Chapter 5
Feature Selection for Large Datasets

> .. the object of data analysis is not to model the fleeting random patterns of the moment, but to model the underlying structures which give rise to consistent and replicable patterns. ..' (Hand, 1998)

It was stated in chapters 2 and 3 that the selection of a good subset of predictive features results in the reduction of the variance component of a predictive model. To the author's knowledge, there are very few reported studies on research that addresses feature selection in the presence of large datasets. One such study has been reported by Liu and Setiono (1998a, 1998b). Research has been reported on validation of class-feature correlation coefficients using fake variables (Stoppiglia et al, 2003; Bi et al, 2003). Since this method of validation has only been applied to small datasets, it is useful to establish whether the use of fake variables for validation can be effectively applied to feature selection from large datasets. It was also argued in chapter 3 that algorithms that conduct feature subset search should use clearly specified definitions of feature relevance.

The purpose of this chapter is to report the experimental results of the study of feature subset selection in the presence of sampling from large datasets. Experimental results are reported on studies that were conducted on two correlation measures, two validation methods for correlations, and three algorithms for feature subset selection. Hand (1998) has made the insightful observation about data analysis as quoted at the beginning of this chapter, and it is in the spirit of this observation that experiments for this chapter were designed and conducted. It is argued in this chapter that statistical methods can be used to make inferences on the expected values of the feature correlations for large datasets when many samples are used. The use of many samples for correlation measurement should lead to better decisions for feature selection since the correlation values obtained are more reliable. It is further argued that features that are selected when domain-specific definitions of feature relevance are incorporated into the feature selection procedures are the best features for the prediction task at hand. In the context of processing large datasets in data mining, the following research questions are answered in this chapter:

1. How can class-feature correlations be measured in order to produce a reliable ranking of features for a dataset?
2. What methods of validation for feature correlations result in reliable feature selection?
3. How can domain-specific definitions of feature relevance be incorporated into feature selection procedures?

The rest of this chapter is organised as follows. Section 5.1 gives a summary of the feature selection problem. Section 5.2 presents the different approaches to feature selection that were studied. Empirical studies of feature ranking, feature subset search and predictive performance of selected feature subsets are respectively discussed in sections 5.3, 5.4 and 5.5. The discussion of the experimental results and conclusions are respectively given in sections 5.6 and 5.7.

5.1 The feature selection problem revisited

It was stated in chapter 3 that the initial selection of features is typically done by a domain expert, based on the data mining task at hand. Subsequent to this, a process of selecting the most relevant features and eliminating redundant features must be conducted. It is this process which is addressed in this chapter. Further, Guyon and Elisseeff (2003) have observed that there is not just the one method of feature selection that suits all datasets, all algorithms and all data mining tasks. With Guyon and Elisseeff's (2003) observations in mind, the methods discussed in this chapter were directed at large datasets of moderately high dimensionality.

It was also stated in chapter 3 that filtering methods are preferred to wrapper methods for data mining for reasons of efficiency. The measurement of class-feature and feature-feature correlations is at the core of many filtering methods for feature selection. In the experiments reported for this chapter, many of the correlation measures commonly used in filtering methods for feature selection (Yu & Liu, 2004; Hall, 1999, 2000) were adopted to establish feature relevance and redundancy in the presence of sampling. For feature relevance, these measures are based on the strength of the correlation between a feature and the class variable. For redundancy, the measures are based on the strength of the correlations between the features. The correlation measures were presented in chapter 3. While studies of feature

selection most commonly use one sample (i.e. the whole dataset) to establish the feature correlation values, the studies reported in this chapter were directed at using many samples to establish the correlation values for the dataset features.

5.2 Alternative approaches to feature selection for large datasets

When large datasets are available the question arises as to whether relevant features should be selected based on the whole dataset, one sample from the dataset, or several samples taken from the dataset. When a dataset is very large, it is not feasible to compute correlation values using all the available data. If only one sample is taken there is a great risk of making the wrong decisions about which features are the most relevant. Based on Hand's (1998) observations as quoted at the beginning of this chapter, the purpose of feature selection should not be to identify the best features for the specific training sample that has been chosen (or happens to be available) for model creation, but rather to identify the best features for model creation regardless of the specific training sample that is chosen. In other words, the objectives of feature selection should be directed at the data generating process and not solely at the data sample that happens to be available. In attempting to answer the question:

How can class-feature correlations be measured in order to produce a reliable ranking of features for a dataset?

the author hypothesised that the use of many samples to measure class-feature and feature-feature correlations should provide reliable estimates of these correlations. In order to provide evidence to support this hypothesis, the following alternatives were considered and studied:

Alternative 1

Use one small sample (e.g. 100 or 500 instances) to measure correlations and select the features, and assume that these will be the relevant features for the instance space and prediction task regardless of the specific training sample used for classification model construction. The motivation for this alternative is that, first of all, the computation of correlation values from small samples is faster than for large samples. Secondly, statistical theory points to the fact that relationships that appear

to be strong in small samples are generally stronger than relationships which only appear in large samples of data. The foregoing observation can be used to argue that features that have strong correlations in small samples are the strongest predictors and will have globally predictive power. The term *globally predictive* is used to mean that a feature will have predictive power in all regions of the instance space.

Alternative 2

Use one large sample (e.g. 1000 instances) to measure correlations and select the features and assume that these will be relevant features for the instance space and prediction task regardless of the specific training sample used for classification model construction. The rationale here is that those features which are not strongly predictive may be eliminated when a small sample is used. The use of a large sample increases the chances of identifying more features for the prediction task.

Alternative 3

Use many small samples of one size to select the features and assume that these will be relevant features for the instance space and prediction task regardless of the specific training sample used for classification model construction. The rationale here is the same as for alternative 1. Additionally, taking the mean values of the correlations measured on many samples, and using statistical inference to select features is more reliable than the use of a single sample correlation.

Alternative 4

Use many large samples of one size to select the features and assume that these will be relevant for the instance space and prediction task regardless of the specific training sample used for classification model construction. Again, taking the mean values of the correlations measured on many samples, and using statistical inference to select features is more reliable than using a single sample correlation.

The next section provides the experimental results for the investigation of the above four alternatives.

5.3 Empirical study of feature ranking methods for large datasets

For feature ranking the selection criteria are applied to each feature, without any consideration of the contribution of the other features to the prediction performance. The ranking criteria reported in this section are based on feature correlation measures. The results of the experiments that were conducted on feature selection based on *pure ranking* of features are reported in this section. The experimental procedures that were used are given in section 5.3.1. A comparison of Pearson's and Kendall's correlation measures is given in section 5.3.2. Sections 5.3.3 and 5.3.4 respectively provide the experimental results and discussions for feature ranking based on a single samples and feature ranking based on many samples.

5.3.1 Experimental procedure for the study of feature ranking

The datasets presented in chapter 4 were used for the experiments. The sequential random sampling method (SRS), described in chapter 4, was used to obtain random samples. Probes (fake variables) with values drawn from both Gaussian and uniform distributions were used. Probes may be added to the datasets prior to taking samples for feature selection. However, adding probes to a very large dataset is a computationally lengthy and unnecessary process. The probes can be added during or after the sampling step. The generation of pseudo-random numbers is a process of sampling from the specified distribution (Thomas et al, 2007). The sampling approach used for the experiments was to first take samples from the large dataset and then sample from the chosen probability distributions for the probes (fake variables). Three types of probes were used with values drawn from a Gaussian distribution, a uniform distribution, and uniform binary distribution. For the Gaussian probe, the Marsaglia-Bray algorithm was used to generate the pseudo-random numbers (Thomas et al, 2007). For the uniform probes, the Borland C++ function for generating random numbers was used. The datasets that were used in the experiments for this thesis contain quantitative (discrete and continuous), and qualitative (nominal and ordinal) features. The forest cover type and KDD Cup 1999 also contain binary (quantitative discrete) features. Furthermore, even though the correlation values (for quantitative features) and symmetrical uncertainty (SU) coefficient values (for qualitative features) are comparable, the functions used to

compute the correlations are different. The functions for computing Pearson's correlation Kendall's correlation and SU coefficients were presented in chapter 3. It is statistically meaningful to compare a true binary feature with a fake binary feature and a true qualitative feature with a fake qualitative variable. For this reason one Gaussian and two uniform probes were used. Table 5.1 shows the characteristics of the probes used for the datasets.

Table 5.1 Characteristics of the probes for the datasets

Probe name	Value range	Description
Probe1GaussCont	0- 999	Gaussian distribution with mean = 500, stdev = 100
Probe2UniformCont	0-999	uniform distribution
Probe3UniformBin	0,1	uniform distribution with binary values

Class-feature and feature-feature correlation coefficients were computed from samples drawn from a large dataset, for both the true features and the probes (fake variables). Two methods were studied for computing correlations: Pearson's correlation coefficient and Kendall's *tau* correlation coefficient. Two criteria were studied for feature ranking selection: statistical significance with the t-test on mean values for correlations and symmetrical uncertainty (SU) coefficients, and statistical significance based on probes. Two algorithms, See5 for classification trees and Nearest Neighbours (5NN) were used for comparison. It should be noted that these two classification algorithms differ significantly in their treatment of predictive features during model construction. The 5NN algorithm does not have the ability to rank features or select relevant features, while the classification tree algorithm performs an implicit ranking of features and also performs tree pruning to ensure that only statistically significant information provided by the features is used in the final classification tree.

5.3.2 Comparison of Pearson's and Kendall's correlation measures

For the comparison of Pearson's and Kendall's correlation coefficients, experiments were conducted to compare the mean values of the class-feature correlations using 10 samples to compute each mean value. Correlation values for the top 10 variables are shown in table 5.2. Table 5.2 shows the class-feature correlation values for the three datasets: Forest cover type, KDD Cup 1999 and Abalone3C. For each dataset, the top 10 features as ranked by Kendall's *tau* are shown. It should be noted that the

forest cover type dataset has 54 features, the KDD Cup 1999 dataset has 41 features, and the abalone3C dataset has eight features. Only the top 10 features for the forest cover type and KDD Cup 1999 datasets are shown in table 5.2 for purposes of concise presentation, and for illustration of the differences between the Pearson's *r* and Kendall's *tau* coefficients.

Table 5.2 Comparison of mean values for Kendall's tau and Pearson's r

Dataset	Top 10 features as ranked by Kendall's tau	Mean values for correlation coefficients for 10 test samples			
		Sample size = 1000		Sample size = 500	
		Kendall's tau	Corresponding Pearson's r	Kendall's tau	Corresponding Pearson's r
Forest cover type	WildernessArea4	0.86	0.22	0.81	0.22
	SoilType12	0.70	0.14	0.72	0.16
	SoilType1	0.69	0.08	0.44	0.06
	SoilType38	0.68	0.12	0.60	0.12
	SoilType39	0.68	0.11	0.58	0.11
	SoilType2	0.64	0.07	0.58	0.09
	SoilType4	0.64	0.10	0.57	0.11
	SoilType6	0.60	0.08	0.56	0.09
	SoilType22	0.59	0.14	0.57	0.14
	SoilType10	0.58	0.13	0.47	0.11
KDDCup99	SerrorRate	0.92	0.51	0.87	0.45
	NumCompromised	0.92	0.23	0.85	0.26
	SrvSerrorRate	0.91	0.50	0.90	0.43
	WrongFragment	0.90	0.21	0.81	0.18
	DstHostSrvSerrorRate	0.85	0.50	0.83	0.43
	DstHostSrvRerrorRate	0.85	0.34	0.76	0.27
	SrvRerrorRate	0.85	0.35	0.80	0.28
	Hot	0.84	0.11	0.78	0.14
	DstHostSerrorRate	0.84	0.51	0.81	0.44
	RerrorRate	0.82	0.34	0.76	0.27
Abalone 3C (all features)	Diameter	0.50	0.41	0.50	0.41
	Shellweight	0.52	0.40	0.53	0.40
	Height	0.51	0.37	0.52	0.39
	WholeWeight	0.49	0.38	0.50	0.38
	VisceraWeight	0.49	0.38	0.49	0.38
	ShuckledWeight	0.45	0.34	0.45	0.34
	Length	0.17	0.14	0.18	0.15
	Gender (qualitative)	0.12	0.12	0.13	0.13

The mean correlation values shown in table 5.2 for sample sizes of 500 and 1000 indicate that for the forest cover type and KDD Cup 1999 datasets the class-feature correlations as measured by Kendall's *tau* are generally much larger than the class-feature correlations measured using Pearson's correlation coefficient. Secondly,

even among the top 10 features out of 54 features for forest cover type, the two features *SoilType1* and *SoilType2* have strong class-feature correlations based on Kendall's *tau* but have insignificant correlations based on Pearson's *r*. A feature ranking method based on Pearson's *r* would eliminate the features *SoilType1* and *SoilType2*. Thirdly, for both forest cover type and KDD Cup 1999 the feature rankings based on Kendall's *tau* are different from the rankings based on Pearson's *r*.

The reader will recall from chapter 3 that the point was made that Wilcox (2001) has cautioned against the interpretation of Pearson's *r* for measuring correlations when there is no guarantee that the correlation between two variables is linear, and when outliers have not been given special treatment. A small Pearson's correlation coefficient between two variables does not necessarily mean that the two variables are not strongly correlated. It could be the case that the correlation is not linear or the correlation is masked by the presence of outliers in the data. On the other hand, Kendall's *tau* is a robust measure of correlation which will provide reliable correlation values even when the correlation is not linear and even when outliers are present in the data (Wilcox, 2001). For the Abalone3C dataset, the results of table 5.2 indicate that the differences between the class-feature correlations measured with Kendall's *tau* and Pearson's *r* are marginal and the feature rankings based on both correlation measures are nearly the same. Based on Wilcox's (2001) observations, it can be deduced that Pearson's *r* is a suitable correlation measure for the Abalone3C dataset because there are no outliers in the data and the predictive features are linearly correlated to the class variable. It can be deduced from table 5.2 that Pearson's *r* is not a suitable correlation measure for the forest cover type and KDD Cup 1999 datasets because the datasets either have outliers or the correlations between the features and the class variables are non-linear.

Table 5.3 and tables D.1, D.4 and D.7 of appendix D respectively give the number of features that would be selected for the forest cover type, KDD Cup 1999 and Abalone 3C datasets based on the Students t-test of means. The test was conducted to determine the features whose mean class-feature correlation coefficient is greater than or equal to *0.1*. The reader will recall from the discussion of chapter 3 that Cohen (1998) has advised that a correlation value with a magnitude in the interval [0, 0.1) has no practical significance in any domain for data analysis and a correlation value with a magnitude in the interval [0.1, 1.0] may have practical significance. For the forest cover type dataset only 6 out of 54 features would be selected based on

Pearson's *r*. Based on the foregoing observations all subsequent experiments for feature selection were based on Kendall's *tau* as the correlation measure.

Table 5.3 Comparison of the number of selected features for Kendall's tau and Pearson's r

Dataset (no. of features)	Sample size	Number of features with a mean $corr_{cf}$ or mean *SU* coefficient that is significant ($corr_{cf}$ >= 0.1, significance level 0.01)	
		Kendall's *tau*	Pearson's *r*
Forest cover type (54)	500	35	6
	1000	38	6
KDDCup99 (41)	500	36	26
	1000	30	21
Abalone (3 class) (8)	500	6	5
	1000	7	5

5.3.3 Feature ranking based on a single sample

The problems and consequences of using a single small or large sample are investigated and made explicit in this section. Ten small samples, 10 medium samples, and 10 large samples were taken from the forest cover type dataset using sequential random sampling (SRS). Table 5.4 shows the number of features selected for each sample by the Gaussian probe and Z-test based on class-feature correlations measured using Kendall's *tau*. For the probes, the selection criterion is a class-feature correlation coefficient greater than that of the Gaussian probe. The number of features selected by the uniform probe and uniform-binary probe are given in tables D.1, D.4 and D.7 of appendix D. Since only one correlation value is available for each predictive feature for these experiments, the Z-test for a single correlation value was used to test the hypothesis that the class-feature correlation value is greater or equal to 0.1, that is, features that have a correlation value which is of practical significance (Cohen, 1988). The Z-test for a single correlation measurement was discussed in chapter 3. The first problem that can be deduced from table 5.4 is that sample sizes of 100 result in very few features being selected. The second problem is that the number of selected features varies from sample to sample. Smyth (2001) has argued that if a single sample is used to measure correlations between variables, then features may be lucky (or unlucky) in the sample and get selected (or eliminated) based on the single correlation measurement.

