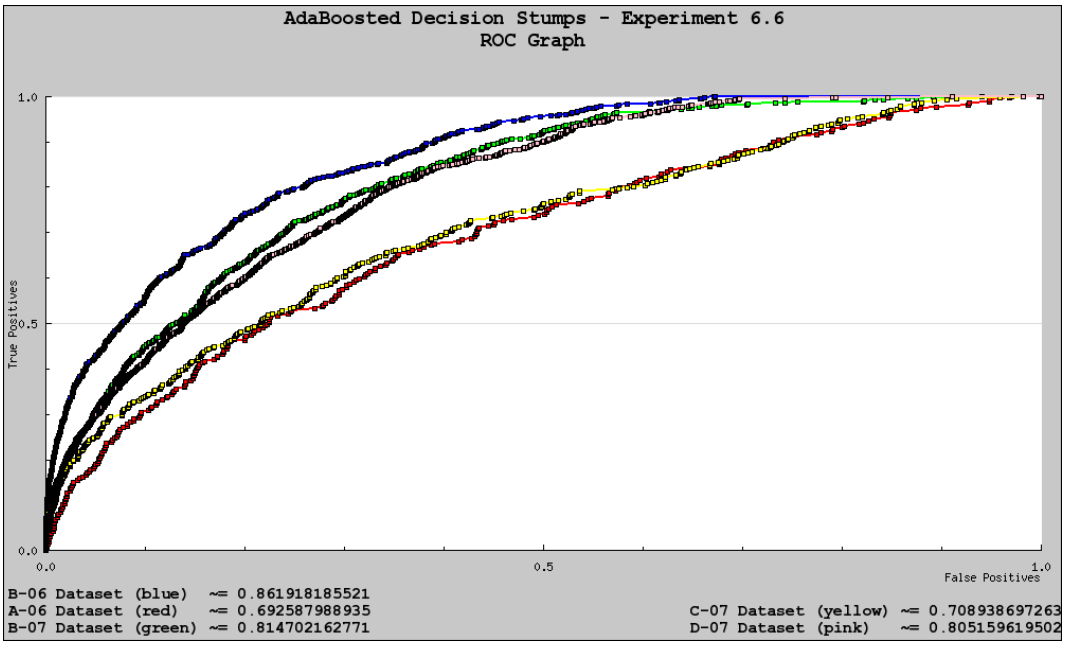


Figure 7.17: AdaBoosted Decision Stumps ROC graph for Removal of Standard Deviation Differences Experiment (6.6)

Classifier	B-06		A-06		B-07	
	ROC	PR	ROC	PR	ROC	PR
AdaBoosted Stumps	**0.8617**	0.1856	**0.6926**	**0.0222**	0.8147	0.0783
Bagged Unpruned J48	0.8083	**0.2086**	0.6625	0.0176	**0.8331**	**0.1123**
Random Forest	0.8043	0.1904	0.5739	0.0119	0.7789	0.0721
SMO with RBF Kernel	0.7826	0.1501	0.6052	0.0114	0.7091	0.0648

Table 7.6: Summarised Results for Removal of Standard Deviation Differences Experiment (6.6) - A-06, B-06 and B-07 herds

AdaBoosted Decision Stumps, Bagged J48 and Random Forest ROC and Precision-Recall graphs are given. This experiment really only added new herds so the same classifiers were picked to be graphed.

There was no significant change in the values for the three old herds. A little increase in the A-06 herd shows it has recovered from the loss in the values experienced when mating events were introduced in the Mating Prediction Using Multiple Time Windows Experiment (6.5). Despite this turnaround the

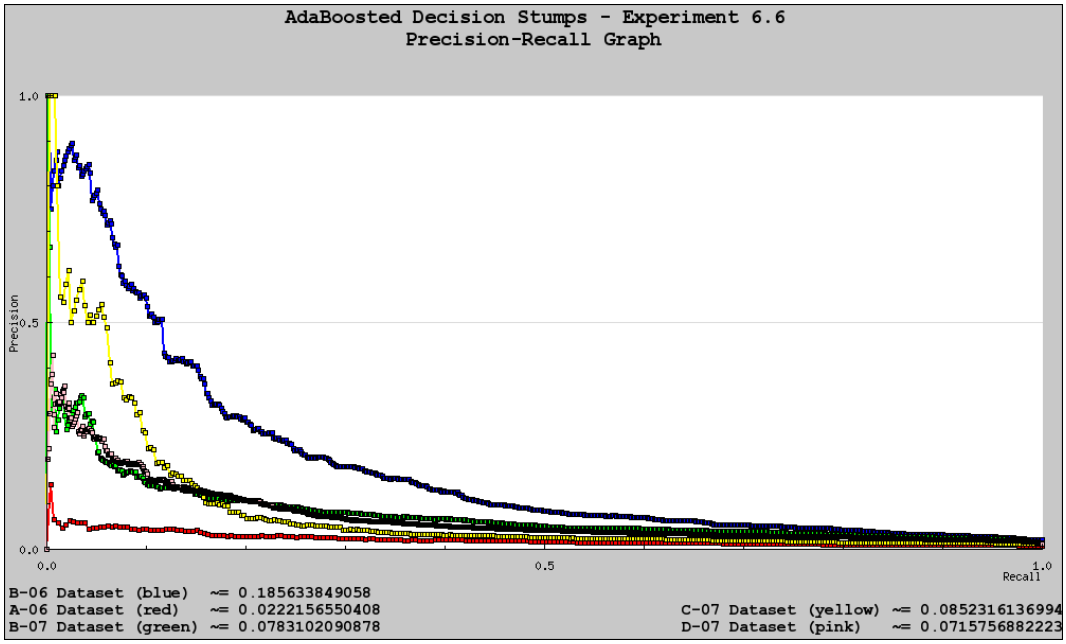


Figure 7.18: AdaBoosted Decision Stumps Precision-Recall graph for Removal of Standard Deviation Differences Experiment (6.6)