It could be argued that as sample sizes get larger the variability in the measured correlation coefficient will decrease. However, even for sample sizes of 1000 which is large for statistical hypothesis testing, one can see from table 5.4 that the variability in the number of features selected is still high. A second problem that arises when a

single sample is used for feature selection is illustrated in table 5.5. Table 5.5 shows the class-feature correlation values for four of the features in the KDD Cup 1999 dataset, as measured using Kendall's *tau* with samples of size 1000. It can be deduced from table 5.5 that a feature (e.g. *NumFailedLogins*) can have no correlation, small correlation, medium correlation, or high correlation with the class variable depending on the sample that is used, even when the sample size for correlation measurements is large. Alternatives 1 and 2 as stated in section 5.2 were discarded due to the three problems discussed above and no further studies of correlation measurement with small sample sizes (size = 100) were conducted.

Table 5.4: Number of selected features based on single samples for forest cover type

Sample ID	**Number of relevant features** with a significant class-feature Kendall's tau correlation selected by the Gaussian probe and Z-test for forest cover type					
	size = 100		size = 500		size = 1000	
	Gaussian Probe	Z-test $\|corr_{cf}\| >= 0.1$	Gaussian Probe	Z-test $\|corr_{cf}\| >= 0.1$	Gaussian Probe	Z-test $\|corr_{cf}\| >= 0.1$
S1	35	22	46	31	46	35
S2	34	14	46	37	43	40
S3	30	13	46	34	43	35
S4	35	18	47	36	48	39
S5	39	16	46	34	49	41
S6	30	23	42	32	49	37
S7	34	15	42	32	48	38
S8	35	21	47	33	46	34
S9	34	16	46	34	48	37
S10	35	19	39	32	49	41

Table 5.5: Kendall's correlations for four features for KDD Cup 1999

Feature	Sample ID										$corr_{cf}$	
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Mean	Stdev
NumFailedLogins	0.33	0.58	0	0	0.58	0.39	0	0.33	0.49	0.33	0.30	0.23
NumShells	0.21	0.21	0.42	0	0.34	0.32	0	0	0.2	0.35	0.20	0.16
NumAccessFiles	0.35	0	0	0.33	0.43	0.22	0	0	0	0.44	0.18	0.20
SUAttempted	0	0.21	0	0	0	0	0	0	0	0	0.02	0.07

5.3.4 Feature ranking based on many samples

The rationale behind using many samples is that the use of one sample will lead to misleading conclusions as demonstrated above in section 5.3.3. Taking the mean over the correlation values for many samples should provide a more reliable estimate of the correlation values. Smyth (2001) has argued that a feature may be highly correlated with the class for a given sample, simply because it is lucky in that particular sample. In fact, the results of section 5.3.3 have illustrated this point precisely. The use of many samples for correlation measurement also enables the

validation of selected feature subsets using more robust (less prone to error) statistical methods. For 4 datasets, 10 medium sized samples (size = 500) and 10 large samples (size=1000) were used to compute the Kendall's *tau* and the *SU* coefficient for the class-feature associations. Two criteria were used for feature selection. The first criterion was to select features based on the confidence interval of the mean correlation value of the probe. If a feature has a confidence interval whose lower and upper values are both greater than the lower and upper values of the confidence interval for the probe then the feature is selected, otherwise it is rejected. The second criterion was to use Student's t-test on the mean value of Kendall's *tau* or *SU* coefficient at the 0.01 significance level.

Table 5.6: Number of selected features based on 10 samples

Dataset (no. of features)	Sample size	Number of relevant features with mean $corr_{cf}$ (Kendall's tau) or mean *SU* that is statistically significant. Number of samples = 10			
		Selection based on probes			t-test for ($corr_{cf}$>=0.1 or SU>=0.1 at the 0.01 level)
		Probe1 (Gaussian)	Probe2 (uniform-cont)	Probe3 (uniform-bin)	
Forest cover (54)	500	47	47	44	35
	1000	49	48	47	38
KDDCup99 (41)	500	36	36	36	34
	1000	36	36	35	30
Abalone (8)	500	8	8	8	6
	1000	8	8	8	7
Mushroom(22)	500	21	15	14	2

Table 5.6 shows the results for the number of selected features based on two criteria. The two uniform probes selected approximately the same number of features. The Gaussian probe selected approximately the same number of features as the uniform probes, except in the case of the mushroom dataset. The t-test is very strict as it selects the smallest number of features. The details of the features selected by the Gaussian probe for the forest cover type and KDD Cup 1999 are given in tables D.2 and D.5 of appendix D.

For the experiments of this section many samples were used to measure class-feature correlations and to conduct validation for the selected features using probes and the t-test for mean correlation values. The experimental results demonstrated that for medium sized samples (size = 500) and large sized samples (size = 1000) each validation method selects nearly the same number of features. However, different validation methods select different numbers of features. The Gaussian

probe is the least strict of all the methods as it generally selects more features. The t-test is the most strict as it generally selects the smallest number of features.

Based on the results of table 5.6, alternatives 3 and 4 as stated in section 5.2 provided useful options for correlation measurement, feature ranking and validation. For validation based on probes the variability in the number of selected features is low for both medium size (500) and large size (1000) samples even though the Gaussian probe does not work well for the mushroom dataset (all features are qualitative). Performance of the feature subset search algorithms based on the features selected in this section as inputs, are discussed in the next section.

5.4 Empirical study of feature subset search

Feature subset search is the process of searching for an optimal subset of features based on specified criteria. A common criterion is to select that subset of features (from a set of identified relevant features) that maximises relevance and minimises redundancy in the selected subset. Feature subset search methods and examples of the merit measures that are employed in heuristic search for feature subsets were discussed in detail in chapter 3. The experiments reported in this section are for feature subset selection using forward search. Forward search algorithms that employ the correlation-based feature selection (CFS) merit measure (Hall, 1999) and differential prioritisation (DP) measures (Ooi et al, 2007) were implemented and tested using the features selected in the last section as inputs. Section 5.4.1 provides a discussion and analysis of the implementation of feature relevance and redundancy definitions by the CFS (Hall, 1999) and DP (Ooi et al, 2007) search procedures. The weaknesses of the merit measures employed by the CFS and DP search procedures are made explicit. A new algorithm for feature subset search is proposed in section 5.4.2 and the algorithm's feature selection performance is compared to the CFS and differential prioritisation methods.

5.4.1 Implementation of feature relevance and redundancy definitions

A good feature ranking method should be followed by a good search procedure. A good feature subset search procedure should not have a *search bias* which forces it to prefer an irrelevant feature to a relevant one.

When the *search bias* is based on precise and domain-specific definitions of *weak*, *medium*, and *strong* feature correlations then the selected feature subset should be the best for that application domain (Lutu & Engelbrecht, 2010). If fake variables are included in the initial feature set, then they should only be used to indicate when the search should stop. In other words, if the search procedure finds that the best feature to select at a given point is a fake variable, then the search procedure should terminate. Possible terminating criteria in the presence of fake variables (probes) should then be: (1) *stop when a pre-specified number of features have been selected* or (2) *stop when a probe is encountered as the next best choice*.

Definitions of feature relevance (Blum & Langley, 1997) and feature redundancy (Koller & Sahami, 1996) were given in chapter 3. It was also stated in chapter 3 that many implementations of feature selection implement the meanings of relevance and redundancy using the level of class-feature and feature-feature correlations. For feature selection implementations it is generally accepted that a relevant feature is one which is *highly correlated* with the class variable and a redundant feature is one that is *highly correlated* with other features (Ooi et al, 2007; Yu & Liu, 2004; Hall, 1999, 2000). Table 5.7 provides a summary of common interpretations of levels of class-feature and feature-feature correlations for purposes of identifying relevant and redundant features. One problem with heuristic procedures for feature subset search, for example DP (Ooi et al, 2007) and CFS (Hall, 1999, 2000) is that the merit measures they use do not have sufficient precision to distinguish between *high correlation* as opposed to *not-high correlation*. It is demonstrated later in this section that there are several situations where the CFS search procedure (Hall,1999, 2000) and DP search procedure (Ooi et al,2007) prefer features with very low feature-feature correlations at the cost of eliminating features with high class-feature correlations.

Table 5.7: Interpretation of levels of feature correlations for heuristic search

Situation	class-feature correlation for feature *f*	mean feature-feature correlation of selected features if *f* is added to selected features	Interpretation according to the literature
s1	not high	not high	*f* is irrelevant
s2	not high	High	*f* is redundant
s3	High	not high	*f* is relevant
s4	High	High	*f* is redundant

Experiments for feature subset selection were conducted on the forest cover type and KDD Cup 1999 datasets since these are large datasets with large numbers of features as commonly encountered in predictive data mining (Hand et al, 2001; Hand, 1998). The purpose of the experiments was to establish the behaviour of the CFS and differential prioritization algorithms for feature subset search. Table 5.8 shows a partial trace of the computations of the CFS search procedure for the datasets. For one iteration of the CFS algorithm (column 5), the CFS algorithm selects the best feature (column 2) based on the value of the CFS merit measure (column 6). The CFS merit measure (discussed in chapter 3) is computed using the mean values of the class-feature and feature-feature correlations for the candidate feature subsets. For each iteration, columns 3 and 4 of table 5.8 show the value of the class-feature correlation for the selected feature and total feature-feature correlation for the subset of selected features.

For the situations depicted in table 5.7, when making a choice between a feature whose situation is *s1* and one whose situation is *s3*, CFS chooses the situation *s1* feature. For the forest cover type features, at iteration number 26, it would be preferable to choose one of *SoilType13* or *SoilType39* instead of the binary-valued probe since each of these features has a high class-feature correlation and its selection would result in a low level of feature-feature correlation for the selected features. At iteration 39 it would be better to choose *SoilType13* or *SoilType39* instead of *SoilType 25.* Similarly, at iteration number 21 for the KDD Cup 1999 dataset, it should be preferable to choose one of *DstHostSrvSerrorRate, DstHostSrvRerrorRate* or *Count*, instead of the Gaussian probe for the same reasons as stated above. The *search bias* of the CFS search procedure forces it to choose the Gaussian probe instead, since CFS does not have sufficient information to make the distinctions that are made in table 5.7.

Table 5.8: Trace of the CFS search procedure for the forest cover type and KDD Cup 1999

Dataset	Selected feature F	class-feature correlation ($corr_{cf}$) for *f*	Total feature-feature correlation ($corr_{ff}$) for selected features	Iteration	Merit
Forest cover type	Probe3UniformBin	0.051	10.752	26	1.518
	Probe2UniformCont	0.044	10.752	27	1.509
	Probe1GaussCont	0.037	10.752	28	1.499
	SoilType20	0.161	11.352	29	1.489
	SoilType25	0.08	11.569	30	1.48
	SoilType13	0.527	14.095	31	1.471
	SoilType15	0.028	14.095	32	1.462
	SoilType39	0.676	17.99	33	1.447
KDD Cup 1999	Probe1GaussCont	0.032	12.589	21	1.309
	Probe2UniformCont	0.028	12.589	22	1.299
	SUAttempted	0.021	12.589	23	1.289
	Land	0.001	12.589	24	1.276
	IsHostLogin	0	12.589	25	1.263
	DstHostSrvSerrorRate	0.855	17.715	26	1.25
	DstHostSrvRerrorRate	0.847	23.547	27	1.237
	Count	0.631	28.713	28	1.22

Figures 5.1 and 5.2 respectively show the plots of the merit measures when the forward search procedure was implemented with the CFS merit measure, and when it was implemented with the differential prioritization (DP) measure. For the DP measure figures 5.1 and 5.2 show the plots of merit values for $\alpha = 0.5$ (DP050 Merit), $\alpha = 0.75$ (DP075 Merit) and $\alpha = 0.95$ (DP095 Merit). The plots show the values of the merit measures for each iteration of the search procedure until all features have been processed. A disturbing observation is that it is not obvious from the plots when the search procedure should terminate. Hall (1999) has stated that in the presence of feature interactions, CFS may fail to select the optimal subset of features. The discussion of detailed executions of the CFS search procedure provided earlier in this section demonstrated that the CFS merit measure can favour noise over true features. For the differential prioritisation measure, the stopping criterion is easier to determine, since Ooi et al (2007) have stated that one objective of the search procedure which uses this measure is to identify a pre-specified number of features. For the differential prioritisation measure it is difficult to justify the choice of the α value in relation to any precise definition of relevance and redundancy unless domain-specific information is available.

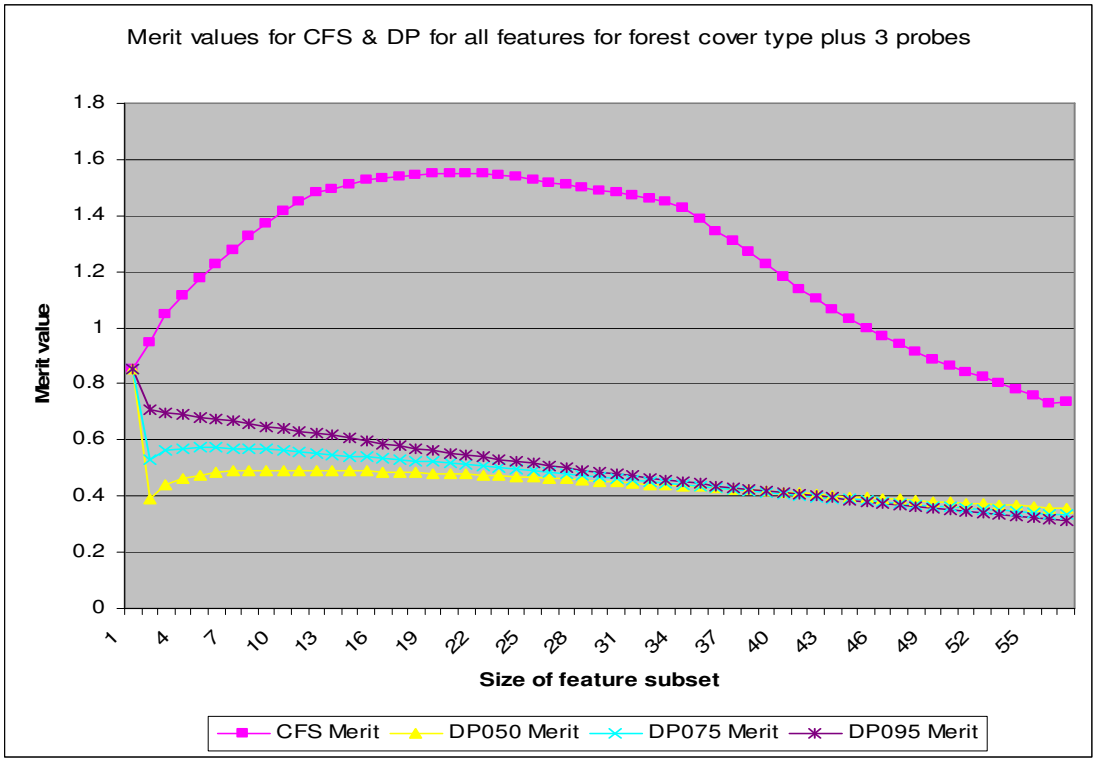

Figure 5.1: Merit values for the forest cover type dataset without pre-selection

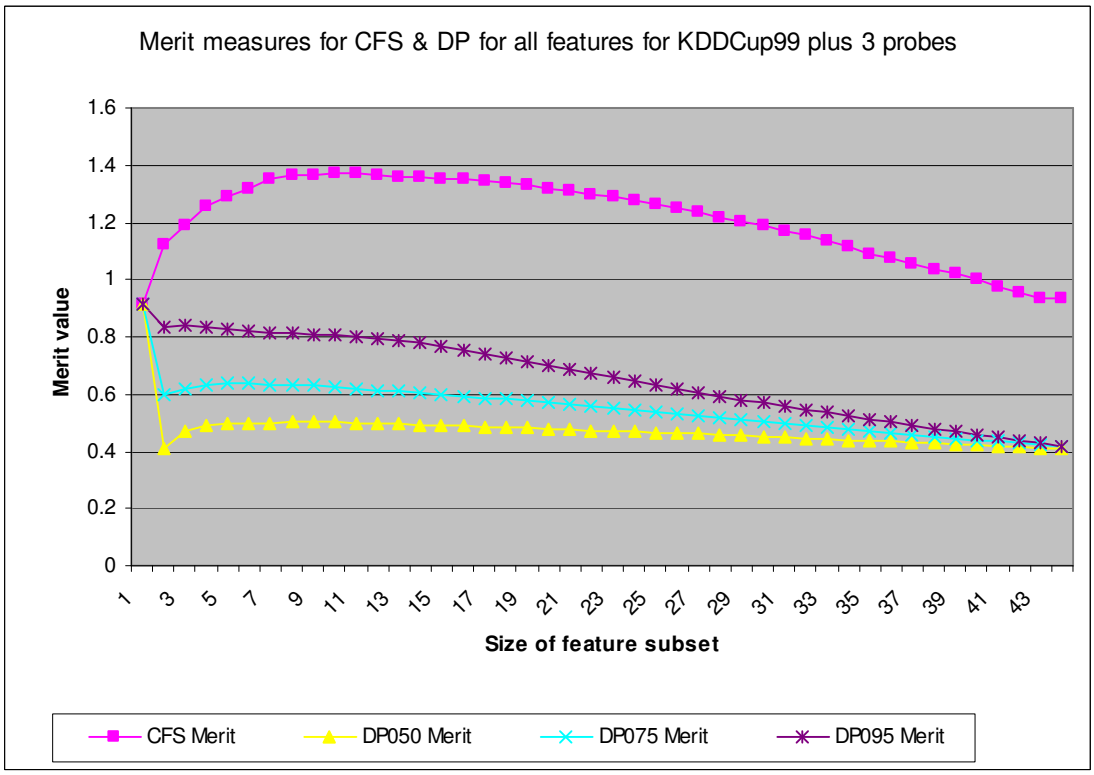

Figure 5.2: Merit values for the KDD Cup 1999 dataset without feature pre-selection

The experimental results reported in this section have revealed two weaknesses of the merit measures employed by the CFS and differential prioritisation search

procedures. Firstly, the mathematical functions used as merit measures sometimes select pure noise in preference to predictive features. Secondly, the stopping criteria for the two search procedures can be difficult to establish for some datasets. This was found to be the case for the forest cover type dataset and for the KDD Cup 1999 dataset. Based on the foregoing observations the author was led to hypothesise that the use of more precise definitions for interpretation of correlation values should eliminate the above problems that arise with the CFS and differential prioritisation merit measures.

5.4.2 A reliable search procedure for feature subset search

One possible solution to the problems exhibited by the CFS and DP merit measures is to use a feature selection criterion that precisely implements a given definition of relevance and redundancy. The definition of feature relevance should be supplied by domain experts in terms of what values of correlations are considered to be *low*, *medium* and *high*. The idea of incorporating user supplied domain knowledge in model construction is not new. Osei-Bryson (2004) has proposed the incorporation of user-specified preferences and value functions in the post pruning of classification trees. Yu and Liu (2004) have proposed the incorporation of user-specified threshold values of class-feature correlations for feature relevance analysis and selection. The method of differential prioritisation proposed by Ooi et al (2007) enables a user to control the levels of feature relevance and redundancy for the selected feature subset.

Formal definitions of feature relevance and redundancy were given in chapter 3. For feature selection based on relevance and redundancy analysis, Yu and Liu (2004) have defined four categories of features, namely (1) *irrelevan*t, (2) *weakly relevant and redundant,* (3) *weakly relevant and non-redundant*, and (4) *strongly relevant*. Yu and Liu (2004) have argued that the optimal subset of features should consist of features that fall in categories 3 and 4, that is, *weakly relevant and non-redundant*, and *strongly relevant*. The four categories of features proposed by Yu and Liu (2004) are based on Blum and Langley's (1997) definition of feature relevance and Koller and Sahami's (1996) definition of redundancy as discussed in chapter 3.

A feasible refinement of the feature relevance and redundancy definitions of table 5.7 is shown in table 5.9 for purposes of heuristic feature subset search. The refinement

is based on all possible combinations of the levels *insignificant, low, medium* and *high* correlation for class-feature and feature-feature correlations. Columns 5 and 6 respectively show the interpretation of each combination and the source of motivation for the interpretation. Column 2 shows the symbols used to label the distinct interpretations of the correlation level combinations. The categorisation suggests that unselected features fall into one of six categories at the time when the search algorithm needs to make a decision as to which feature to select next for inclusion in the set of already selected features. The six categories denoted by A,B,C,D,E and F correspond to the interpretations *strongly relevant* (category A), *relevant* (category B), *weakly relevant* (category C), *weakly redundant* (category D), *redundant* (category E), and *irrelevant* (category F).

Yu and Liu (2004) have defined four categories of features for relevance and redundancy analysis, as discussed earlier in this section. For the proposed categorisation of table 5.8 two categories (*weakly redundant* and *redundant*) are used to represent redundant features and two categories (*relevant* and *strongly relevant*) are used to represent strongly relevant features. The motivation for using six categories was to provide the heuristic search procedure with the ability to make higher precision distinctions between features compared to the level of precision provided by the CFS and DP measures.

As an example of the interpretation of the correlation levels *insignificant*, *low, medium and high* shown in table 5.9, Cohen's (1988) proposal for the interpretation of correlation coefficients for behavioural sciences research could be used. The reader will recall that according to Cohen's (1988) definitions, a correlation coefficient in the range [0,0.1) indicates a correlation with no practical significance (*insignificant*). A correlation coefficient in the range [0.1, 0.3) indicates a *low* correlation. A correlation coefficient in the range [0.3, 0.5) indicates a *medium* correlation, and a correlation coefficient in the range [0.5, 1.0] indicates a strong (*high*) correlation. The categorisation of table 5.9 can then be used as follows. When the input variables to the search procedure are pre-selected, for example, using the t-test or a probe, then (situation sp1, category F) will not arise during the search. If there is no pre-selection of features, then the situation (situation sp1, category F) may arise. The merit measure should then be replaced by clear logic which implements the interpretation of class-feature and feature-feature correlations based on table 5.9 It should be noted that the categories are dynamic, that is, the category of a given feature will

change based on the currently selected features since the mean correlation with already selected features (column 4 of table 5.9) is not a static quantity.

Table 5.9: Proposed definition of feature relevance and redundancy based on user specified levels

Situation (new interpre-tation)	Category	class-feature correlation for *f*: $corr_{cf}(f)$	mean correlation with selected features: $\overline{corr_{ff}}(f)$	Proposed new interpretation	Source of motivation for interpretation of category
sp1	F	insignificant	any level	irrelevant	(Blum & Langley, 1997) and (Yu & Liu,2004)
sp2	C	Low	insignificant	weakly relevant	
sp3	C	Low	Low	weakly relevant	
sp4	C	Low	Medium	weakly relevant	
sp5	F	Low	High	irrelevant	
sp6	B	Medium	insignificant	relevant	
sp7	B	Medium	Low	relevant	
sp8	D	Medium	Medium	weakly redundant	(Koller & Sahami,1996) and (Yu & Liu,2004)
sp9	E	Medium	High	redundant	
sp10	A	High	Insignificant	strongly relevant	(Blum & Langley, 1997) and (Yu & Liu,2004)
sp11	A	High	Low	strongly relevant	
sp12	D	High	Medium	weakly redundant	(Koller & Sahami,1996) and (Yu & Liu,2004)
sp13	E	High	High	redundant	

A new search algorithm was designed to use the categorisation shown in table 5.9 to conduct a search for the best subset of features. In general, a heuristic search procedure creates a search tree whose nodes represent various states of the search space (Luger & Stubblefield, 1993; Pearl, 1984). The heuristic search procedure will expand that node which is most promising based on the value of a heuristic (merit) measure. Typical implementations of heuristic search employ several lists to record the current state of the search tree. The new algorithm, which is given in figure 5.3, uses a list called FEATURES to hold all currently unselected features, a list called CHILDREN to hold all the nodes for features that are candidates for selection, and a list called SELECTED to hold all currently selected nodes. Initially, the SELECTED list holds the feature with the highest $corr_{cf}$ value. When making a decision on the next feature to include in the SELECTED list, the algorithm will prefer a strongly relevant feature, if such a feature exists. If there are no strongly relevant features at that point, the algorithm will prefer a relevant feature. If there are no strongly relevant

or relevant features at that point, the algorithm will prefer a weakly relevant feature. If there are no strongly relevant, relevant or weakly relevant features at that point, the algorithm will prefer a weakly redundant feature. If there is no feature which falls in one of the categories strongly relevant, relevant, weakly relevant or redundant, the algorithm terminates. The motivation for allowing the algorithm to select weakly redundant features is due to the fact that Guyon & Elisseeff (2003) have reported experiments which demonstrate that feature interactions are not necessarily harmful.

Initialise

Step 1: Place all features and their correlations on the FEATURES list

Step 2: Create a node for the feature with the highest $corr_{cf}$ and place the node on the SELECTED list

Process:

Step 3: If the FEATURES list is not empty, create a node for each feature on FEATURES and place it on the CHILDREN list. If the FEATURES list is empty, go to step 11.

Step 4: For each node on the CHILDREN list, establish the $corr_{cf}$ and mean $corr_{ff}$ with already selected features on SELECTED.

Step 5: Assign a category to each node on the CHILDREN list, based on table 5.9.

Step 6: Delete from the CHILDREN list any node which belongs to the E and F categories. This leaves the categories strongly relevant (A), relevant (B), weakly relevant (C) and weakly redundant (D).

Step 7: If the CHILDREN list has only one feature in category A (strongly relevant), put it on SELECTED and delete it from the FEATURES list and go to step 3.

If there is more than one feature in category A, then call function *GetBestInCat* to select the best one. Go to step 3. Otherwise go to step 8.

Step 8: If the CHILDREN list has only one feature in category B (relevant), put it on SELECTED and delete it from the FEATURES list and go to step 3.

If there is more than one feature in category B, then call function *GetBestInCat* to select the best one and go to step 3. Otherwise go to step 9.

Step 9: If the CHILDREN list has only one feature in category C (weakly relevant), put it on SELECTED and delete it from the FEATURES list and go to step 3.

If there is more than one feature in category C, then call function *GetBestInCat* to select the best one. Go to step 3. Otherwise go to step 10.

Step 10: If the CHILDREN list has only one feature in category D (weakly redundant), put it on SELECTED and delete it from the FEATURES list and go to step 3.

If there is more than one feature in category D, then call function *GetBestInCat* to select the best one and go to step 3. Otherwise go to step 11.

Step 11: Terminate the search procedure and return the nodes on SELECTED as the selected features.

Figure 5.3: Decision rule-based algorithm based on definitions of relevance and redundancy

At the time of selecting a feature for inclusion in the SELECTED list, if more than one feature fall in the preferred category, the algorithm uses the decision rules shown in table 5.10 to choose between two features, f_1 and f_2. The rules of table 5.10 are implemented in the function *Better_than(f1,f2)* which returns true if f_1 is better than f_2 (i.e. the decision should be to choose f_1). Figure 5.4 shows the algorithm for the function *GetBestInCat(CT)* for searching for the best feature of the *CT* category. This function utilises the function *Better_than(f1,f2).*

```
1.  first =  index of first node in  category CT
2. best = CHILDREN[first]
3. count = number of nodes on the CHILDREN list
4. For (i=first+1; i < count, i++)
        feature = CHILDREN[i]
        if (feature belongs to category CT) and Better_than(feature, best)
              best = feature
  end-for
4.  Return best
```

Figure 5.4: The algorithm GetBestInCat(CT) to select the best features in one category

Table 5.10: Decision rules for choosing between two features of the same category

Class-feature correlation	mean feature-feature correlation with selected features	Decision	Reason
$corr_{cf}(f_1) > corr_{cf}(f_2)$	$corr_{ff}(f_1) \leq corr_{ff}(f_2)$	choose f_1	prefer feature with higher class-feature & lower feature-feature correlation
	$corr_{ff}(f_1) > corr_{ff}(f_2)$	choose f_2	prefer feature with lower feature-feature correlation
$corr_{cf}(f_1) < corr_{cf}(f_2)$	$corr_{ff}(f_1) < corr_{ff}(f_2)$	chose f_2	prefer feature with higher class-feature correlation
	$corr_{ff}(f_1) \geq corr_{ff}(f_2)$	choose f_2	
$corr_{cf}(f_1) = corr_{cf}(f_2)$	$corr_{ff}(f_1) < corr_{ff}(f_2)$	choose f_1	prefer feature with lower feature-feature correlation
$corr_{cf}(f_1) = corr_{cf}(f_2)$	$corr_{ff}(f_1) > corr_{ff}(f_2)$	choose f_2	
$corr_{cf}(f_1) = corr_{cf}(f_2)$	$corr_{ff}(f_1) = corr_{ff}(f_2)$	break tie randomly	identical levels of relevance & redundancy

The results for the feature subset search for the KDD Cup 1999 and forest cover type datasets are respectively listed in tables 5.11 and 5.12. The tables show the search results when the input list consists of all features, including probes. The summary results for the four datasets used in the experiments are given in table 5.13. The results of tables 5.11 and 5.12 show that the search algorithm does not select any probes.

Table 5.11: Output of the decision rule-based search algorithm without feature pre-selection for KDD Cup 1999

Feature	Category	Selection Reason	$corr_{cf}(f)$	*mean* $corr_{ff}(f)$	Number of elected features
SerrorRate			0.916	0	1
DstHostRerrorRate	A	Strongly relevant	0.805	0.272	2
NumRoot	A	Strongly relevant	0.677	0.281	3
WrongFragment	A	Strongly relevant	0.901	0.281	4
Flag	B	Relevant	0.428	0	5
NumFailedLogins	B	Relevant	0.303	0	6
DstHostSerrorRate	A	Strongly relevant	0.835	0.278	7
DstHostSrvCount	B	Relevant	0.313	0.227	8
SrvCount	B	Relevant	0.423	0.268	9
DstHostCount	B	Relevant	0.368	0.258	10
Hot	A	Strongly relevant	0.845	0.289	11
Service	C	Weakly relevant	0.236	0	12
NumAccessFiles	C	Weakly relevant	0.177	0	13
NumCompromised	A	Strongly relevant	0.915	0.298	14
Counted	A	Strongly relevant	0.631	0.281	15
ProtocolType	C	Weakly relevant	0.151	0	16
SrvDiffHostRate	B	Relevant	0.455	0.286	17
SrcBytes	B	Relevant	0.49	0.297	18
RootShell	C	Weakly relevant	0.108	0	19
NumShells	C	Weakly relevant	0.204	0.098	20
NumFileCreations	C	Weakly relevant	0.297	0.139	21
DstHostSrvRerrorRate	A	Strongly relevant	0.847	0.295	22
DstHostSameSrcPortRate	C	Weakly relevant	0.284	0.262	23
DstHostDiffSrvRate	C	Weakly relevant	0.144	0.268	24
DstHostSameSrvRate	C	Weakly relevant	0.224	0.296	25
Duration	C	Weakly relevant	0.254	0.361	26
DstBytes	D	Weakly redundant	0.584	0.332	27
DstHostSrvDiffHostRate	D	Weakly redundant	0.439	0.352	28
DstHostSrvSerrorRate	D	Weakly redundant	0.855	0.42	29
DiffSrvRate	D	Weakly redundant	0.727	0.444	30
SrvRerrorRate	D	Weakly redundant	0.845	0.456	31
RerrorRate	D	Weakly redundant	0.822	0.482	32

Secondly, the algorithm never selects irrelevant or redundant features as defined in table 5.9. Thirdly, for forest cover type the algorithm selects nearly the same number of features when pre-selection is done using probes as shown in table 5.13. Table 5.13 provides a summary of the number of features selected by the decision rule-based algorithm for different input feature sets that were generated in the experiments of the last section. The results for the features selected by the validation and ranking methods were given in table 5.6. The results of table 5.13 show that the t-test is far more restrictive compared to the pre-selection of features using probes. A comparison of tables 5.6 and 5.13 shows that the decision rule-based search algorithm selects nearly all the features that are pre-selected by the t-test.