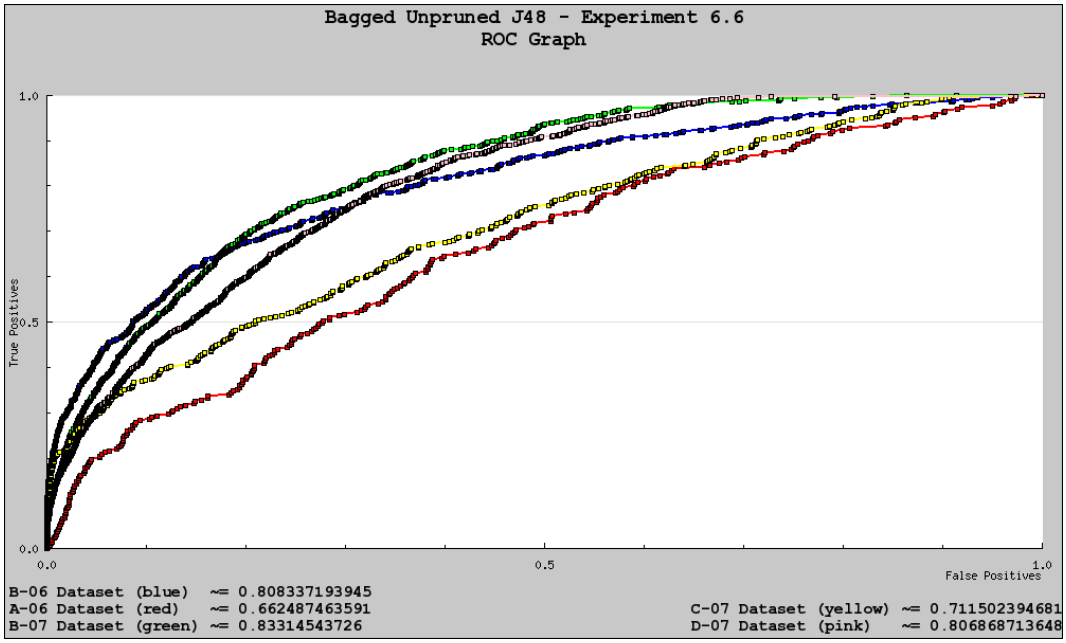


Figure 7.19: Bagged Unpruned J48 ROC graph for Removal of Standard Deviation Differences Experiment(6.6)

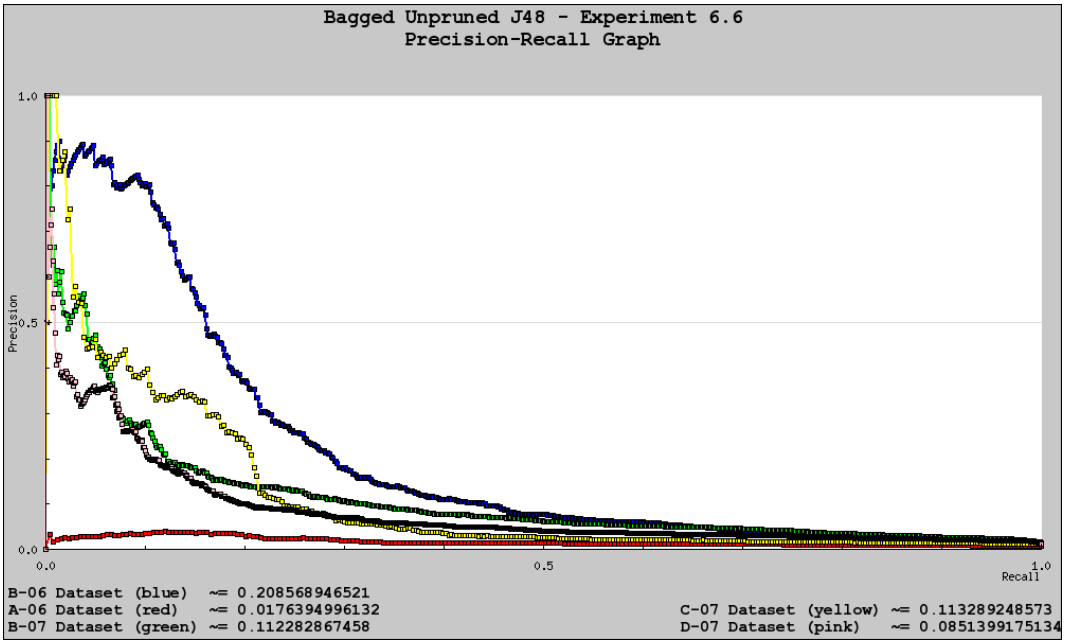


Figure 7.20: Bagged Unpruned J48 Precision-Recall graph for Removal of Standard Deviation Differences Experiment (6.6)

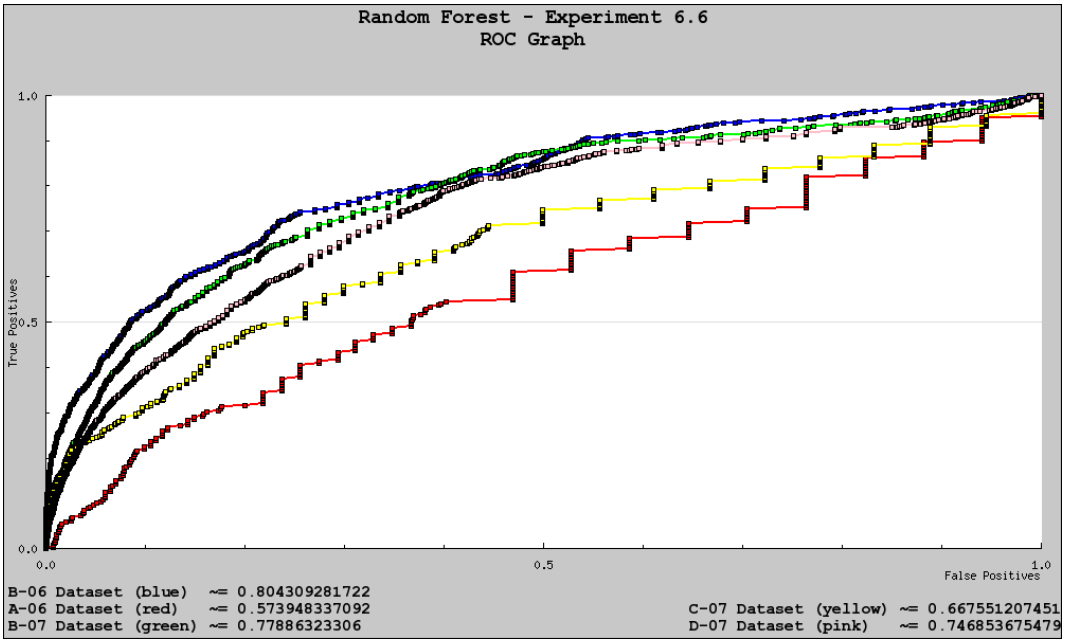


Figure 7.21: Random Forest ROC graph for Removal of Standard Deviation Differences Experiment (6.6)

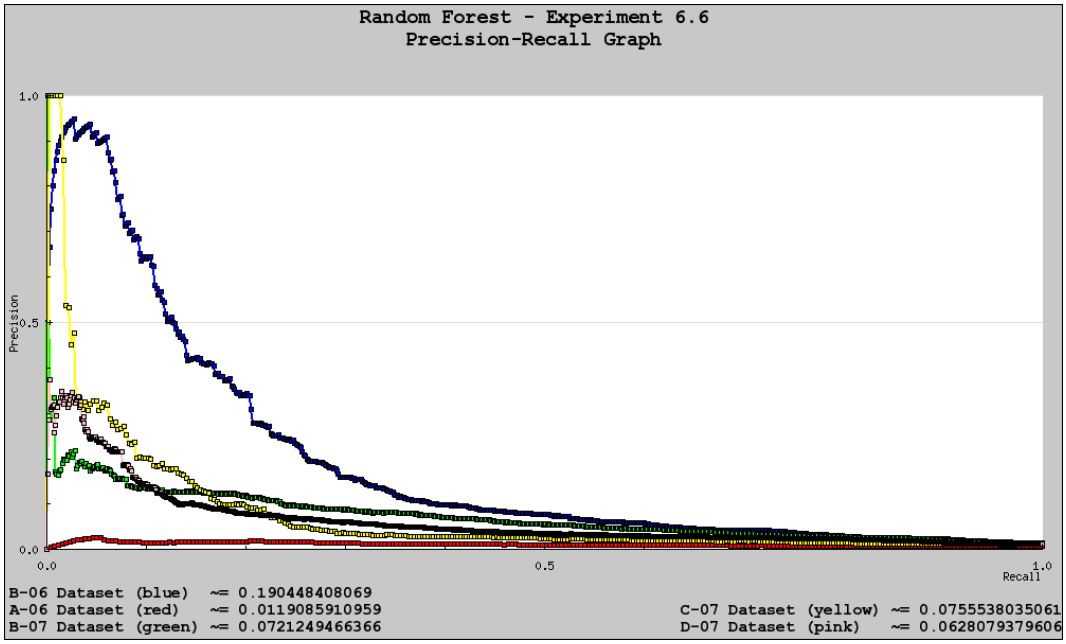


Figure 7.22: Random Forest Precision-Recall graph for Removal of Standard Deviation Differences Experiment (6.6)

Classifier	C-07		D-07	
	ROC	PR	ROC	PR
AdaBoosted Stumps	0.7089	0.0852	0.8052	0.0716
Bagged Unpruned J48	**0.7115**	**0.1133**	**0.8069**	**0.0851**
Random Forest	0.6676	0.0756	0.7469	0.0628
SMO with RBF Kernel	0.5892	0.0256	0.6236	0.0194

Table 7.7: Summarised Results for Removal of Standard Deviation Differences Experiment (6.6) - C-07 and D-07 herds

A-06 herd still shows the worst results of all the herds available.

The newly introduced herds seem to follow the trends of the B-07 herd more than the B-06 herd (this is best seen in Figures 7.18, 7.20 and 7.22). This leads to the suspicion that the B-06 herd has captured some special information in its attributes that is not available to classifiers for the other herds. This led directly into the next experiment; an experiment to identify how much predictive power each attribute source has.

7.7 Results of Comparing Effectiveness of Attributes

This experiment was conducted to evaluate the effectiveness of individual attributes. The parameters of the experiment can be summarised as follows. Each of the three attribute sources, rank, weight and yield (yield and speed are combined) are used, individually, to generate a dataset for each herd. These single attribute source files are then used to build a model with the same classifier. Bagged J48 trees were picked as the comparison classifier due to its reliably high results in previous experiments.

The Bagged J48 classifiers were predicting mating events, had the daily herd adjustment applied and were used on all five herds worth of data.

Attribute	B-06		A-06		B-07	
	ROC	PR	ROC	PR	ROC	PR
Rank	0.7229	**0.1444**	0.5697	**0.0116**	0.7270	0.0579
Weight	**0.7300**	0.0474	**0.6081**	0.0114	**0.7407**	0.0443
Yield	0.7242	0.0293	0.5448	0.0093	0.7259	**0.0626**

Table 7.8: Summarised Results for Comparing Effectiveness of Attributes Experiment (6.7) - A-06, B-06 and B-07 herds

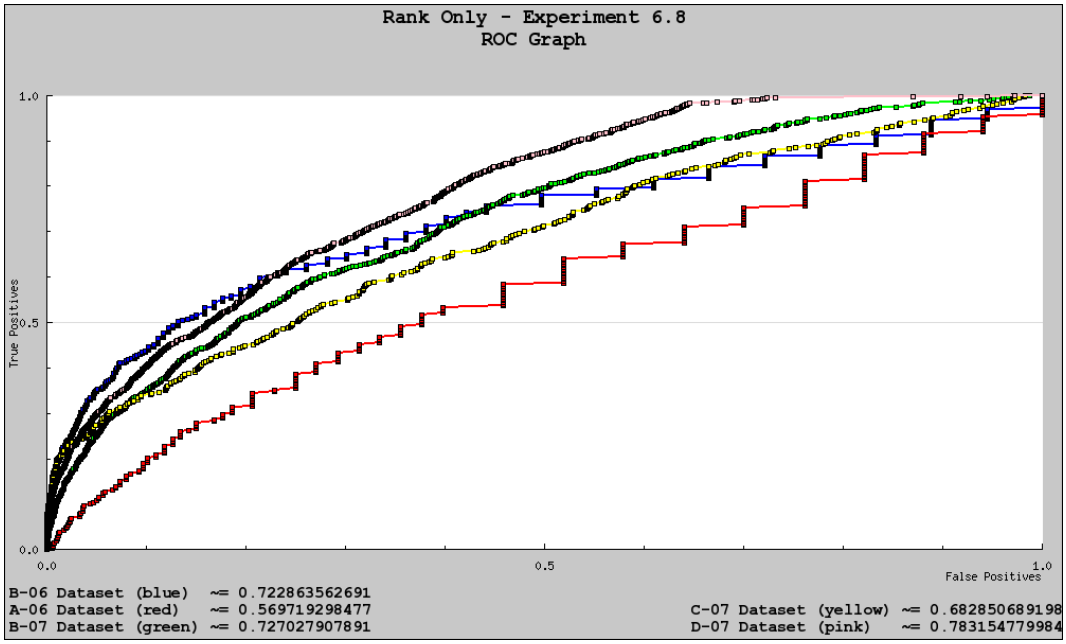


Figure 7.23: Rank Attribute ROC graph for Comparing Effectiveness of Attributes Experiment (6.7)

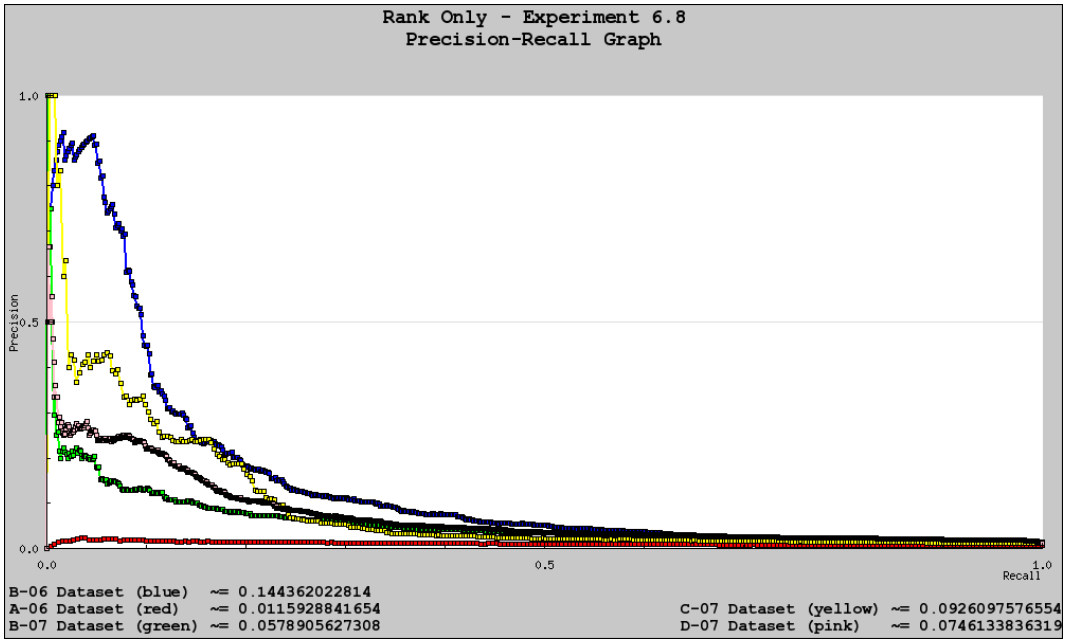


Figure 7.24: Rank Attribute Precision-Recall graph for Comparing Effectiveness of Attributes Experiment (6.7)