Table 5.12: Output of the decision rule-based search algorithm without feature pre-selection for forest cover type

Feature	Category	Selection Reason	$corr_{cf}(f)$	mean $corr_{ff}(f)$	Selected feature count
WildernessArea4			0.855	0	1
SoilType2	A	Strongly relevant	0.642	0.243	2
SoilType40	A	Strongly relevant	0.547	0.146	3
SoilType38	A	Strongly relevant	0.676	0.162	4
SoilType4	A	Strongly relevant	0.638	0.164	5
SoilType1	A	Strongly relevant	0.686	0.178	6
SoilType3	A	Strongly relevant	0.548	0.185	7
SoilType6	A	Strongly relevant	0.603	0.192	8
SoilType13	A	Strongly relevant	0.527	0.199	9
SoilType39	A	Strongly relevant	0.676	0.283	10
SoilType21	B	Relevant	0.322	0	11
SoilType35	B	Relevant	0.443	0.02	12
SoilType12	A	Strongly relevant	0.704	0.286	13
SoilType34	B	Relevant	0.4	0.021	14
SoilType19	B	Relevant	0.351	0.02	15
SoilType22	A	Strongly relevant	0.593	0.287	16
SoilType18	B	Relevant	0.44	0.04	17
SoilType26	B	Relevant	0.431	0.038	18
SoilType17	B	Relevant	0.433	0.066	19
SoilType10	A	Strongly relevant	0.579	0.295	20
SoilType5	B	Relevant	0.36	0.066	21
SoilType16	B	Relevant	0.329	0.084	22
SoilType11	B	Relevant	0.476	0.162	23
WildernessArea2	B	Relevant	0.391	0.221	24
SoilType30	B	Relevant	0.34	0.266	25
SoilType14	C	Weakly relevant	0.231	0	26
SoilType8	C	Weakly relevant	0.176	0	27
SoilType37	C	Weakly relevant	0.126	0	28
SoilType9	C	Weakly relevant	0.283	0.018	29
SoilType28	C	Weakly relevant	0.215	0.018	30
SoilType27	C	Weakly relevant	0.147	0.017	31
SoilType23	B	Relevant	0.399	0.293	32
SoilType20	C	Weakly relevant	0.161	0.052	33
SoilType24	C	Weakly relevant	0.259	0.203	34
SoilType31	C	Weakly relevant	0.223	0.25	35
HorizDistToFire	C	Weakly relevant	0.156	0.263	36
HorizDistToRoad	C	Weakly relevant	0.158	0.266	37
Slope	C	Weakly relevant	0.124	0.283	38
SoilType33	C	Weakly relevant	0.183	0.323	39
SoilType32	C	Weakly relevant	0.207	0.349	40
Elevation	C	Weakly relevant	0.277	0.377	41
SoilType29	C	Weakly relevant	0.295	0.465	42

The point was made in the last section that the t-test is more strict than the probes at eliminating irrelevant features. When there is no pre-selection of features for the input to the decision rule-based search algorithm, or when the input consists of features

pre-selected using the Gaussian probe, the decision rule-based search algorithm eliminates a larger number of the input features compared to when the input features are pre-selected by the t-test. A tentative conclusion that can be made from this observation is that the probes admit some features that are possibly irrelevant. A more detailed discussion of this issue is given later in this chapter.

Table 5.13: Features selected by the decision rule-based algorithm for sample sizes of 1000

Dataset (number of features)	Number of features selected by the decision rule algorithm when the input is pre-selected using:				
	no pre-selection	Gaussian probe	Uniform probe	Uniform-bin probe	t-test (sig =0.01)
Forest cover type (54)	42	41	41	41	36
KDD Cup 1999 (41)	32	34	34	34	30
Abalone (8)	3	3	3	3	2
Mushroom (22)	14	14	14	14	-

5.5 Predictive performance for features selected with different methods

Classifiers were constructed to compare the predictive performance of the features selected by the different methods of feature selection. The 5NN and See5 classification tree algorithms were used for classification. In this section the results and analysis of the predictive performance of the classifiers are reported. Section 5.5.1 provides a description of the experimental procedures. The Predictive performance of the forest cover type and KDD Cup 1999 classifiers are respectively given in sections 5.5.2 and 5.5.3. Predictive performance results for the small dataset classifiers (abalobe3C and mushroom) are presented in section 5.5.4.

5.5.1 Experimental procedure for classifier creation and testing

The point was made in section 5.2 that the objective of feature selection should **not be** to identify the best features for the training sample that has been chosen (or happens to be available) but rather to identify the best features for model creation for any sample taken from the instance space for the prediction task. This observation was based on Hand's (1998) advice quoted at the beginning of this chapter. Based on the foregoing observation, training samples much larger than the samples used for feature selection were used for the experiments. Classification models were created for purposes of testing the predictive performance of the feature subsets selected by the feature selection methods presented in sections 5.3 and 5.4. Two

classification algorithms 5NN and See5, were used in the experiments. For each dataset the same training set (samples) and same test set (samples) were used for 5NN and See5 classification. Predictive accuracy (on instances not seen during training) was established on 10 test sets for each classifier.

In section 5.3 feature ranking was reported for four datasets, two sizes of samples (500 and 1000) for measuring correlations and four validation methods (three probes and t-test). In section 5.4 feature subset search was reported for the decision rule-based algorithm for four datasets, five types of input features (no pre-selection, three probes and t-test) selected using sample sizes of 1000 to measure correlations. To conduct experiments to test all the feature ranking methods on two algorithms would require 4 x 2 x 4 x 2 = 64 classifiers to be created. To generate test results for 10 test sets (samples) for each classifier would result in 640 test runs. To conduct experiments to test feature subsets selected with the decision rule-based algorithm would require 5 x 4 x 2 = 40 classifiers. The generation of test results for 10 test sets (samples) for each classifier would result in 400 test runs. Additionally, four classifiers must be created with all the features (no selection) and tested on 10 test sets for comparison with the classifiers created with selected feature subsets, which results in an additional 40 test runs. The total number of test runs would then be 640 + 400 + 40 = 1080. If two types of class distributions are used, as was done for the experiments, this number would double to 2160.

To avoid the factorial explosion in the number of test runs as described above, researchers are advised to sample from the space of all possible factor combinations (Cohen, 1995:pg 88). The decision made for the experiments was as follows: Only feature subsets selected using correlations measured with samples of size 1000 were used. For feature ranking methods feature subsets selected by one probe were used (Gaussian probe for forest cover type, KDD Cup 1999 and abalone3C, and uniform probe for mushroom). These were compared to classifiers constructed with all the features. Classifiers were also constructed for feature subsets selected by the decision rule-based method with one type of input (no pre-selection of features). For the forest cover type dataset (the largest dataset) two sample sizes of 6000 and 12000 instances were used. For the other three (smaller datasets), one sample size was used. Additionally, for the large datasets classifiers were created for two types of class distributions to illustrate the difficulty of establishing the true positive rates (TPRATE) for individual classes when the parent dataset class distribution is used in

the presence of minority classes. This resulted in the number of test runs being reduced to 400.

5.5.2 Classification results for forest cover type

Classifiers were constructed to compare the predictive performance obtained when the feature sets selected by the different methods as discussed in section 5.5.1 are used. Tables 5.14 shows the classification results for the forest cover type dataset samples on two class distributions. Column 4 gives the classification results for tests based on the class distribution of the parent dataset. 10-fold cross validation was used to measure the predictive accuracy. Column 5 shows the classification results for tests based on an equal class distribution. Ten test samples were used to measure the predictive accuracy. The results are shown for two sample sizes (6000 and 12000). For each sample size the predictive accuracy on all the classes is shown for all features (54), features selected by the Gaussian probe (49), and features selected by the decision rule based search algorithm (42).

Table 5.14: Predictive accuracy for forest cover type based on two class distributions

Algorithm	Feature selection method (number of features)	Sample size	Mean predictive accuracy and 95% CI of mean	
			10-fold cross validation parent dataset distribution	10 test sets equal class distribution
5NN (nearest neighbours)	All features (54)	6000	71.2 ± 1.1	75.1 ± 1.4
		12000	76.2 ± 0.8	80.1 ± 1.0
	Gaussian probe (49)	6000	71.5 ± 1.1	75.1 ± 1.1
		12000	76.1 ± 0.8	79.4 ± 0.8
	Decision rule search (42)	6000	68.5 ± 1.4	71.6 ± 1.2
		12000	70.4 ± 1.1	74.4 ± 0.9
See5 (classification tree)	All features (54)	6000	74.8 ± 1.2	73.6 ± 0.9
		12000	74.5 ± 0.8	76.6 ± 0.9
	Gaussian probe (49)	6000	73.8 ± 1.2	73.6 ± 0.9
		12000	75.4 ± 1.1	76.5 ± 0.9
	Decision rule search (42)	6000	72.8 ± 0.8	72.3 ± 0.7
		12000	74.5 ± 0.6	76.9± 1.0

Table 5.15 gives the results of the statistical tests to compare the predictive performance of the classifiers based on the parent dataset class distribution. The paired samples t-test is not applicable in this case since there are no paired tests, so the independent samples t-test was used to compare the performance of two classifiers. The independent samples t-test revealed that the predictive performance of the 5NN classifiers constructed with all 54 features and those constructed with 49 features as selected by the Gaussian probe do not statistically differ in predictive

accuracy, for all sample sizes. However, the classifiers constructed with 42 features (as selected by the decision rule-based search) have a predictive performance that is significantly lower than that when all 54 features are used. For the classifiers constructed using the See5 classification tree algorithm, there is no statistically significant difference between using all features (54) and features selected by the Gaussian probe (49). There was also no significant difference between using all features (54) and features selected by the decision rule search algorithm (42).

Table 5.15: Statistical tests to compare the accuracy of forest cover type classifiers for different feature subsets for parent dataset class distribution

Algorithm	Groups for independent samples t-test, sample size, number of features		Student's independent samples t-test (9df) (equal variances not assumed)		
	Group A (mean & CI)	Group B (mean & CI)	95% CI of mean difference	p-value (2 tails)	Group A better than Group B?
5NN (nearest neighbour)	6000; 54 (71.2 ± 1.1)	6000; 42 (68.5 ± 1.4)	[1.8, 3.8]	0.000	yes
	12000; 54 (76.2 ± 0.8)	12000; 42 (70.4 ± 1.1)	[5.2, 6.4]	0.000	yes
	6000; 54 (71.2 ± 1.1)	6000; 49 (71.5 ± 1.1)	[-1.0, 0.5]	0.468	no
	12000; 54 (76.2 ± 0.8)	12000; 49 (76.1 ± 0.8)	[-0.4, 0.6]	0.640	no
See5 (classification tree)	6000; 54 (74.8 ± 1.3)	6000; 42 (72.8 ± 0.8)	[0.3, 3.6]	0.019	yes
	12000; 54 (74.5 ± 0.8)	12000; 42 (74.5 ± 0.6)	[-1.0, 1.0]	0.952	no
	6000; 54 (74.7 ± 1.2)	6000; 49 (73.8 ± 1.2)	[-0.9, 2.8]	0.283	no
	12000; 54 (74.5 ± 0.8)	12000; 49 (75.4 ± 1.1)	[-2.4, 0.5]	0.176	no

Table 5.16 shows the results for the Student's paired samples t-test for the predictive accuracy on the 10 test sets. Columns 2 and 3 respectively provide the description of the classifiers that were compared. A specification of the training set size for the classifier, the size of the feature set that was used for the classifier and the mean predictive accuracy of the classifier on 10 test samples are given. Column 4 gives the 95% confidence interval of the mean difference for the predictive accuracy of the two classifiers specified in columns 2 and 3. Columns 5 and 6 respectively give the p-value for the paired samples t-test and the interpretation of the p-value based on the reasoning given in table 4.11 of chapter 4.

For the 5NN classifiers created with training sample sizes of 6000 and 12000 instances, the 54-feature classifiers provided a higher level of predictive accuracy compared to the 42-feature classifiers. The 49-feature classifiers provided a higher level of predictive accuracy than the 42-feature classifiers. These results indicate that the decision rule-based algorithm based on Cohen's (1998) thresholds for *insignificant, low, medium* and *high* correlations eliminates some features which have predictive power for the 5NN algorithm. For the 5NN classifiers created with training samples of 6000 and 12000 instances there is no statistically significant difference in predictive accuracy between the 54-feature classifiers and the 49-feature classifiers. These results indicate that the Gaussian probe eliminates only features with no predictive power for 5NN.

For the See5 classifiers created with training sample sizes of 6000 instances, the 54-feature classifiers provided a higher level of predictive accuracy than the 42-feature classifiers. The 49-feature classifiers also provide a higher level of predictive accuracy that the 42-feature classifiers. For classifiers created with training sample sizes of 12000 instances there was no statistically significant difference in predictive accuracy between the 54-feature and 42-feature classifiers, and between the 54-featute and 42-feature classifiers. These results indicate that for the See5 classifiers the 42 features selected by the decision rule-based algorithm based on Cohen's (1998) guidelines are sufficient for prediction with very large samples (e.g. 12000 instances).

A detailed analysis of the 5NN and See5 classifiers was conducted for the classifiers of sample size 12000 in order to establish the TPRATE values for the individual classes. The analysis results are given in table 5.17. The analysis was done to compare the predictive performance of the 49 features selected by the Gaussian probe and the 42 features selected by the decision rule-based search algorithm. The results of the Student's paired samples t-test for the 5NN classifiers indicate that for three of the classes (1, 2, 7) there is no statistically significant difference between using 49 features and 42 features for 5NN classification. However, for four of the classes (3, 4, 5, 6) there is a statistically significant increase in the TPRATE values when 49 features are used. The results of the Student's paired samples t-test for the See5 classifiers indicate that there is no statistically significant difference between using 49 features and 42 features for five of the classes (2, 3, 4, 5, 6). The TPRATE for the 49-feature classifier is statistically significantly higher than that for the 42-

feature classifier for class 1. The TPRATE for the 42-feature classifier is statistically significantly much higher than that for the 49-feature classifier for class 7.

Table 5.16: Statistical tests to compare the accuracy of forest cover type classifiers for different feature subsets for equal class distribution

Algorithm	Groups for paired tests sample size; number of features		Student's paired t-test (9df)		
	Group A (mean & CI)	Group B (mean & CI)	95% CI of mean difference	p value (2 tails)	Group A better than Group B?
5NN (nearest neighbours)	6000; 49 (75.1 ± 1.1)	6000; 42 (71.6 ± 1.2)	[1.7, 5.2]	0.002	yes
	12000; 49 (79.4 ± 0.8)	12000; 42 (74.4 ± 0.9)	[3.6, 6.3]	0.000	yes
	6000; 54 (75.1 ± 1.4)	6000; 42 (71.6 ± 1.2)	[1.9, 5.0]	0.000	yes
	12000; 54 (80.1 ± 1.0)	12000; 42 (74.4 ± 0.9)	[4.2, 7.2]	0.000	yes
	6000; 54 (75.1 ± 1.4)	6000; 49 (75.1 ± 1.1)	[-1.5, 1.4]	0.451	no
	12000; 54 (80.1 ± 1.0)	12000; 49 (79.4 ± 0.8)	[-0.6, 1.8]	0.290	no
See5 (classification tree)	6000; 49 (73.6 ± 0.9)	6000; 42 (72.3 ± 0.7)	[0.3, 2.3]	0.014	yes
	12000; 49 (76.5 ± 0.9)	12000; 42 (76.9 ± 1.0)	[-1.3, 0.5]	0.324	no
	6000; 54 (73.6 ± 0.9)	6000; 42 (72.3 ± 0.7)	[0.3, 2.3]	0.014	yes
	12000; 54 (76.6 ± 0.9)	12000; 42 (76.9 ± 1.0)	[-1.2,0.6]	0.490	no
	6000; 54 (73.6 ± 0.9)	6000; 49 (73.6 ± 0.9)	no variance	no variance	no
	12000; 54 (76.6 ± 0.9)	12000; 49 (76.5 ± 0.9)	[-0.01, 0.3]	0.495	no

When samples are randomly selected from a large dataset and the class-feature correlations are measured, the correlation values obtained reflect the predictive power of the features over the whole instance space. This is called *global predictive power* in this thesis. If a large dataset was clustered and samples taken from each cluster to measure the class-feature correlations, then the correlation values obtained would reflect the predictive power of the features for a given cluster. Features that have significant class-feature correlations only for a cluster and not for the whole instance space are said to be locally predictive within that cluster. This is called *local predictive power* in this thesis.