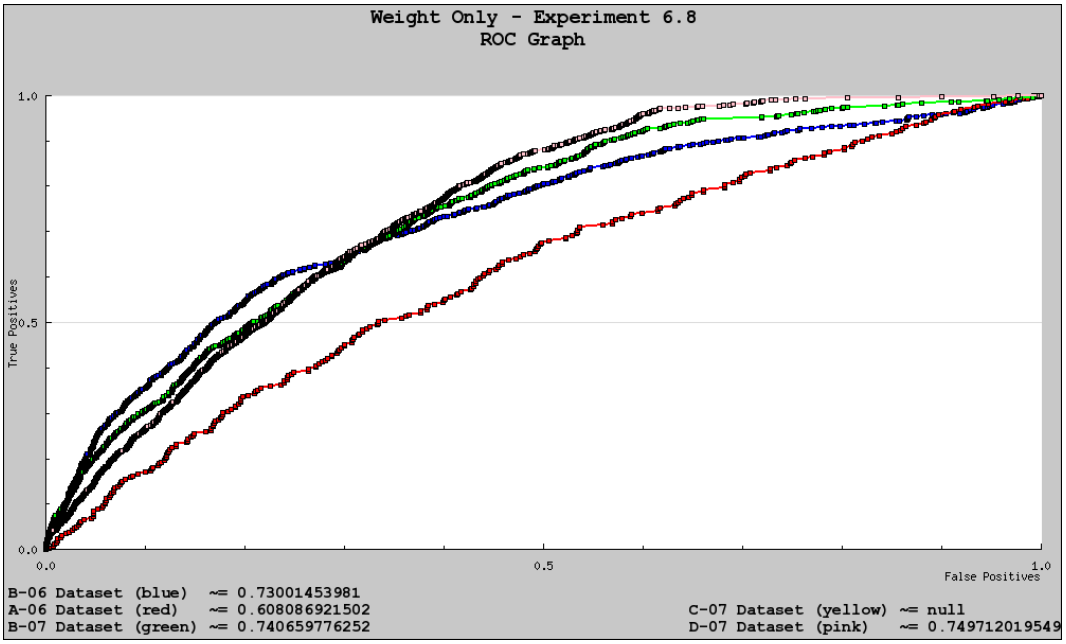


Figure 7.25: Weight Attribute ROC graph for Comparing Effectiveness of Attributes Experiment (6.7)

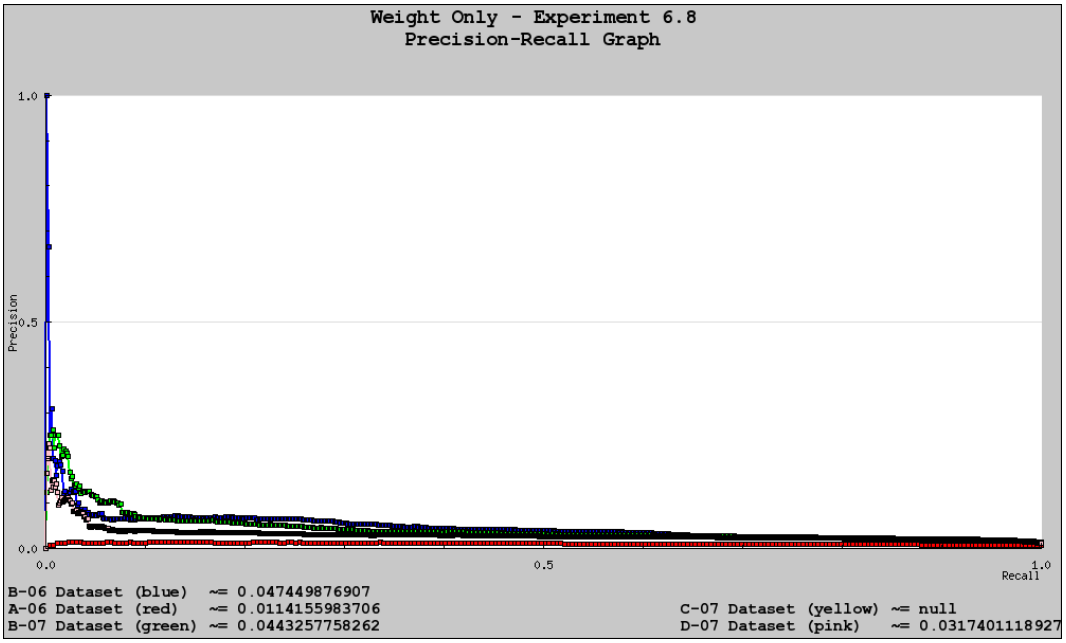


Figure 7.26: Weight Attribute Precision-Recall graph for Comparing Effectiveness of Attributes Experiment (6.7)

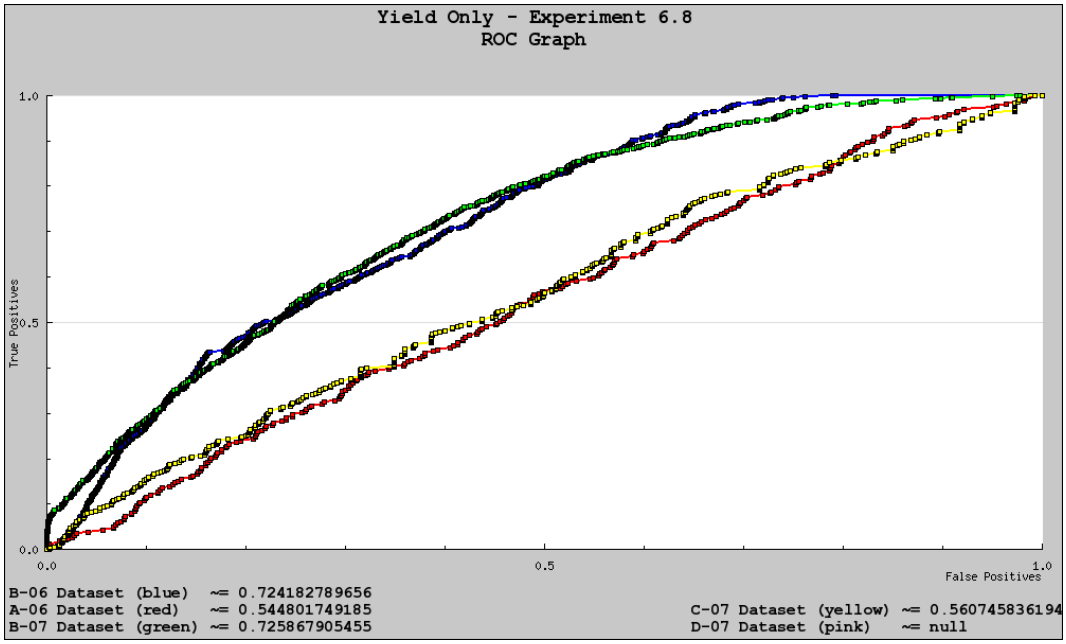


Figure 7.27: Yield Attribute ROC graph for Comparing Effectiveness of Attributes Experiment (6.7)

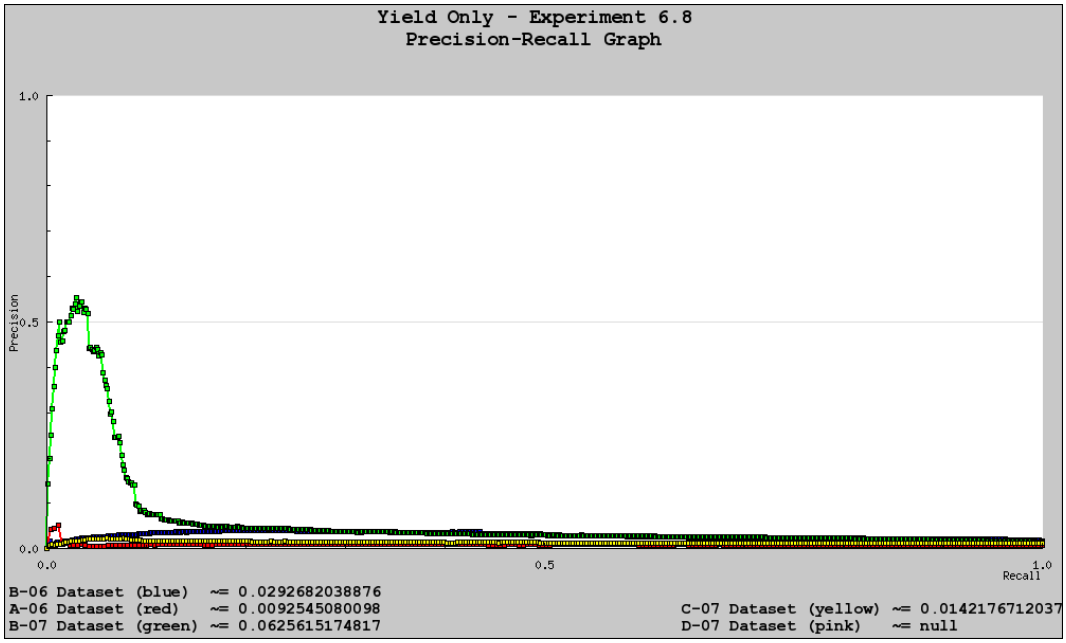


Figure 7.28: Yield Attribute Precision-Recall graph for Comparing Effectiveness of Attributes Experiment (6.7)

Attribute	C-07		D-07	
	ROC	PR	ROC	PR
Rank	**0.6829**	**0.0926**	**0.7832**	**0.0746**
Weight	n/a	n/a	0.7497	0.0317
Yield	0.5607	0.0142	n/a	n/a

Table 7.9: Summarised Results for Comparing Effectiveness of Attributes Experiment (6.7) - C-07 and D-07 herds

Two graphs are given for each attribute, one showing ROC, the other showing Precision-Recall curves. All graphs are for Bagged J48 trees. The reason for the null values in the graphs is that C-07 and D-07 do not have weight and yield values respectively, thus no results are available.

It can be clearly seen from Figures 7.23 through to 7.28 that the Rank attributes are what put the B-06 herd so far ahead of the others when comparing Precision-Recall values. These results seem to show that with the Weight or Yield attributes only the lead that the B-06 herd has over the other herds is significantly diminished.

More generally it seems that models based on rank are more useful than those on weight, weight is more useful than yield and yield based models perform poorly overall. The sole exception to this conclusion is the B-07 herd where results for ROC areas seem to be almost constant between all sets of attributes and for Precision-Recall the Yield experiment comes out the best! The unexpected bump in the B-07 herd Precision-Recall curve, seen in Figure 7.28, suggests the B-07 herd gains more from the yield attributes than the other herds.

7.8 Results of Application of Classifiers Designed to Target Skewed Data

This experiment was performed hoping to achieve the same successes on the LIC Data as were found on the various skewed datasets in Chapter 4 when learning a model using a classifier designed to work with skewed data. As the LIC data is heavily skewed it makes sense that the skewed data techniques should perform well when applied to the dairy herd data.

Under Bagging and Roughly Balanced Bagging were selected as the classifiers to apply due to their success in Chapter 4. A range of classifier parameters were selected to be used based on results of the Parameter Tuning Experiment (4.3). It was expected that two or three majority class examples per minority example would perform the best overall. All five herds worth of data and the daily herd adjustments were used in these experiments. These experiments predicted mating events. Standard Bagging is included in these results for comparison. J48 was used as the base classifier in all cases.

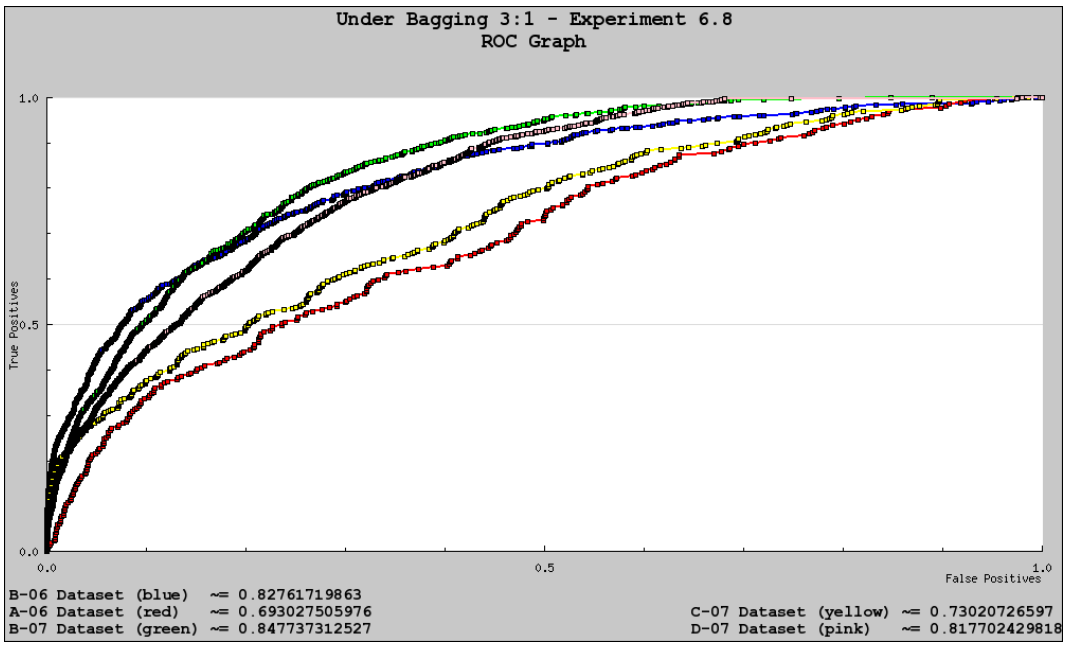


Figure 7.29: Under Bagging 3:1 Ratio ROC graph for Application of Classifiers Designed to Target Skewed Data Experiment (6.8)