Table 5.17: Statistical tests to compare TPRATE performance of forest cover type classifiers for different feature subsets for training sample size 12000

Algorithm	Groups for paired tests sample size; number of features		Student's paired t-test (9df)		
	Group A (12000;49)	Group B (12000;42)	95% CI of mean difference	p value (2 tailed)	Group A better than Group B?
5NN (nearest neighbours)	All classes-A (79.4 ± 0.8)	All classes-B (74.4 ± 0.9)	[3.6, 6.3]	0.000	yes
	Class 1-A (60.6 ± 3.0)	Class 1-B (60.4 ± 4.0)	[-6.3, 6.7]	0.473	no
	Class 2-A (46.2 ± 3.1)	Class 2-B (47.8 ± 4.4)	[-4.4, 1.2]	0.224	no
	Class 3-A (67.2 ± 4.0)	Class 3-B (54.4 ± 2.1)	[7.7, 17.9]	0.000	yes
	Class 4-A (97.4 ± 0.4)	Class 4-B (90.4 ± 3.0)	[3.3, 10.7]	0.002	yes
	Class 5-A (97.6 ± 0.8)	Class 5-B (93.8 ± 0.9)	[2.5, 5.1]	0.000	yes
	Class 6-A 82.0 ± 2.9)	Class 6-B (72.2 ± 2.9)	[5.1, 14.5]	0.000	yes
	Class 7-A (94.8 ± 1.0)	Class 7-B (92.4 ± 4.1)	[-2.0, 6.8]	0.250	no
See5 (classification tree)	All classes-A (76.5 ± 0.9)	All classes-B (76.9 ± 1.0)	[-1.3,0.5]	0.324	no
	Class 1-A (61.4 ± 4.1)	Class 1-B (57.4 ± 3.4)	[1.3, 6.7]	0.008	yes
	Class 2-A (61.2 ± 3.0)	Class 2-B (63.8 ± 3.0)	[-5.2, 0.2]	0.050	no
	Class 3-A (64.8 ±3.4)	Class 3-B (60.8 ± 3.3)	[-0.4, 8.4]	0.034	yes
	Class 4-A (96.6 ± 1.0)	Class 4-B (96.8 ± 1.0)	[-2.2, 1.8]	0.414	no
	Class 5-A (84.0 ± 1.7)	Class 5-B (86.2 ± 2.4)	[-5.2, 0.8]	0.128	no
	Class 6-A (79.8 ± 2.5)	Class 6-B (77.8 ± 3.3)	[-0.1, 2.1]	0.062	yes
	Class 7-A (87.6 ± 3.4)	Class 7-B (95.6 ± 1.6)	[-11.2, -4.8]	0.000	no

The observations for the test results of tables 5.16 and 5.17 led the author to hypothesise as follows: If a large dataset has features that only have local predictive power, such features will have small class-feature correlations and will therefore appear to be non-relevant when one of the validation methods (Student's t-test of

means) and decision rule-based search algorithm studied in this chapter are used. This hypothesis was not tested in this thesis, and is left for future work.

5.5.3 Classification results for KDD Cup 1999

5NN and See5 classification models were also constructed for the KDD Cup 1999 dataset. The challenge for the KDD Cup 1999 dataset is to achieve a high level of accuracy on the attack classes R2L and U2R on the test dataset. The KDD Cup 1999 test dataset was presented in chapter 4. Classification performance results for this dataset are most commonly presented in terms of the TPRATE values for the classes (Lee et al, 2002; Lee & Stolfo, 2000). The predictive performance results are therefore presented here in terms of the accuracy on all classes as well as the TPRATE values for each of the 5 classes. Table 5.18 shows the predictive performance of the classifiers. The performance results are shown for 10-fold cross validation on the training set, and for 10 test samples drawn from the test dataset.

For 10-fold cross validation, a training sample of 4500 instances was used. For the minority class, U2R, all 52 instances of that class were included in the sample. For the remaining four classes, sequential random sampling was used. For the classifiers based on an equal distribution of the classes, a training sample of 4500 instances was created with 1000 instances from each of the four classes NORMAL, DOS, PROBE and R2L, and 500 instances for the class U2R. The 500 instances of the class U2R were obtained using bootstrap sampling of the 52 instances that appear in the training dataset. The aim was to try as much as possible to achieve an equal distribution, but a decision was made not to bootstrap the U2R class beyond ten times the actual size. The test samples were created by taking all 70 instances of the class U2R in the test dataset and using sequential random sampling to obtain 70 instances for each of the remaining classes.

From table 5.6 it can be deduced that the 3 probes select nearly the same numbers of features for the KDD Cup 1999 dataset. The results of table 5.13 show that all input feature subsets result in nearly the same number of features being selected, with the decision rule-based search algorithm selecting the smallest number of features. Classifiers were constructed with all the 41 features (all features) and with the 32 features selected by the decision rule based search algorithm. Classifiers were not constructed with the 36 features selected by the Gaussian probe as initial

exploratory studies (Cohen, 1995) revealed that the predictive performance of 41 features is not significantly different from that of the 32 features.

Table 5.18: Predictive performance of KDD Cup 1999

Algorithm (training sample size)	Feature selection method (number of features)	Class	Natural distribution mean TPRATE% for 10-fold cross validation	Equal class distribution mean TPRATE% for 10 test sets
5NN (nearest neighbours) (size = 4500)	All features (41)	All classes (accuracy)	97.0 ± 0.5	73.5 ± 0.9
		NORMAL	98.4	85.6 ± 3.2
		DOS	97.5	67.3 ± 5.0
		PROBE	60.0	95.9 ± 1.2
		R2L	61.9	73.1 ± 2.2
		U2R	65.2	45.7 ± 0.0
	Decision rule (32)	All classes (accuracy)	96.4 ± 0.5	69.9 ± 1.3
		NORMAL	98.2	84.7 ± 3.2
		DOS	97.0	67.1 ± 4.8
		PROBE	94.7	95.9 ± 1.2
		R2L	72.3	70.3 ± 3.3
		U2R	38.0	31.4 ± 0.0
See5 (classification tree) (size = 4500)	no selection (41)	All classes (accuracy)	98.6 ± 0.5	66.3 ± 1.2
		NORMAL	99.6	97.3 ± 1.1
		DOS	99.5	74.3 ± 6.2
		PROBE	98.2	86.2 ± 2.2
		R2L	76.5	25.4 ± 2.3
		U2R	71.2	48.6 ± 0.0
	Decision rule (32)	All classes (accuracy)	97.5 ± 0.5	66.3 ± 1.2
		NORMAL	99.3	97.3 ± 1.1
		DOS	96.3	74.3 ± 6.2
		PROBE	98.4	86.2 ± 2.2
		R2L	68.4	25.4 ± 2.3
		U2R	65.4	48.6 ± 0.0

From the 10-fold cross validation results of table 5.18 it can be deduced that predictive accuracy does not differ significantly for the 41-feature and 32-feature 5NN classifiers. The 5NN classifier TPRATE values for the classes NORMAL and DOS do not differ significantly. There are significant differences in the 5NN classifier TPRATE values for the classes PROBE, R2L and U2R. The 10-fold cross validation results for the See5 classifiers indicate that the TPRATE values for the majority classes (NORMAL, DOS, PROBE) are nearly identical. The See5 TPRATE values for the 41-feature classifiers are significantly higher than those for the 32-feature classifier for the minority classes (R2L, U2R). Overall, it is difficult to establish differences in performance when the parent dataset class distribution of the KDD Cup 1999 dataset is used.

A training dataset of size 4500 randomly selected instances was created to with a class distribution that is very close to an equal class distribution. An equal class distribution is one where instances from all the classes appear in equal proportions in a dataset. One thousand instances were selected for each of the classes, NORMAL, DOS, PROBE and R2L. The 52 instances for the minority class (U2R) were bootstrapped to 500 instances. Test samples of 350 instances with an equal class distribution were created.

Table 5.19: Statistical tests to compare the performance of KDD Cup 1999 classifiers for different feature subsets

Algorithm (training size)	Groups for the paired tests training sample size; number of features		Student's paired t-test (9df)		
	Group A (4500;41)	Group B (4500;32)	95% CI of mean difference	p value (2 tailed)	Group A better than Group B?
5NN (nearest neighbour) (size=4500)	All classes-A (73.5 ± 0.9)	All classes-B (69.9 ± 1.3)	[3.1, 4.1]	0.000	yes
	NORMAL-A (85.6 ± 3.2)	NORMAL-B (84.7 ± 3.2)	[0.1, 1.6]	0.026	yes
	DOS-A (67.3 ± 5.0)	DOS-B (67.1 ± 4.8)	[-0.4, 0.8]	0.492	no
	PROBE-A (95.9 ± 1.2)	PROBE-B (95.9 ± 1.2)	[-0.01, 0.03]	0.342	no
	R2L-A (73.1 ± 2.2)	R2L-B (70.3 ± 3.3)	[0.5, 5.1]	0.020	yes
	U2R-A (45.7 ± 0.0)	U2R-B (31.4 ± 0.0)	no variance	no variance	no variance
See5 (classification tree) (size=4500)	All classes-A (66.3 ± 1.2)	All classes-B (66.3 ± 1.2)	no variance	no variance	no
	NORMAL-A (97.3 ± 1.1	NORMAL-B (97.3 ± 1.1)	no variance	no variance	no
	DOS-A (74.3 ± 6.2	DOS-B (74.3 ± 6.2)	no variance	no variance	no
	PROBE-A (86.2 ± 2.2	PROBE-B (86.2 ± 2.2)	no variance	no variance	no
	R2L-A (25.4 ± 2.3)	R2L-B (25.4 ± 2.3)	no variance	no variance	no
	U2R-A (48.6 ± 0.0)	U2R-B (48.6 ± 0.0)	no variance	no variance	no

5NN and See5 classifiers were constructed from the training dataset of 4500 instances and tested on 10 test sets. Table 5.18 provides a summary of the classification results for the 41-feature and 32-feature classifiers. Table 5.19 shows

the results of the Student's paired samples t-test to compare the 41-feature and 32-feature classifiers. The results indicate that there is no statistically significant difference in the TPRATE values for both the 5NN and See5 classifiers the DOS and PROBE classes. The TPRATE values for the NORMAL and R2L classes are marginally higher for the 41 feature classifiers compared to the 32 features classifier. The 5NN classifier TPRATE for the U2R class is 14.3% higher for the 41-feature classifier compared to the 32-feature classifier. However, due to lack of variance, the paired t-test could not be applied. The See5 classifier TPRATE values for all classes for the 41-feature and 32-feature classifiers are equal. Again, due to the absence of variance, the paired samples t-test is not applicable.

The foregoing discussion led the author to hypothesise as follows: It is possible that features that are predictive of minority and severely under-represented classes will be eliminated when class-feature correlations are measured for all classes using instances randomly selected to represent the whole instance space. Such features are eliminated because the class-feature correlations cannot be reliably estimated. This is the case for the U2R class. This hypothesis was not tested for this thesis and is left for future work.

5.5.4 Classification results for the small datasets

Classifiers were constructed for the abalone3C and mushroom datasets to compare the predictive performance obtained when there is no feature selection and when the features selected by the decision rule-based algorithm are used. The results of table 5.13 show that for the abalone3C dataset the probes did not eliminate any features. For the mushroom dataset the number of features selected by the decision rule-based algorithm is the same as the number of features selected by the uniform-binary probe. 5NN and See5 classifiers were constructed for both datasets using 10-fold cross validation on randomly selected training samples from the datasets. The training sample size used for abalone3C was 3000 instances. Training sample sizes of 600 and 3000 instances were used for the mushroom dataset since training sample sizes of 3000 instances produced a ceiling effect (Cohen, 1995). A ceiling effect is encountered when the performance level of a system on a given task is so high that it is not possible to demonstrate performance improvements with any intervention (Cohen, 1995). Experiments with 10 test sets were not conducted as was done for the large datasets since the abalone3C and mushroom datasets have

balanced class distributions. Tables 5.20 show the classification results for the two datasets.

The predictive accuracy for each dataset is shown for all features and for features selected by the decision rule based search algorithm. The 5NN 8-feature classifiers provided a higher level of predictive accuracy compared to the 3-feature classifiers for the abalone3C dataset. Clearly the 95% confidence intervals for the mean predictive accuracy for these classifiers do not overlap.

Table 5.20: Predictive accuracy for the small datasets based on the parent dataset class distribution

Dataset (training set size)	Classifier	Feature selection method (number of features)	Mean predictive accuracy & 95% CI of mean with 10-fold cross validation
Abalone3C (3000)	5NN (nearest neighbours)	All features (8)	59.8 ± 2.1
		Decision rule search (3)	52.3 ± 3.2
	See5 (classification tree)	All features (8)	63.3 ± 1.3
		Decision rule (3)	56.7 ± 1.6
Mushroom (3000)	5NN (nearest neighbours)	All features (22)	97.9 ± 1.4
		Decision rule search (14)	97.6 ± 2.1
	See5 (classification tree)	All features (22)	100
		Decision rule search (14)	99.9 ± 0.0
Mushroom (600)	5NN (nearest neighbours)	All features (22)	96.7 ± 2.1
		Decision rule search (14)	95.8 ± 2.8
	See5 (classification tree)	All features (22)	99.2 ± 0.6
		Decision rule search (14)	99.2 ± 0.9

The See5 8-feature classifiers also provided a higher level of predictive accuracy compared to the 3-feature classifiers for the abalone3C dataset. The 22-feature and 14-feature See5 and 5NN classifiers created with training sample sizes of 600 and 3000 provided the same level of very high predictive accuracy the mushroom dataset. Clearly the 95% confidence intervals of mean accuracy for the classifiers overlap. This is a ceiling effect which makes it difficult to make any conclusions on this dataset. The same ceiling effect was observed for both the mushroom 5NN 22-feature and 14-feature classifiers created with training sample sizes of 600 and 3000.

Table 5.21 gives the results of Student's independent samples t-test for means for the two datasets. The groups compared for each dataset were the 10-fold cross validation results when all features were used and when the features selected by the decision rule-based algorithm were used.

Table 5.21: Statistical tests to compare the predictive performance of small dataset classifiers

Dataset (training sample size)	Classifier	Groups for independent samples tests; number of features		Student's independent samples t-test (18df, equal variances assumed)		
		Group A (mean & CI)	Group B (mean & CI)	95% CI of mean difference	p value (2 tails)	Group A better than Group B?
Abalone3C (3000)	5NN	All-classes; 8 (59.8 ± 2.1)	All-classes; 3 (52.3 ± 3.2)	[3.4, 11.6]	0.001	yes
		Class-young; 8 (74.1 ± 4.5)	class-young,3 (73.7 ± 6.2)	[-7.8, 8.7]	0.912	no
		Class-middle; 8 (43.1 ± 4.9)	Class-middle;3 (51.1 ± 6.5)	[-16.8, 0.7]	0.70	no
		Class-old; 8 (61.5 ± 3.4)	Class-old; 3 (37.2 ± 6.7)	[16.3, 32.4]	0.000	yes
	See5	All-classes; 8 (63.3 ± 1.3)	All-classes; 3 (56.7 ± 1.6)	[4.3, 8.8]	0.000	yes
		Class-young; 8 (74.0)	Class-young;3 (74.5)	based on arithmetic comparison of TPRATE		no
		Class-middle; 8 (47.6)	Class-middle;3 (20.4)	based on arithmetic comparison of TPRATE		yes
		Class-old; 8 (67.3)	Class-young;3 (72.7)	based on arithmetic comparison of TPRATE		no
Mushroom (3000)	5NN	All-classes; 22 (97.9 ± 1.4)	All-classes; 14 (97.6 ± 2.1)	[-2.5, 3.0]	0.856	no
	See5	All-classes; 22 (100 ± 0.0)	All-classes; 14 (99.9 ± 0.0)	[-0.3, 0.1]	0.230	no
Mushroom (600)	5NN	All-classes; 22 (96.7 ± 2.1)	All-classes; 14 (95.8 ± 2.8)	[-3.0, 4.7]	0.652	no
	See5	All-classes; 22 (99.2 ± 0.6)	All-classes; 14 (99.2 ± 0.9)	[-1.1, 1.1]	0.970	no

Student's paired samples t-test for means is not applicable here as cross validation tests were not paired. The class TPRATE values for the abalone3C classifiers are also given. TPRATE values are not given for the mushroom classes as there were no (interesting) differences in the TPRATE values for the different feature subsets. The statistical tests on the TPRATE values for the abalone3C 5NN classifiers indicate that the performance of the 8-feature and 3-feature classifiers does not differ significantly for the classes *young* and *middle*. For the 8-feature and 3-feature See5

classifiers the TPRATE values for the classes *young* and *old* do not differ significantly, but the TPRATE for the class *middle* is much higher for the 8-feature classifier.