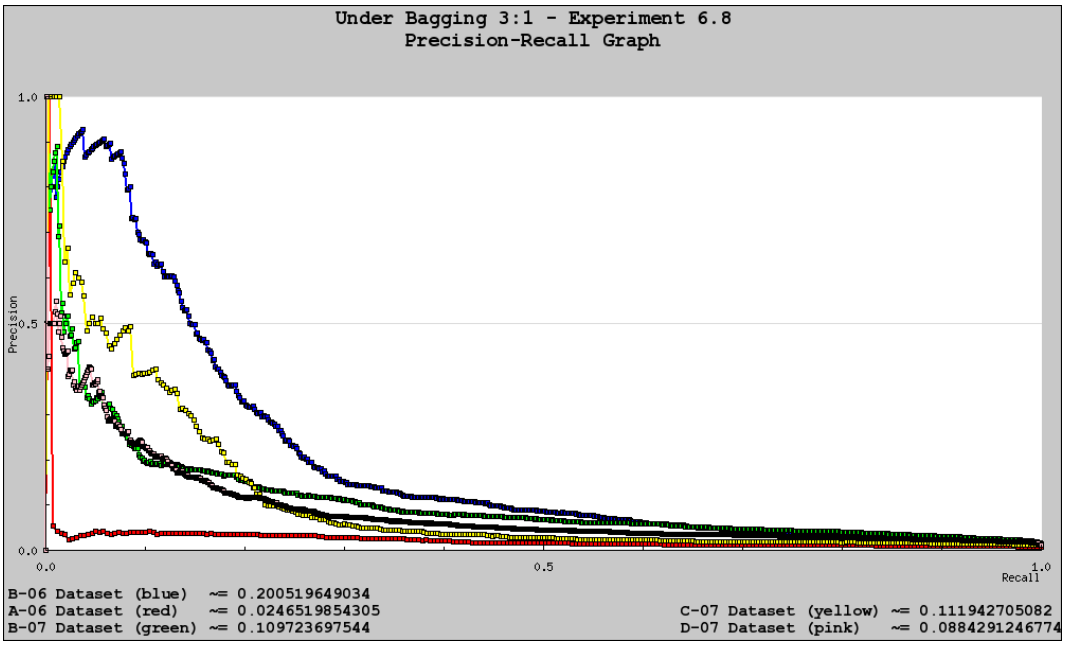


Figure 7.30: Under Bagging 3:1 Ratio Precision-Recall graph for Application of Classifiers Designed to Target Skewed Data Experiment(6.8)

Once again, both the ROC and Precision-Recall graphs are provided for the classifier examples selected. As the graph results were all very similar between parameter combinations only a single example of each skewed data classifier has been provided here. The 3:1 ratio classifiers for Under Bagging and Roughly Balanced Bagging were selected due to Under Bagging 3:1 performing well in the overall summary. As the experiment process is unaltered at this point, the results for Bagging Unpruned J48 shown in Figures 7.19 and 7.20 from the Removal of Standard Deviation Differences Experiment (6.6) apply to this experiment as well.

It can be observed in Tables 7.10 and 7.11 that there is no single parameter combination for the skewed data classifiers that excels over all the herds when it comes to Precision-Recall or ROC area. Although the highest values are marked it should be noted that all the skewed classifiers got results that were very close to the highest values for a given herd, with most within 0.01 of that maximum value.

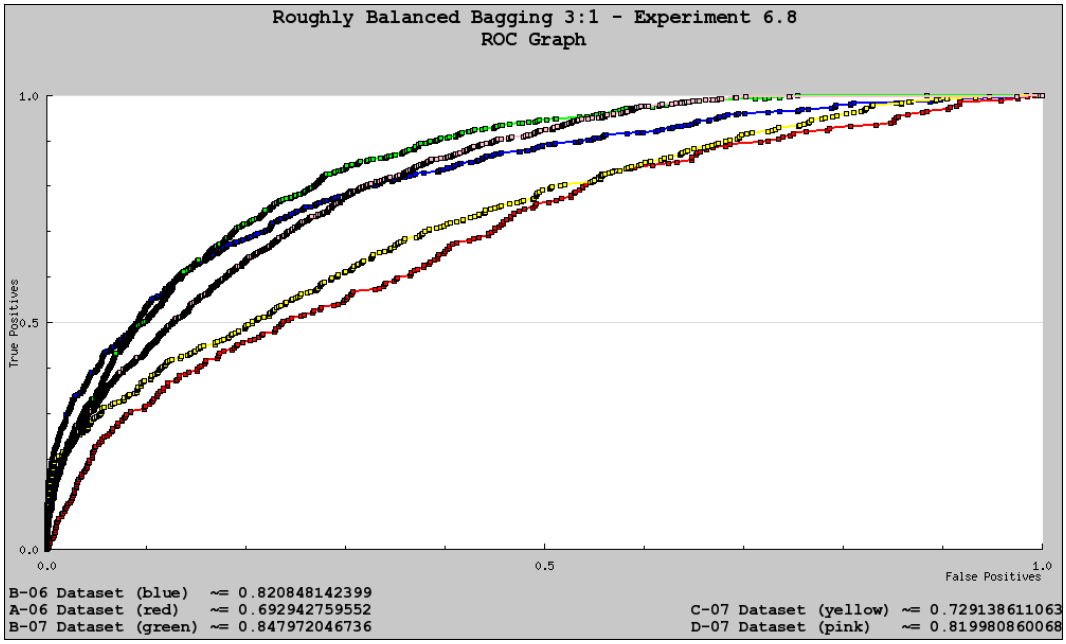


Figure 7.31: Roughly Balanced Bagging 3:1 Ratio ROC graph for Application of Classifiers Designed to Target Skewed Data Experiment (6.8)

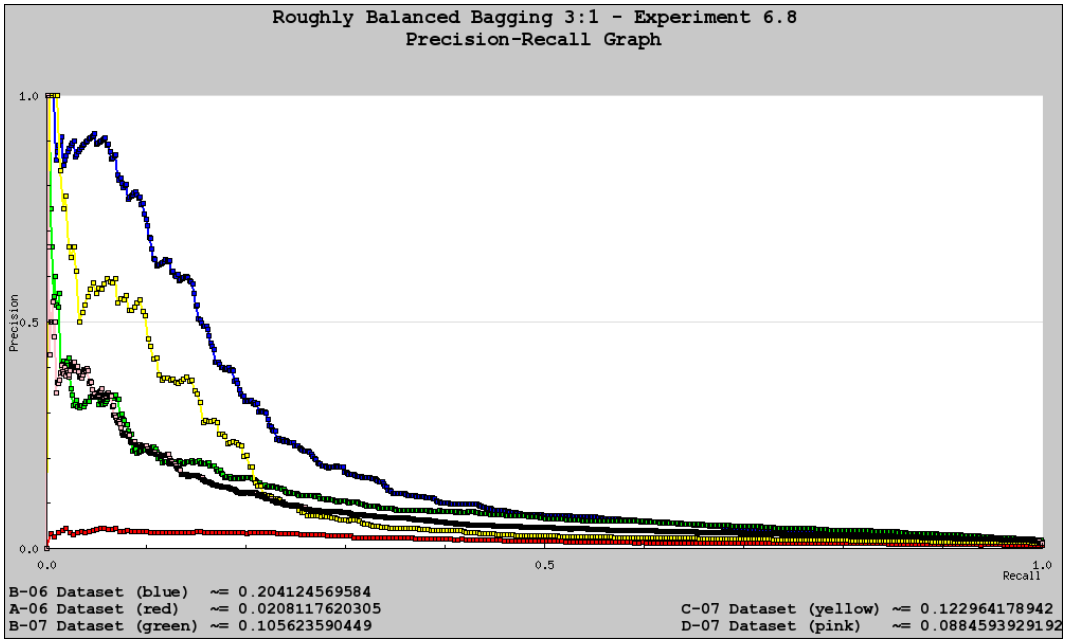


Figure 7.32: Roughly Balanced Bagging 3:1 Ratio Precision-Recall graph for Application of Classifiers Designed to Target Skewed Data Experiment (6.8)

Classifier	B-06		A-06		B-07	
	ROC	PR	ROC	PR	ROC	PR
Bagged Unpruned J48	0.8083	**0.2086**	0.6625	0.0176	0.8331	**0.1123**
Under Bagging 1:1	0.8247	0.1860	**0.7084**	0.0223	0.8379	0.0852
Roughly Balanced 1:1	0.8230	0.1980	0.7047	0.0233	0.8373	0.0881
Under Bagging 2:1	0.8257	0.1953	0.6971	0.0218	0.8452	0.1007
Roughly Balanced 2:1	0.8243	0.1934	0.6990	0.0223	0.8445	0.1011
Under Bagging 3:1	**0.8276**	0.2005	0.6930	**0.0247**	0.8477	0.1097
Roughly Balanced 3:1	0.8208	0.2041	0.6929	0.0208	**0.8480**	0.1056

Table 7.10: Summarised Results for Application of Classifiers Designed to Target Skewed Data Experiment (6.8) - A-06, B-06 and B-07 herds

Attribute	C-07		D-07	
	ROC	PR	ROC	PR
Bagged Unpruned J48	0.7115	0.1133	0.8069	0.0851
Under Bagging 1:1	**0.7434**	0.1142	0.8169	0.0814
Roughly Balanced 1:1	0.7373	0.1165	0.8168	0.0840
Under Bagging 2:1	0.7298	0.1148	0.8156	0.0850
Roughly Balanced 2:1	0.7301	0.1196	0.8191	0.0837
Under Bagging 3:1	0.7302	0.1119	0.8177	0.0884
Roughly Balanced 3:1	0.7291	**0.1230**	**0.8200**	**0.0885**

Table 7.11: Summarised Results for Application of Classifiers Designed to Target Skewed Data Experiment (6.8) - C-07 and D-07 herds

A rather interesting observation can be made when comparing the ROC areas for the skewed data classifiers to the standard Bagging classifier; all the skewed classifiers out perform standard Bagging in terms of ROC area. However, when comparing Precision-Recall areas, it is clear that standard Bagging is ahead. Bagging manages to get the top value for Precision-Recall on the B-06 and B-07 herds and is always very close to the skewed classifiers in terms of Precision-Recall area for the other herds.

7.9 Investigation into the Effectiveness of Classifiers on the B-06 Herd

With the B-06 herd outperforming the other herds by a large margin in all experiments, especially in terms of Precision-Recall area, it seemed important to conduct an investigation and find out exactly why classifiers could predict heat so well for this herd.

The first hint to what might explain the difference in results lay in the results for the Comparing Effectiveness of Attributes Experiment (6.7). The Rank attribute was clearly shown to be necessary to bring the Precision-Recall area for the B-06 herd above the values for the other herds.

Having identified the source of the improved results in the B-06 herd it was then a matter of inspecting the database tables for the rank information. The data rows were grouped by cow and sorted from earliest to latest milking date to put all milking periods in chronological order. It was immediately obvious that the source of the higher performance lay in what appeared to be some change in the milking procedure when cows were identified as ready to be inseminated.

The procedure, which appeared to be present in a large number of cases, seemed to involve cows being milked two or three hours later in the afternoon

of a day they were identified as being in heat. Additionally, the same cows would be milked about two or three hours later during the morning milking following the day they had been identified as being in heat. This information would have been seen by all classifiers in the form of large rank values as the attribute generation step would assume all of the milking had occurred as part of a single milking period. Thus, the vast majority of cows in the B-06 herd that had mating events would appear to have large afternoon rank values for the day it was recorded.

Typically having a powerful predictor like the milking times discovered here would be a positive thing in a machine learning problem. However, it was initially suspected, and then later confirmed (with expertise from LIC) that this milking procedure was a reaction by the farmer to identifying heat himself. As a result, it was not effective to use this information to identify heat in the first place as it would not exist!

Chapter 8

Conclusions

It was identified in Chapter 3 that Unskewed Sampling and SMOTE were both outperformed by the iteratively applied Bagging techniques of the standard Bagging, Roughly Balanced Bagging and Under Bagging classifiers. In the case of SMOTE this is even true while taking a similar time to train as the iterative classifiers. However, it is worth noting that there may be room for optimisations in the SMOTE code to improve performance and that it is possible SMOTE could outperform Bagging classifiers in some problem domain that was not explored as part of this investigation.

In contrast to the results for Unskewed Sampling and SMOTE were the results for the Bagging based skewed data approaches. The Roughly Balanced Bagging and Under Bagging techniques, both based on the Bagging iteratively applied classification technique, achieved relatively similar results to one another in the Chapter 3 experiments. What is most useful is that these classifiers can achieve, on skewed datasets, at least equal and often better performance than the standard Bagging classifier with a considerably lower training time. This lower training time is only true when parameters are chosen that will create a suitable ratio of majority to minority examples. Ratios of two or three majority examples for each minority example seemed to give the best results.

The data provided by LIC was skewed data; just like the datasets explored in Chapter 3. It seemed reasonable to use the Roughly Balanced Bagging and Under Bagging techniques that were successful on the skewed datasets used in Chapter 3 on the LIC data as well in hopes of getting similar successes. The Application of Classifiers Designed to Target Skewed Data Experiment (6.8) explored this idea. While there were positive results in terms of ROC areas when applying the skewed data classifiers, with the added Precision-Recall measure available on the LIC data it was shown that Roughly Balanced Bagging and Under Bagging did not make a significant improvement over standard Bagging when comparisons between Precision-Recall areas were made.