The abalone3C dataset is a major challenge for feature subset search algorithms which attempt to maximise class-feature association and minimise feature-feature association. Table D.8 of appendix D gives the feature-feature correlation values for this dataset. It is clear from table D.8 that generally all the abalone3C features are strongly correlated with each other. The results of table 5.21 clearly indicate that the use of eight features (all features) provides a higher level of predictive performance compared to the use of a subset of the features. Obviously, the feature interactions for abalone3C have predictive power for the class variable.

5.6 Discussion

This section provides a discussion of the experimental results of this chapter. The discussion is divided into three subsections covering correlation measurement, feature subset selection and the problems associated with the measurement of class-feature correlations for feature selection. The recommendations for feature selection from large datasets are given in chapter 10 where the general discussion of the research findings is provided. Section 5.6.1 provides a discussion of correlation measures and feature ranking. A discussion of feature subset selection is given in section 5.6.2. Section 5.6.3 discusses the problems associated with the global measurement of class-feature and feature-feature correlations.

5.6.1 Correlation measures and feature ranking

When there are no outliers in the data and associations between variables are linear Pearson's correlation coefficients are suitable for measuring correlations (Wilcox, 2001) and determining feature ranking for feature selection. This was demonstrated with the abalone3C dataset experiments of section 5.3.2. When data contains outliers or when the association between variables is non-linear, robust measures of association will provide more accurate estimates of correlation values (Wilcox, 2001). The experimental results of section 5.3.2 and 5.3.4 demonstrated that Kendall's correlation coefficient is a more accurate measure of correlation compared to

Pearson's correlation coefficient for the large datasets used in the experiments. However, this does not exclude the possibility that Pearson's correlation coefficients could work well for some large datasets.

Robust measures for correlations (e.g. Kendall's tau) should be the preferred measure for ranking numeric features in feature selection for purposes of estimating the class-feature and feature-feature correlations, when the expense of removing outliers becomes prohibitive. Given a *d*-dimensional instance space and a sample size of *n* instances for correlation measurement, computation of Kendall's *tau* (and generally any robust correlation measure) has a larger time complexity of $O(d.n^2)$ compared to Pearson's $O(d.n)$ for the computation of class-feature correlations. The computation of feature-feature correlations has a time complexity of $O(d^2.n^2)$ for Kendall's *tau* (and generally any robust correlation measure) and $O(d^2.n)$ for Pearson's correlation coefficients. The extra computation time is worthwhile, since it allows more accurate feature rankings, even when moderately small sample sizes are used for the correlation estimates. If, on the other hand, there is sufficient computing power, then the winsorised Pearson's correlation coefficient (Wilcox, 2001) discussed in chapter 3 may be used for correlation measurement. Computation of winsorised Pearson's correlation coefficients will remove the effect of outliers but will not solve the problem of non-linear associations (Wilcox, 2001).

The experimental results further demonstrated that, a correlation coefficient is a random variable in the presence of sampling. This was demonstrated by the results of table 5.5 for the KDD Cup 1999 dataset and in fact Smyth (2001) has discussed this problem. When feature ranking and validation is based on correlations measured for one sample the feature ranking and number of selected features will vary from sample to sample. This was demonstrated in table 5.4. Based on the foregoing observations, feature rankings should not be based on a single sample, but rather on the mean values of the coefficients measured with many samples.

From a statistical perspective, using many (relatively) small randomly selected samples from very large datasets makes it possible to accurately and more efficiently estimate class-feature and feature-feature correlations. This was demonstrated in section 5.3.4. The samples used for feature ranking for large datasets should not be very small. When samples are small, the variability of the correlation coefficients can

change dramatically from sample to sample causing the confidence intervals of the mean correlation coefficients to be wide. It then becomes difficult to trust the feature ranking even when the rankings are based on mean values. This problem is demonstrated in table D.1 of appendix D for the forest cover type dataset samples of size 100.

Probes (fake variables) provide useful information for elimination of irrelevant features. It was empirically found that the three probes used generally eliminated the same (number of) features for the forest cover and KDD Cup 1999 datasets. The Gaussian probe did not work well for the mushroom dataset (all features are qualitative). However, the uniform and uniform-binary probe both eliminated nearly the same number of features for the mushroom dataset. None of the probes eliminated any features for the abalone3C dataset. Probes are random variables. The use of the confidence interval of the mean for a probe provided a better criterion for feature elimination. It was empirically found that probes also selected several features whose correlation values are not of practical significance, as defined by Cohen (1988). However, these features were found to have a small amount of predictive ability for the forest cover type dataset. Statistical significance with the t-test for means selected features that have predictive power. However, for all datasets the t-test eliminated features with no practical significance, even though these features have a small amount of predictive ability.

5.6.2 Feature subset selection

Feature selection methods that search for the best subset of features depend on an initial ranking of features. If this ranking is not accurate, then the search method will not select the best subset of features. The experimental results of section 5.3.4 demonstrated that the use of mean values of correlation coefficients for feature ranking and validation provided more accurate input values for feature subset selection. The experimental results of section 5.4.1 demonstrated that a mathematical function (merit measure) will not necessarily always precisely reflect the definition of what is required for feature subset selection. It was demonstrated that the search procedure can select irrelevant features in preference to more relevant features. The use of decision rules in place of a merit measure provides an alternative method of implementing the definition of feature relevance and redundancy to a search algorithm. The experimental results of section 5.4.2

demonstrated that irrelevant features were not selected when decision rules were used in place of mathematical functions.

The results of section 5.5.4 provided evidence to support the observation that apparently redundant features can have predictive power. This was the case for the abalone3C dataset. Cohen (1995:pg 68) has discussed two types of relationships between random variables. A *simple relationship* between two variables *X* and *Y* can be expressed in the form *X influences Y* (or *X* is correlated with *Y*). An *interaction relationship* involves three variables and is expressed in the form X_i *and* X_j *in concert influence Y*. The feature selection methods proposed in this chapter are not capable of detecting feature interactions.

5.6.3 Problems associated with the global measurement of correlations

For the empirical studies reported in section 5.4 some of the eliminated features were those features that are either good predictors for one or more of the classes in the dataset, or good predictors of some local areas of the instance space, or both. It was observed that one or more of the eliminated features for the KDD Cup 1999 dataset were those features that could be good predictors for the minority and severely under-represented class (U2R). It was hypothesised that if a large dataset is pre-processed to create clusters prior to feature selection then the above problems should not arise. The study of this hypothesis was left for future work.

5.7 Conclusions

The first research question that was addressed in this chapter was: *How can class-feature correlations be measured in order to produce a reliable ranking of features for a dataset?* The method that was studied and demonstrated to work well is to use many samples to measure correlations coupled with a robust measure of correlation. The samples should be large enough to avoid large variability in the measured correlation values as discussed in section 5.6.1. The mean values of the correlations should then be used to conduct validation and feature ranking for the prediction task.

The second research question that was addressed in this chapter is: *What methods of validation for feature correlations result in reliable feature selection?* The experimental results reported in this chapter have demonstrated that a comparison of mean values of class-probe and class-feature correlations provides useful information for more accurately determining which features have no relevance to the classification task. A second method of validation that was studied is the use of Student's t-test of means to determine the practical significance of class-feature correlations. The experimental results indicated that the t-test method of validation eliminated several features which have predictive power.

The third research question that was addressed in this chapter is: *How can domain-specific definitions of feature relevance be incorporated into feature selection procedures?* The method that was studied was to incorporate domain-specific definitions of the meaning of *insignificant*, *low, medium* and *high* correlation, in terms of the ranges of values that should be interpreted as *insignificant*, *low, medium* and *high* correlations. A new algorithm was designed, implemented and used for the selection of the best subset of features. The algorithm used decision rules based on the definitions of values for *insignificant*, *low, medium* and *high* correlations based on Cohen's (1988) definition. Experiments using the decision rule-based algorithm demonstrated that the algorithm selected good feature subsets which have global predictive power. The experimental results have also demonstrated that selecting features based on the measurement of class-feature correlations for samples obtained from all the regions of the instance space does not necessarily result in the selection of all the features that have predictive power. This problem was left for future research. The next three chapters provide a discussion of the studies that were conducted for training dataset selection.

Chapter 6

Methods for Dataset Selection and Base Model Aggregation

> 'Where you lead me I will follow, where you lead me I will follow, wherever you lead me I will follow. I will be with you always. Ngiyakuthanda moya oyingcwele…' (Benjamin Dube, 2007)

It was stated in chapter 2 that a limited amount of research on the combination of dataset partitioning, sampling and aggregate model construction from large datasets has been reported in the literature. To the author's knowledge, only one research effort by Chan and Stolfo (1998) has been reported. Chan and Stolfo's (1998) studies were aimed at improving predictive performance of 2-class datasets with skewed class distributions. It was argued in chapter 2 that, when large datasets are available, training datasets can be designed to achieve bias and variance reduction of the prediction error, without having to re-use training data. It was also argued in chapter 2 that more information is made available to the modeling process when a large amount of data is used in the training process.

The purpose of this chapter is to present the two proposed methods for combining dataset partitioning, sampling and aggregate model construction for large datasets. The methods used in the experiments for the evaluation of aggregate model performance are also presented. The proposed methods are specifically aimed at multi-class prediction tasks. The proposed methods were designed to support two types of base models: One-Versus-All (OVA) models and positive-Versus-negative (pVn) models. OVA modeling (Ooi et al, 2007; Rifkin & Klautau, 2004) is discussed in this chapter and the performance evaluation of this method is presented in chapter 7. pVn modeling is a new method of aggregate model construction, proposed in this thesis. pVn modeling is introduced in this chapter and the performance evaluation of this method is presented in chapter 8. The main difference between OVA and pVn modeling is that each OVA base model is designed to predict one of the *k* classes while each pVn base model is designed to predict more than one of the *k* classes.

It is claimed in this thesis that the proposed methods have the potential to provide a high level of predictive accuracy through the implementation of highly diverse and competent base models that are designed to provide high predictive performance when combined into an aggregate model. Syntactic diversity and high expertise of base models were discussed in chapter 2. Syntactic diversity refers to level of structural differences between the base models that constitute the aggregate model. High expertise refers to the level of predictive accuracy of the base models. The higher the predictive accuracy the higher the expertise. In the context of having large amounts of data available for the modeling process, the methods presented in this chapter are aimed at providing answers to the following questions:

1. How should training datasets be designed in order to create base models that are syntactically diverse and highly expert at prediction for aggregate models?

2. How should training datasets for the base models be designed in order to achieve high accuracy for the aggregate model?

The rest of this chapter is organised as follows: Problem decomposition for OVA and pVn modeling is discussed in section 6.1. A recap of methods for improving predictive performance is given in section 6.2. Methods for training and test dataset selection are discussed in section 6.3. Methods for creation and testing of OVA and pVn models are presented in sections 6.4. Section 6.5 provides a summary of the chapter.

6.1 Problem decomposition for OVA and pVn modeling

Problem decomposition is the process of converting a classification task into several classification sub-problems (Ooi et al, 2007; Dietterich & Kong, 1995). It was stated in the last section that problem decomposition has the potential to reduce the bias component of the prediction error (Dietterich & Kong, 1995). The two methods of problem decomposition that were studied are discussed in this section. The methods are *One-versus-all (OVA)* classification and *positive-versus-negative (pVn)* classification. OVA classification is discussed in section 6.1.1 and pVn classification is discussed in section 6.1.2.

6.1.1 Problem decomposition for OVA modeling

OVA classification (Ooi et al, 2007; Rifkin & Klautau, 2004) is a method of classification where a *k*-class prediction problem is decomposed into *k* sub-problems for classification. OVA classification is commonly used for (binary) support vector machines (SVMs) (Boser et al, 1992) for creating classifiers from multi-class datasets. Given a classification problem with *k* classes, $c_1,\ldots,c_k$, OVA classification involves the creation of *k* sub-problems $ova_1,\ldots, ova_k$. For each sub-problem, ova_i, the task is to create a base classifier, OVA_i, that differentiates between instances of class c_i and instances that belong to all the other classes. In other words, each base classifier specialises in the prediction of one class. The base classifiers, $OVA_1,\ldots, OVA_k$, are combined into one aggregate model using the method of parallel aggregation that was discussed in section 2.2 of chapter 2.

OVA classification was selected as one of the problem decomposition methods to be studied for the following reasons: Firstly, OVA classification enables the creation of base models where each base model is an expert on classification for one specific class. Secondly, since each OVA classifier solves a 2-class problem, the training sample size required to achieve a high level of accuracy is reduced. This is an implication of the Probably Approximately Correct (PAC) learning theory which was discussed in section 2.4 of chapter 2. Equation (2.1) of section 2.4 specifies the theoretical relationship between the samples (complexity) size *n*, classification accuracy $1-\varepsilon$, and hypothesis space size |*H*| . For a fixed level of classification accuracy, reduction of the hypothesis space size |*H*| results in a reduction in the samples size required to achieve a given level of classification accuracy.

6.1.2 Problem decomposition for pVn modeling

Positive-Versus-negative (pVn) classification is a proposed modification of OVA classification proposed in this thesis. For pVn classification, each base model specializes in the prediction of a subset of classes, instead of just one class, as is the case for OVA classification. For *pVn* classification, a *k*-class prediction problem is

decomposed into a set of sub-problems, pvn_1..., pvn_l , (*l*< *k*). For each sub-problem, the task is to create a base classifier pVn_i which specializes in the prediction of *j* classes (*j* < *k*), which are a subset of the *k* classes. The *j* classes are referred to as the *positive* classes. All the other classes whose instances are included in the training dataset for the pVn model are collectively referred to as the *negative* classes. The name *positive-Versus-negative (pVn)* was used to represent the fact that a pVn base model can predict the positive classes in contrast to other classes which are simply treated as negative classes. The *pVn* models are combined into one aggregate model using the method of parallel aggregation.

The initial motivation for *pVn* modeling was as follows: If a multi-class problem has many classes, then many OVA base classifiers must be created. If on the other hand, each of the base models is specialized on more than one class, the number of base models to be created is reduced. After the OVA and pVn modeling experiments reported in chapters 7 and 8 were conducted, it became clear that pVn modeling solves other problems which are discussed in chapter 8.

6.2 Methods for improving predictive performance

The methods for improving predictive performance were discussed in detail in chapter 2. A summary of these methods is given in this section, and details are provided for the objectives for the methods that were studied for training dataset selection. Section 6.2.1 provides a discussion of the methods for bias and variance error reduction for small datasets. Section 6.2.2 provides a discussion of the methods for bias and variance error reduction for large datasets. High competence and syntactic diversity are discussed in section 6.2.3.

6.2.1 Reduction of bias and variance errors for small datasets

It was stated in chapter 2 that the three major factors that affect the predictive performance of a model are the bias, variance, and intrinsic error components of the prediction error. The bias of a predictive model reflects the error in the estimation process for the model (Giudici, 2003; Friedman, 1997; Geman et al, 1992). The variance reflects the sensitivity of the predictive model to the sample used to create

the model (Giudici, 2003; Friedman, 1997; Geman et al, 1992). The intrinsic error is the irreducible component of the prediction error. Various methods of bias and variance reduction were discussed in detail in section 2.11 of chapter 2. For small datasets, bias and variance reduction have been achieved primarily through two methods. The first method involves the creation of many base models through the re-use of the training data either through bootstrap sampling (Breiman, 1996) or re-using those training instances that are difficult to predict (Freund & Schapire, 1997). The second method involves the use of many base models, each with a different structure, in order to achieve syntactic diversity (Ho, 1998; Ali & Pazzani, 1996; Krogh & Vedelsby, 1995; Kwok & Carter, 1990; Hansen & Salamon, 1990). Additionally, various methods for base model aggregation have been studied (Ooi et al, 2007; Sun & Li, 2008; Ali & Pazzani, 1996).

6.2.2 Reduction of bias and variance errors for large datasets

The studies reported in the next two chapters were aimed at the design of aggregate models and the selection of training datasets for the base models, with the objective of reducing the bias and variance components of the prediction error. The training dataset selection methods used for aggregate modeling from small datasets were adapted in this thesis for the selection of training datasets when large amounts of data are available. While the dataset selection methods for small datasets have relied on the re-use of training data, there is generally no need to re-use training data for large datasets, except in those cases where one or more classes are severely under-represented in the dataset. In such cases, bootstrap sampling was employed for the studies of chapter 7 and 8, to increase the number of instances for the severely under-represented classes.

The following methods for bias and variance reduction were incorporated into the base model design and training dataset selection for the studies reported in chapter 7 and 8 of this thesis:

(1) Variance reduction through the use of a different training sample for each of the base models. The objective here was to use as much training data as possible in order to achieve a high level of coverage of the instance space.

(2) Variance reduction through the use of relatively small training samples (relative to the size of the large dataset) for each of the base models. The objective here was to reduce the effects of noise, and chance or phantom structure in the data (Smyth, 2001) as discussed in chapter 2.

(3) Bias and variance reduction through the use of training datasets with a sample composition aimed at providing a high level of coverage of the decision boundary regions of the instance space. The objective here was to provide as much data as possible for the regions where it is difficult to make correct predictions. A second objective was to ensure that the predictive performance does not degrade due to conflicting base model predictions when base models are combined into an aggregate model.

(4) Variance reduction through the selection of good feature subsets. These studies were reported in chapter 5.

(5) Bias reduction through the decomposition of *k*-class problems into 2-class problems as is done for OVA classification, and *j*-class (*j*<*k*) problems which was implemented using the proposed method of pVn classification.

6.2.3 High competence and syntactic diversity of base models

Several researchers have argued that syntactic diversity of base models may lead to a higher level of predictive accuracy for the aggregate model (Sun & Li, 2008; Ho, 1998; Ali & Pazzani, 1996; Krogh & Vedelsby, 1995; Kwok & Carter, 1990; Hansen & Salamon, 1990). Several researchers have also argued that a higher level of predictive performance may be achieved by making each member of the aggregate model as competent as possible (Sun & Li, 2008; Ho, 1998; Ali & Pazzani, 1996). Furthermore, Chan and Stolfo (1998) have demonstrated that the use of carefully designed samples from partitions, and creation of aggregate models from the samples, may result in an increased level of predictive performance of 2-class datasets with skewed class distributions.

It is the author's opinion that syntactic diversity and high competence of the base models, both lead to a reduction in the bias and variance components of the prediction error. Syntactic diversity should lead to a reduction in variance errors

(errors due to sampling variations) since the modeling process is conducted using a large amount of information on the data generating process. Syntactic diversity should also lead to a reduction in variance (sensitivity to the sample used for training), since several samples are used in the modeling process. High competence should lead to a reduction in bias since the methods used in the estimation process of a highly competent model will necessarily minimize the errors in the model estimation process.

For the achievement of syntactic diversity and high competence of the base models, the following methods were incorporated in the base model design and training dataset selection:

(1) Syntactic diversity through the use of base models where each base model predicts a different set of classes.

(2) Syntactic diversity through the use of a training sample with a different composition for the training samples of the other base models.

(3) High competence through the use of base models where each model is specialized on a simpler hypothesis space with fewer classes than for the single k-class model.

(4) High competence through the design of training samples to provide a high coverage of those regions of the instance space where correct prediction is difficult.

6.3 Design and selection of training and test datasets

This section provides a detailed discussion of the method of training dataset selection that were adopted for the experiments of chapters 7 and 8 for OVA and pVn modeling. The methods were designed to achieve three main objectives. The first objective was to maximise diversification of the base models. The second objective was to maximise individual expertise of the base models. The third objective was to ensure that when base models are combined into one aggregate model, the class confusion (occurrence of conflicting predictions) is minimised. Section 6.3.1 provides a discussion of the strategy that was adopted for base model design, training and test data selection, and model testing. Section 6.3.2 provides a discussion of the

motivation for the sample composition of the training and test datasets. Section 6.3.3 presents the methods employed for large dataset partitioning and sampling. Section 6.3.4 provides a discussion of the sampling process from the dataset partitions.

6.3.1 Strategy for dataset selection and model creation

Seven distinct steps were identified for purposes of predictive model design, training dataset selection, model creation, and testing. The steps are shown in figure 6.1. Step 1 involves the design of the base models. Step 2 involves the selection of the relevant feature set for each base model. Step 3 involves making a decision on the partitions that should be created, and then creating the partitions. Steps 2 and 3 could be interchanged. Feature selection is done to ensure that irrelevant features are removed in order to make the individual base models as competent as possible. Step 4 involves the selection of training data and test data. Step 5 involves the creation, validation and testing of each base model. Step 6 involves the combination of the predictions of the base models. Step 7 involves the measurement of the performance gains realized from using an aggregate model versus a single model.

Step 1: Design the base models
Step 2: Select the relevant feature sets for the training datasets
Step 3: Decide on, and create the dataset partitions
Step 4: Select the training datasets and test datasets from the partitions
  a. Create training and test data partions
  b. Create training datasets
  c. Create test datasets
Step 5: Create, validate and test each of the base models
Step 6: Combine the predictions of the base models
Step 7: Measure the performance gain for using an aggregate model versus a single model

Figure 6.1 Steps for dataset partitioning, model creation and testing

6.3.2 Motivation for the sampling methods

It is important to make a decision on the class distribution of the training set and test set samples when sampling is employed. Two alternatives exist: The first alternative is to select samples which have the same class distribution as the large dataset from which the samples are drawn. The assumption here is that the class distribution of

the large dataset is a true representation of the class distribution of the population. The second alternative, called *oversampling* (Berry & Linoff, 2000) is to use a different class distribution in the samples. One common motivation for employing *oversampling* is to increase the coverage of the minority classes that appear in the large dataset.

Berry and Linoff (2000) have cautioned against the use of *oversampling* and argued that *oversampling* changes the meaning of the scores (class posterior probabilities) that are assigned to the predictions by a probabilistic classifier. Recall from chapter 4 that a classification algorithm outputs a prediction for a test or query instance in the form of a pair (*class, score*). Berry and Linoff (2000) have advised that the *lift factor* which was discussed in chapter 4 should be interpreted with care when *oversampling* is used. Provost and Fawcett (2001) on the other hand, have cautioned against the assumption that the class distribution for the large dataset is always a true representation of the population class distribution. Provost and Fawcett (2001) have argued that, firstly, the true class distribution is rarely ever known precisely for most domains. Secondly, the class distribution for a large dataset is subject to change for many application domains. Provost and Fawcett (2001) have provided the example of fraud detection as a domain where the class distribution for large datasets changes often.

Recall from chapter 2 that boosting is a statistical method for modeling which aims to increase the coverage of those regions of the instance space where correct prediction is more difficult. Boosting will necessarily result in changes to the class distribution of the training dataset to make it different from the class distribution of the large dataset. Given the foregoing discussion of Berry and Linoff (2001), and Provost and Fawcett (2001), the author made a decision to use boosted training samples with a class distribution determined by the base model design. Test samples with an equal class distribution for all the classes were used. The motivation here was that the performance of single and aggregate models should be compared class by class for the same number of test instances of each class. The net result of the adopted approach is *oversampling.* For purposes of measuring model performance, calculation of *lift factors* was avoided and ROC analysis was used instead. Recall from chapter 4 that ROC analysis is not dependent on the class distribution of the training and test data.

6.3.3 Partitioning and sampling for dataset selection

The proposed methods of dataset selection involve the use of stratified sampling (Berry & Linoff, 2000; Rao, 2000) in order to obtain the required sample composition for the training and test datasets. Stratification is achieved through the creation of large dataset partitions (strata) with each partition (stratum) consisting of instances of one class. Training datasets are then created by taking random samples from each partition (stratum) with each class having a different level of representation in each of the training datasets. The level of representation of a given class is based on the objectives of model creation. These objectives are further elaborated on later in this chapter. The proposed method of training dataset selection was used to support two different types of aggregate classification models: OVA classification and *positive-Vs-negative (pVn)* classification.

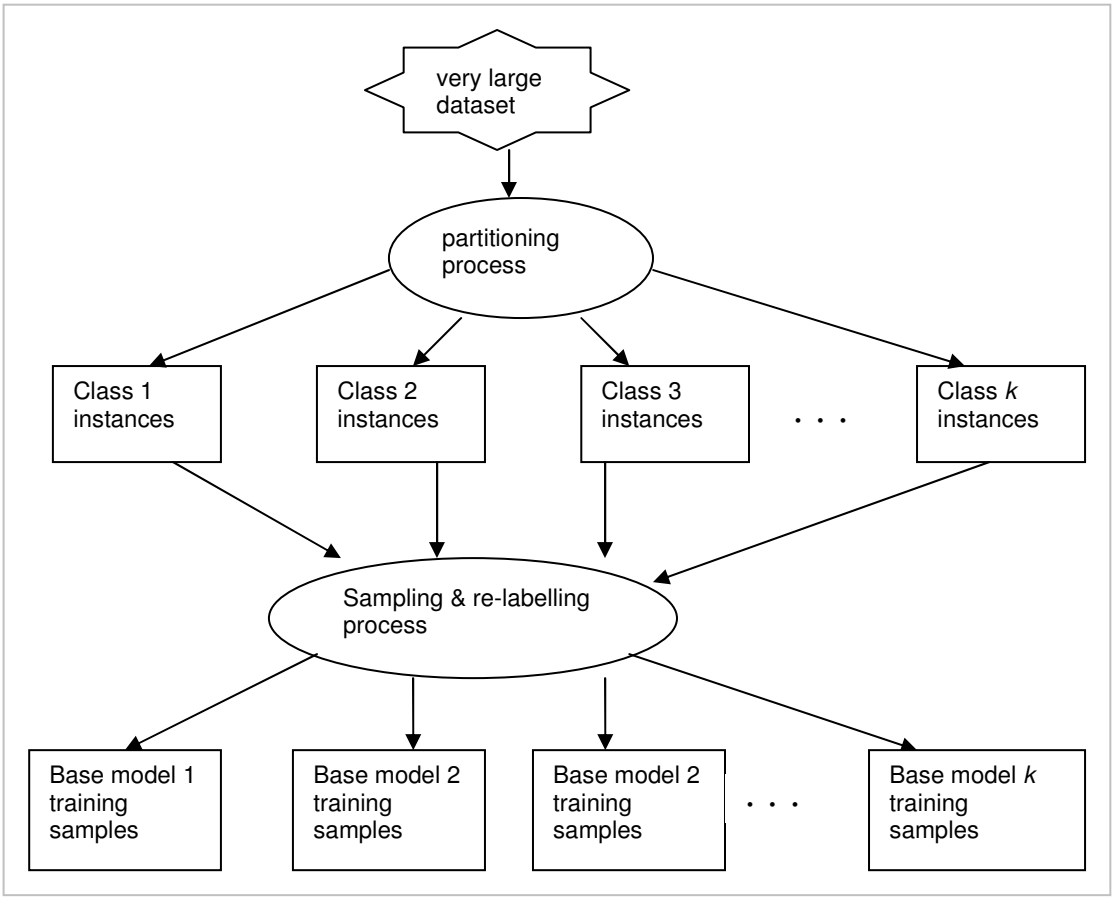

Figure 6.2: Partitioning and sampling process for base model training dataset selection

Figure 6.2 shows the approach that was studied for creating the partitions and obtaining samples from the partitions. This corresponds to steps 3 and 4 of figure 6.1. In step 3 the large dataset is partitioned into *k (k > 2)* partitions, where each

partition consists of instances of the same class. Step 4 involves various activities. The first activity is to sub-divide each partition into a training data and test data partition. The second activity is to create training datasets for the base models by selecting instances from the partition determined by the base model design. The third activity is to create test datasets by selecting instances from each partition. Simple random sampling was used for selecting instances from the partitions.

Each of the model training sets for OVA classification consists of instances from the class it is designed to be expert at predicting, as well as instances from some or all the other classes. Re-labeling was done to assign all the negative instances to a single class, so that each OVA training sample consists of instances of two classes. Each of the base model training sets for pVn classification is composed of instances from the *p* (positive) classes in equal proportions, and *n* (negative) classes in equal proportions. The proportions of the positive and negative samples were different. Re-labeling was done to assign all the negative instances to a single class. Details of how sampling was done are given in the next section.

6.3.4 Sampling from dataset partitions

Samples were taken from the partitions (strata) for the implementation of step 4 of figure 6.1. It is important to make decisions concerning the proportions of instances (of each class) in each of the base model training samples. When one-class partitions are created there may be great variation in the sizes of the partitions, with the partitions for the majority classes being very large and the partitions for the minority classes being very small. The number of training instances required from each one-class partition was calculated and then simple random sampling was used to obtain the instances from that partition. Details of the calculation of the required number of instances are given in chapters 7 and 8. A situation may arise when the partition size is smaller than the required number of instances for datasets with skewed class distributions. When this is the case, the solution that was used in the experiments of chapters 7 and 8 was to obtain a bootstrap sample from the partition (Rao, 2000). Bootstrap sampling (Breiman, 1996; Cohen, 1995) is a statistical method that is used to generate a large amount of data from a small dataset using simple random sampling with replacement (SRSWR) (Rao, 2000). Test data sets were created using simple random sampling from the test data partitions (strata). Each test dataset was created with an equal (balanced) class distribution.

6.4 Methods for creating and testing OVA and pVn models

Steps 5, 6 and 7 of figure 6.1 involve the creation and testing of the base models, aggregation of the base models, and testing of predictive performance of the aggregate models. The methods used to implement steps 5, 6 and 7 are presented in this section. Section 6.4.1 provides a discussion of the implementation of the OVA and pVn base models and a discussion of the outputs generated by the base models. Section 6.4.2 provides a discussion of the methods that were used to implement the aggregate models. The algorithms for model aggregation are presented in section 6.4.3. The experimental procedure for model aggregation is given in section 6.4.4. The methods for measuring performance gains due to aggregate models are presented in section 6.4.5.

6.4.1 Design and implementation of OVA and pVn base models

Base models were designed, created and tested for each dataset. The design objectives discussed in section 6.2 were adopted for each set of base models that make up an aggregate model. The details of OVA and pVn base model design and testing are given in chapters 7 and 8 respectively. Test datasets were created to include positive instances for the class(es) that a base model predicts, as well as negative instances from all the other classes. The same test sets were used for testing all the base models, as depicted in figure 6.3.

A predictive classification model may output a prediction *pred*, for a test or query instance in the form

$$pred = (c_i, conf_i) \tag{6.1}$$

where c_i is the predicted class, $conf_i$ is the level of confidence that the test or query instance belongs to the predicted class and is defined as

$$conf_i = P_r(c_i \mid \mathbf{x}_q) \tag{6.2}$$

where $\mathbf{x}_q$ is the test or query instance and $P_r(c_i \mid \mathbf{x}_q)$ is the posterior probability that the instance $\mathbf{x}_q$ belongs to class c_i. The value of $conf_i$ is referred to as the score that is assigned by the predictive model for purposes of ROC and lift analysis (Giudici & Figini, 2009; Witten & Frank, 2005; Giudici, 2003; Berry & Linoff, 2000).

Each leaf node of a classification tree stores the posterior probability for prediction of each class at that node. The class with the highest posterior probability is the predicted class for test or query instances that land at that leaf node. The *See5Sam* tool which is part of the *See5* classification software (Quinlan, 2004) that was used for the experiments reported in chapters 5, 7 and 8 outputs a prediction as indicated in equation (6.1).