Throughout all of the experiments performed on the LIC data the B-06 herd outperformed all the other herds in terms of ROC area and, to an even greater degree, outperformed the other herds in terms of Precision-Recall area. It was disappointing to find, when the Investigation into the Effectiveness of Classifiers on the B-06 Herd (7.9) was completed, the reason the herd had been getting such positive results was because heat had already been identified by the farmer. Because the farmer made changes to the milking order of the cows, by milking the cows later, classifiers were able to easily predict which cows were in heat; this is not helpful for automating the prediction of heat though.

While it is certainly possible that other procedural differences could exist, similar to changes made to the B-06 herd where the milking order for cows identified as being in heat, the results of the Comparing Effectiveness of Attributes Experiment (6.7) showed how much each attribute source was contributing to a classification. With one exception, the B-07 herd, the attributes based on the rank attribute source appeared to be the most powerful when it came to predicting heat. It was also probable that the weight attribute source contributed more than the yield attribute source, but there is less data supporting this as complete weight data was not available for the C-07 herd and complete

yield information was not available for the D-07 herd.

Despite the continually refined experimental process for the LIC data, industry changing results were never achieved. Results looked promising in terms of ROC area at times, but when looking at the Precision-Recall curves it was clear there was little chance of predicting heat accurately enough to be useful. This does not mean the process refinement was not effective; it was simply not good enough to make the results usable in a real application. The reality may be that using rank, weight and yield information is simply not sufficient to predict when cows are in heat. LIC has made it clear there is still advancement occurring in the area of dairy farming automation and that as technologies are disseminated to farms it is likely more data will be recorded on a daily basis. Perhaps, as more information is made available for each milking, detection of heat by a human expert could be replaced by machine learning methods at some time in the future.

Another explanation, or at least one that may partially explain the disappointing results, is that the data provided by LIC, for whatever reason, had some obvious flaws. These flaws came to light when performing outlier detection and replacement and consisted of a large number of missing values and outlier data. While the steps performed to remove outliers in Section 5.4 made it possible to use machine learning techniques on the data, it is certainly possible that attributes that would have been strong predictors of heat were compromised by the state of the data provided. There are many possible reasons for the data being incomplete or containing outlier values; possible explanations include: inaccurate data being recorded due to faulty equipment, a human error such as forgetting to activate a piece of recording equipment or even, especially in the case of scales, more than one animal using the equipment at the same time.

While it is true that this work has shown Machine Learning, in the forms it was applied, to be insufficient to automate the detection of oestrus in dairy cows, it is unreasonable to say the work added nothing to the discipline. The first, and perhaps the most significant fact to keep in mind, is that there was only a very small amount of herd data available and over a very limited number of attribute sources. An investigation into automating the detection of oestrus could still be very successful on other dairy cows; perhaps in a more controlled environment. Also, if it were possible to get additional data values that were recorded on a daily basis there would be room for further Machine Learning applications. Additional values, that could be measured daily, might include milk solid levels or the temperature of milk at milking time.

Aside from the dairy specific section of this investigation, there is much room for general work in the area of skewed data problems. It would be very beneficial to the discipline to have a broad comparison of algorithms in the area of skewed problems; a benchmarking experiment scaled up considerably from the skewed data investigation undertaken as part of this work. It would be important to consider many algorithms, including the successful Roughly Balanced Bagging and Under Bagging (developed during this investigation) classifiers as part of such a comparison.

Given the work conducted by Davis and Goadrich[22] supports the use of Precision-Recall curves on skewed data, and that this investigation has shown Precision-Recall curve areas to give a interesting additional measure for comparing classifiers, it would be a beneficial to see the use of Precision-Recall curves increase within Machine Learning. Additionally, if a large scale skewed data classifier comparison investigation was performed, it would be a positive addition to such an investigation to show results as Precision-Recall curves/areas in addition to ROC curves/areas.

References

[1] L. Saitta and F. Neri. Learning in the Real World. *Machine Learning*, 30(2):133–163, 1998.

[2] New Zealand Ministry of Foreign Affairs and Trade. *New Zealand External Trade Statistics*. Statistics New Zealand, June 2007.

[3] Livestock Improvement Corporation Limited. *New Zealand Dairy Statistics 2007-2008*. Livestock Improvement Corporation Limited, 2008.

[4] R.H. Foote. Estrus Detection and Estrus Detection Aids. *Journal of Dairy Science*, 58(2):248, 1975.

[5] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[6] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[7] George H.J. and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.

[8] M. Bayes. An Essay towards Solving a Problem in the Doctrine of Chances. Communicated by Mr. Price, in a letter to John Canton. *Philosophical Transactions*, 53:370–418, 1763.

[9] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–283, 1989.

[10] P. Langley, W. Iba, and K. Thompson. An Analysis of Bayesian Classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[11] G.I. Webb, J.R. Boughton, and Z. Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5–24, 2005.

[12] S. le Cessie and J.C. van Houwelingen. Ridge Estimators in Logistic Regression. *Applied Statistics*, 41(1):191–201, 1992.

[13] P.F. Verhulst. Notice sur la loi que la population suit dans son accroissement. *Correspondance Mathématique et Physique*, 10:113–121, 1838.

[14] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. *Computational Learning Theory*, pages 144–152, 1992.

[15] J.C. Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. *Advances in Kernel Methods-Support Vector Learning*, pages 185–208, 1999.

[16] M.D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, July 2003.

[17] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[19] Y. Freund and R.E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[20] K.A. Spackman. Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 160–163. Morgan Kaufmann, 1989.

[21] A.P. Bradley. The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[22] J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240. Association for Computing Machinery, 2006.

[23] R.J. McQueen, S.R. Garner, C.G. Nevill-Manning, and I.H. Witten. Applying Machine Learning to Agricultural Data. *Computers and Electronics in Agriculture*, 12(4):275–293, 1995.

[24] S.R. Garner, S.J. Cunningham, G. Holmes, C.G. Nevill-Manning, and I.H. Witten. Applying a Machine Learning Workbench: Experience with Agricultural Databases. In *Practice Workshop of the 12th International Conference on Machine Learning*, 1995.

[25] R.S. Mitchell, R.A. Sherlock, and L.A. Smith. An Investigation Into the use of Machine Learning for Determining Oestrus in Cows. *Computers and Electronics in Agriculture*, 15(3):195–213, 1996.

[26] N. Japkowicz. The Class Imbalance Problem: Significance and Strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence*, volume 1, pages 111–117, 2000.

[27] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing Misclassification Costs. In *Proceedings of the 11th International Conference on Machine Learning*, pages 217–225, 1994.

[28] P. Domingos. MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164. Association for Computing Machinery, 1999.

[29] M. Kubat and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *Proceedings of the 14th International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

[30] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(3):321–357, 2002.

[31] S. Hido and H. Kashima. Roughly Balanced Bagging for Imbalanced Data. In *Proceedings of the SIAM International Conference on Data Mining*, pages 143–152, 2008.

[32] UCI Machine Learning Data Repository. http://ics.uci.edu/~mlearn/MLRepository.html.

[33] IEEE International Conference on Data Mining. http://cs.uu.nl/groups/ADA/icdm08cup/data.html.

[34] EMS SQL Manager for InterBase/Firebird. http://sqlmanager.net/en/products/ibfb/manager.

[35] D. Dougherty and A. Robbins. *sed & awk*. O'Reilly Media, 2 edition, 1997.