The 5NN classifier which was used for the experiments of chapters 5, 7 and 8, outputs a prediction *pred*, in the form of a triple

$$pred = (c_i, conf_i, recdist_i) \tag{6.3}$$

where c_i is as defined above, and $conf_i$, the probability that the test or query instance belongs to the predicted class is defined as

$$conf_i = P_r(c_i) = \frac{|U|}{5} \tag{6.4}$$

where the numerator $|U|$ represents the count of nearest neighbours that belong to the predicted class and the denominator is the total number of neighbours used for deciding the predicted class (which is 5 for 5NN). The quantity $recdist_i$ is the sum of reciprocal distances for the neighbours that belong to the predicted class and is defined as

$$recdist_i = \sum_{x \in U} \frac{1}{dist(\boldsymbol{x}_q, \boldsymbol{x})} \tag{6.5}$$

where $dist(\mathbf{x}_q, \mathbf{x})$ is the Euclidean distance between the test or query instance and one of the nearest neighbours. The possible values for $conf_i$ are 0.4, 0.6, 0.8 and 1.0 for the 5NN classifier. These values correspond to the number of nearest

neighbours for the predicted (winning) class. For two nearest neighbours of the predicted class $conf_i$=0.4. For three nearest neighbours of the predicted class $conf_i$ =0.6. For four nearest neighbours of the predicted class $conf_i$=0.8. For five nearest neighbours of the predicted class $conf_i$=1.0.

6.4.2 Implementation of OVA and pVn aggregate models

When multiple base models are used, each model will declare a given test or query instance as belonging to a class c_i. The purpose of the aggregation step is to examine all the predictions of the individual base models and select that class with the strongest supporting evidence. The parallel method of aggregation, discussed in section 2.2.5, was used in the experiments. Recall that all the base model predictions for parallel aggregation are considered at the same time and the best prediction is selected based on the level of confidence in the prediction. The methods for combining base model predictions when each base model is capable of predicting any of the *k* classes for a prediction task were discussed in section 2.2.5.

Recall that these methods include (1) *majority voting* (2) the *product rule* (3) the *sum rule* (4) the *max rule*, and (5) the *min rule*. The *product rule* and *sum rule* are not directly applicable to OVA and pVn base models for the following reasons: Since each OVA base model can predict only one of the *k* classes and each pVn base model can predict only a subset of the classes, it is not possible to have a meaningful majority vote for any given class. It is also not possible to generate a meaningful mathematically combined probabilistic score for each class when OVA or pVn base models are used. The *max rule* and *min rule* can however be applied to OVA and pVn base model predictions as discussed below.

When OVA or pVn base models assign a single score to each prediction, as is the case for the See5 algorithm, then the output of a parallel aggregation algorithm, based on the max rule, is a pair defined as

$$pred = (c_i^*, conf_i^*) \in \{(c_1, conf_1), ..., (c_j, conf_j)\},\ j \leq k \qquad (6.6)$$

where

$$conf_i^* = \max\{conf_1,...,conf_j\} \quad (6.7)$$

and *k* is the number of classes for the prediction task, and c_i^* is the predicted class which has the largest value $conf_i^*$. Equations (6.6) and (6.7) are sufficient for determining the prediction for the See5 classification tree aggregate models. The domain of possible values for $conf_i$ is small for the 5NN models, having 0.4, 0.6, 0.8 and 1.0 as the possible values. The small domain of values results in a high probability of tied $conf_i$ values for the base model predictions. In order to break ties, equation (6.5) was used to compute *recdist* values and the tied prediction with the highest *recdist* value was selected as the best prediction for the 5NN aggregate model. The interpretation of the *recdist* values is as follows: If base model predictions have a tied $conf_i$ value, then select the model which used the shortest Euclidean distances to determine the predicted class. The output of the 5NN aggregation algorithm is a triple defined as:

$$pred = (c_i^*, conf_i^*, recdist_i^*) \in \{(c_1, conf_1, recdist_1),..., (c_j, conf_j, recdist_j)\} \quad (6.8)$$

where *pred*, *k*, c_i^* and $conf_i^*$ have the same interpretation as before and $recdist_i^*$ is the reciprocal distance for the best tied or untied prediction. It should be noted that Ooi et al (2007) have used the *recdist* values as a measure of the level of confidence in a 5NN prediction. The problem with Ooi et al's (2007) approach is that *recdist* values do not have a straightforward interpretation for ROC analysis.

6.4.3 Algorithms for model aggregation

The algorithm of figure 6.3 was used to implement the combination (aggregation) decisions for the See5 OVA and pVn aggregate models using the max rule. A base model may predict class c_i or the class 'other' to indicate that a test instance belongs to one of the other classes. The value $conf_i$ in figure 6.3 is the posterior probability $P_r(c_i \mid \boldsymbol{x}_q)$ for the predicted class as defined in equation (6.2) for See5. The value 'none' indicates that there was no valid prediction. That is, all base models predicted

the class 'other'. A base model predicts 'other' to indicate that the class for the query or test instance is not a class that it is designed to predict. In step 3 of the algorithm in figure 6.3, ties are broken randomly, since there is no other value that can be used to resolve a tie.

1. If only one base model predicts a class c_i, and all the other base models predict '**other**', then the prediction is c_i
2. If more than one base model predicts a class c_i, then select the class c_i which is predicted with the largest value of $conf_i$.
3. If there is a tie on $conf_i$ between winning classes then break the tie randomly
4. If all base models predict the class '**other**', then the prediction is '**none**'

Figure 6.3: Algorithm for combining See5 base model predictions

It was stated in section 6.4.2 that the prevalence of tied predictions (tied on the $conf_i$ values) is high for the 5NN base models. The strategy that was used for the implementation of the algorithm that determines all the tied predictions involves the generation of the complete search space of all possible ties. The generation of all possible ties is a combinatorial search problem (Luger & Stubblefield, 1993) requiring the generation of the number of states given by

$$States = \sum_{j=1}^{k} (k-j)^j \tag{6.9}$$

where *k* is the number of classes for the prediction task. For prediction tasks with a small number for classes the combinatorial explosion of equation (6.9) does not pose a major problem. For example, a prediction task with 5 classes will have 22 possible tied predictions. The derivation of equation (6.9) is given in appendix E. Figure 6.4 gives the data structures and algorithms for the functions that were used to combine the 5NN base model predictions.

Data structures:

Prediction: a base model prediction stored as a tuple
(modelname, modelnumber, predictedclass, score, recdist)

State: a search space state which holds data on tied predictions in the form
(tiedcount, tiedmodels, tiedscore, bestclass, bestrecdist)
OPEN,CLOSED, CHILDREN: list used by *BreadthFirstGenerate* algorithm to generate the complete search space

TIED: List used to hold the states for predictions that are actually tied

PREDICTIONS: list to hold the predictions for all the models.

Algorithm for CheckTies( ):

1. Call *BreadthFirstGenerate( )* to generate the list of all states for possible tied predictions consisting of *j, j-1, …, 2* tied predictions. Save the states on the CLOSED list.
2. For each state on the CLOSED list:
   a. Check if there is a tie in the $conf_i$ scores for all the predictions in the state.
   b. If there is a tie, record the tied score, largest *recdist* and prediction with largest *recdist*
   c. Copy the state to the TIED list
3. Delete from TIED every state whose nodes are all contained in another state on TIED
4. Select *besttiedstate* as the state on TIED with the highest score. Break ties using *recipdist*
5. Return *besttiedstate*

Algorithm for CombinePredictions( ):

1. Assign a unique number to each of the *j* ($j \leq k$) base models
2. Store the predictions for the j base models in the PREDICTIONS list
3. If all base models predict the class 'other', then the prediction is 'none'
4. Check for 2-way tied predictions
5. If there are no 2-way ties
   select prediction with largest $conf_i$ score on the PREDICTIONS list as *bestprediction*
6. else
   a. Call *CheckTies( )* to search for the tied state with the largest number of predictions. Call this *besttiedstate*
   b. *besttiedpred* = prediction in *besttiedstate* with largest *recipdist*
   c. select prediction with largest score on the PREDICTIONS list. Call this *bestuntiedpred*
   d. if (score for *bestuntiedpred* > score for *besttiedpred)*
      bestprediction = *bestuntiedpred*
   e. *else*
      bestprediction = *besttiedpred*
7. Return *bestprediction*

Figure 6.4: Algorithm for combining 5NN base model predictions

The function *CheckTies( )* uses the *BreadthFirstGenerate( )* to generate all the possible ties for base models identified as 1,2,..,*j* (*j* ≤ *k*). The *BreadthFirstGenerate( )* algorithm is based on a breadth-first search strategy (Luger & Stubblefield, 1993) and is given in appendix E. For each possible tied state, if there is an actual tie on the $conf_i$ scores for the predictions, the state is recorded in the TIED list. The tied state with the highest score is then selected as the best tie. The function *CombinePredictions( )* places all predictions on the PREDICTIONS list. If all the base models predict the class 'other' then there is no valid prediction for the aggregate model. The function *CombinePredictions( )* checks if there are any 2-way ties (ties involving two predictions). If there are no 2-way ties then there cannot be any 3-way, 4-way or higher order ties. When there are no tied predictions, the prediction with the highest $conf_i$ score is selected as the prediction for the aggregate model. If 2-way ties exist, *CombinePredictions( )* calls *CheckTies( )* to locate the tied predictions with the highest $conf_i$ score. The $conf_i$ score for the tied predictions is then compared with the highest $conf_i$ score for untied predictions. If the tied predictions have a higher $conf_i$ score, the tied prediction with the highest value of *recdist* is selected as the aggregate model prediction.

6.4.4 Experimental procedure for testing aggregate models

The experimental set up for OVA and pVn base model aggregation is shown in figure 6.5. The base models shown in figure 6.5 may be either all OVA models or all pVn models. Ten test sets were used to measure model performance. Each test set was applied to each of the base models and the test (prediction) results were written to a text file. The test results for each test set were combined into a single file and then used as input to the algorithm for combining the predictions of the base model into one prediction for each test instance.

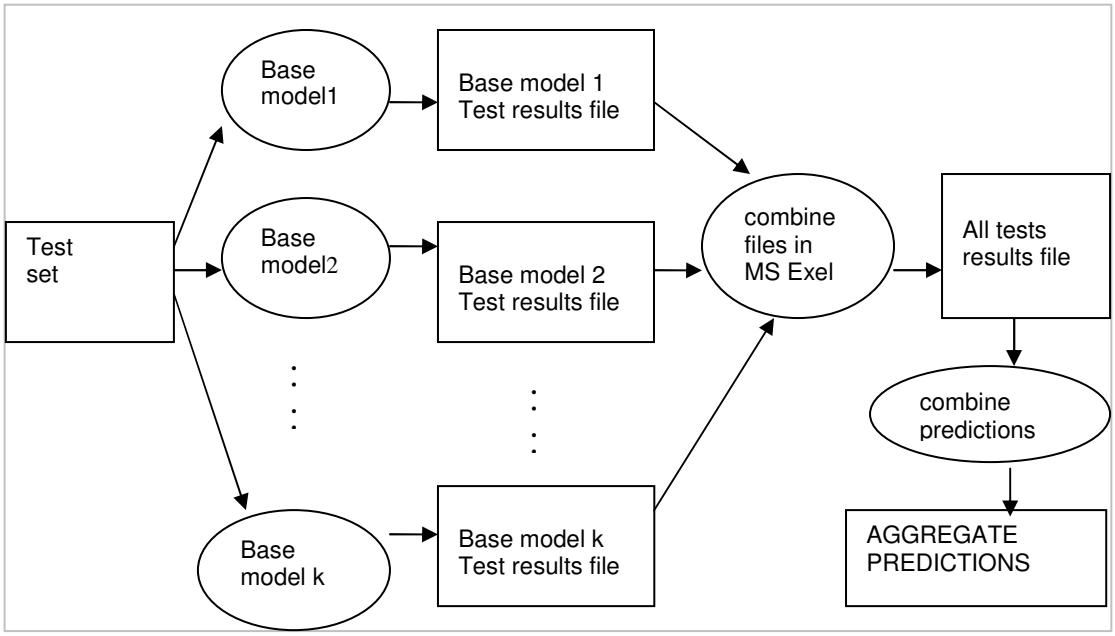

Figure 6.5: Experimental method for aggregate model implementation for one test set

6.4.5 Measurement of performance gains for OVA and pVn aggregate models

Prediction performance gains for aggregate models are typically established through comparison with single models (Ali & Pazzani, 1996). A detailed discussion of the statistical tests used to compare model performance was given in section 4.7 of chapter 4. Given two predictive models, M_A and M_B, Student's paired sample t-test was used to establish whether model M_A provides a higher level of predictive accuracy than model M_B. More precisely, if μ_A and μ_B are the mean values for predictive accuracy for models M_A and M_B respectively, the following hypotheses were tested: $H_0 : \mu_A - \mu_B = 0$ and $H_a : \mu_A - \mu_B \neq 0$. When the null hypothesis is rejected and the mean difference is positive, this gives an indication that the predictive performance of model M_A is generally higher than the performance of model M_B. The mean difference provided by the paired samples t-test, gives an indication of the level of magnitude by which one model is better than the other.

Ali and Pazzani (1996) have conducted studies on different methods of combining the results from various classification models, and have proposed the following measures for computing the error reduction that is realised due to the use of aggregate model:

Measure 1: compute the error difference $error_D$ as

$$error_D = error_S - error_A \tag{6.10}$$

Measure 2: compute the error ratio $error_R$ as

$$error_R = error_A / error_S \tag{6.11}$$

where $error_S$ is the predictive error of a single model, and $error_A$ is the predictive error of the aggregate model obtained from the base models. The larger the error difference, the greater the error reduction due to the aggregate model. The larger the error ratio the greater the error reduction. Ali and Pazzani (1996) have advised that the error ratio is a better measure as it reflects the fact that it becomes very difficult to achieve error reduction using aggregate models when a single model has a very low prediction error. When the mean values of the errors are used in equation (6.10), the equation has a similar interpretation to the mean difference computed by the paired samples t-test.

For purposes of measuring the performance improvements due to the aggregate models, the Ali and Pazzani (1996) measures were re-interpreted by the author of this thesis as shown in table 6.1.

Table 6.1 Interpretation of Ali and Pazzani (1996) measures

Ali & Pazzani Measure	Name used in thesis	Re-interpretation and computation of the measures used in the thesis based on accuracy and TPRATE:	
		accuracy = (1 – error)	*FNRATE = (1 – TPRATE)*
Error difference = $error_S - error_A$	*Diff(A,S)*	$accuracy_A - accuracy_S$	$TPRATE_A - TPRATE_S$
Error ratio = $error_A / error_S$	*Ratio(A,S)*	$\frac{(accuracy_A - accuracy_S)}{(1 - accuracy_S)}$	$\frac{(TPRATE_A - TPRATE_S)}{(1 - TPRATE_S)}$

The measure *Diff(A,S)* represents the performance increase in either the accuracy or TPRATE measures due to the aggregate model. The measure *Ratio(A,S)* represents the fraction (of maximum possible improvement) by which the aggregate model increases the accuracy or TPRATE. A value of *Ratio(A,S)* = 0 indicates that there is no increase in the accuracy or TPRATE. A value of *Ratio(A,S)* = 1 indicates that the

accuracy or TPRATE of the aggregate model is at its maximum value of 1 (or 100%). A negative value for *Ratio(A,S)* indicates deterioration in performance.

Student's paired samples t-test, the *Diff(A,S)* measure, and the *Ratio(A,S)* measure were all used to determine performance improvements due to the aggregate model for the experiments of chapters 7 and 8. Mean values for the accuracy and TPRATE values were used to compute the *Diff(A,S)* and the *Ratio(A,S)* measures.

ROC analysis and lift-factor analysis are commonly used to assess the performance of a predictive classification model and compare different models as discussed in section 4.7.3. It was also noted in section 6.3.2 that lift-factor analysis is difficult to interpret when oversampling is used as was done for this thesis. Multi-class ROC analysis (Fawcett, 2001, 2004, 2006; Provost & Domingos, 2001; Hand & Till, 2001) was used to analyse and compare the performance of the *k*-class single and aggregate models.

6.5 Chapter summary

The methods used for base model design and implementation, dataset partitioning and sampling, training dataset selection, base model aggregation, and performance measurement have been presented in this chapter. The next two chapters report the experimental results of the implementation of these methods for OVA and pVn modeling.