http://waikato.researchgateway.ac.nz/

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

Chapter 1

Introduction

Advances in technology have provided industry with an array of devices for collecting data. The frequency and scale of data collection means that there are now many large datasets being generated. To find patterns in these datasets it would be useful to be able to apply modern methods of classification such as support vector machines. Unfortunately these methods are computationally expensive, quadratic in the number of data points in fact, so cannot be applied to very large datasets.

This thesis considers the problem of reducing large datasets to a size where complex classification algorithms can be run on them without loss of information from those datasets. There are two possible methods that can be employed when reducing a dataset. The first is to select a collection of data points from the dataset as representatives. The other is to create new data points that reproduce key features of the larger dataset using less data points.

This thesis employs the second method. In order to create the new set of data points this thesis proposes a framework whereby commonly used clustering algorithms can be employed to group like data points together. From each of the clusters generated by the clustering methods a new meta-data point can be taken giving a summarised dataset of a size equal to the number of clusters requested from the clusterer.

The simplest means of reducing the size is to take a random selection of points from the dataset. This can however have an adverse impact on the quality of discovered patterns as the data points selected may not correctly represent the underlying relationship present in the dataset. This section describes a proposal for a more intelligent way of reducing the size of the dataset. The *Cluster Classifier* provides a framework where standard clustering algorithms can be used in order to generate a series of meta-data points that represent the data points from the original dataset in a more compact way.

1.1 Motivation

Most classification algorithms perform batch classification, that is, they work on the dataset directly in memory. More complex methods often build large data structures which give them a larger memory footprint. This larger footprint prevents them from being applied to many of the larger datasets. Even when these datasets can be used, the complexity of the algorithm can make the classification task take an inordinately long time. There are solutions to these problems in the form of adding more memory or waiting longer for experiments to finish. However both have a cost in money and time and both may not ultimately solve the problem if the dataset cannot fit in memory when the memory is at a maximum.

A feature of most large datasets is the large number of data points that contain similar information. So one solution is to remove redundant information from the dataset to allow a classifier to concentrate on data points that represent a larger group of points. This allows the construction of a classifier that correctly models the relationship expressed by the data in the dataset. Random sampling is a simple way to remove redundancy and can be very effective on uniformly distributed data. However for other distributions, random sampling provides no guarantees that it is selecting the important data points from a dataset. Random sampling will, in most cases, modify the class distribution. This can be a problem where the class distribution carries useful information. For these distributions, more sophisticated methods are required.

1.2 Data Reduction through clustering

The main idea behind this thesis is to explore the use of clustering to reduce the size of the dataset without losing important information. Clustering is a well established field of statistics and data mining. There are many algorithms in existence for finding cluster relationships within a dataset. A cluster is a group of like data points which share similar properties with each other yet have different properties from other cluster groups. This thesis proposes that by taking the centre of a cluster as a meta-data point to represent the data points that make up that cluster, it is possible to reduce the number of data points in the original dataset while minimising loss of information.

This process is clusterer and classifier agnostic, although a classifier capable of handling weighted instances is desirable. The only requirement on the clusterer is that is provides an option to direct the number of clusters it produces to be close to the size requested by the user. Many clusterers provide this functionality, for example, K-Means, (Hartigan,

1975) where the user requests K clusters based on K cluster centres. In what follows we develop the notion of a meta-classifier, one that embodies the the process of first clustering then classifying. It is referred to as the Cluster Classifier.

1.2.1 Cluster Classification Process

Given a training set the Cluster Classifier performs several pre-processing steps. Data is pre-processed in order to enable the meta-data points produced by clustering to produce a model compatible with the input data points. Supplying an unmodified dataset directly to a clusterer would lead to undesirable results in most cases. For example, only numeric and binary attributes are passed to the clusterers. All nominal attributes are converted to binary as some clusterers cannot handle nominal values. Finally, all numeric attributes are normalised, to prevent different attributes having undue influence on the clustering process, in particular, when used in distance calculations.

The Cluster Classifier requires the number of clusters k to be supplied by the user. The classifier can be used for modelling numeric targets (regression) or nominal targets (classification). For a numeric target class the data is clustered directly. The clusterer is asked to produce exactly as many clusters as specified by the user in k. In a dataset with a nominal target class the class distribution is often uneven with one or two classes making up the majority of the data points while other classes have comparatively fewer data points. Since the clusterer knows nothing about class values it will in most cases cluster data points with different class values in the same clusters. This would result in meta-data points that either have no clear class value, because they incorporate data points from multiple classes, or are modified by and claim support from data points that do not in fact share the same class. It is also quite possible for a clusterer to produce a set of meta-data points where there are no points representing the smaller classes. To resolve these problems the Cluster Classifier separates nominal datasets by class before clustering. So that in a nominal dataset with n classes the clusterer will be called n times.

The clusterer is asked to produce $\frac{k}{n}$ clusters for each class, where k is the total number of clusters requested by the user and n is the number of classes. This brings its own problems. Firstly, it destroys the original class distribution since in the resulting set of meta-data points there will be equal numbers of data points for each class. The data points constructed for the class with greater representation in the original dataset will however have greater weight owing to the greater number of data points supporting them. This will mitigate but not entirely eliminate the problem. The separation also results in more than k clusters being generated in most cases, since $\frac{k}{n}$ will only be a whole number

where k is a multiple of the number of classes. The Cluster Classifier rounds the value up to so that every class is equally represented. The actual number of clusters will be the smallest multiple of the number of classes larger than k.

On occasion, when building the clusters, if there are actually fewer data points than k, then only as many clusters as data points will be generated. This situation may occur in the case of a nominal class, even when the total number of clusters is much smaller than the number of data points, if there is a class represented by less than $\frac{k}{n}$ data points. Usually in this case no clustering will be done, although this is left up to the clusterer to decide, however there is little purpose in clustering when each data point is its own cluster.

Once clustering has been completed the clusters are used to generate the meta-data points. Some clusterers such as K-Means produce cluster centroids that are easily accessible. These centroids are used, by the Cluster Classifier, directly as meta-data points. However if the clusterer produces unweighted centroids or no centroids at all, as is the case for model based approaches such as EM, then the Cluster Classifier can generate them from the cluster predictions. To generate the meta-data point for a cluster the Cluster Classifier takes the mean values of each attribute across all data points assigned to that cluster. The class value for numeric data is arrived at in the same fashion. If a data point is assigned a probability of cluster membership rather than to one fixed cluster, then it adds its weight to those clusters according to the probability with which it is a member. Thus the overall attribute values for a cluster centroid generated this way is the sum of the weights of all contributing data points by the probability of their cluster membership. At this point it is possible to perform some noise reduction by setting a threshold weight for the meta-data points; cluster centres whose weights fall below the threshold have support from few data points, often as few as one. These cluster centres are often indicative of noise in the source data. This is only really applicable for some clusterers, such as Farthest First or K-Means since others do not distribute their clusters in a way that would generate such low levels of support. The resulting set of meta-data points is then ready to be used in a classifier. The full training process is outlined in Figure 1.1.

There are two key assumptions that must hold for this process to work. First, the clusterer must support user specified cluster sizes. This is not optional since the Cluster Classifier needs to be able to ask for a particular cluster size in order to achieve the desired data reduction. Second, the classifier should support weighted data points. While this is not essential it is advisable as the meta-data points are weighted by the number of original data points that were used to create them. If the classifier does not support weighted data

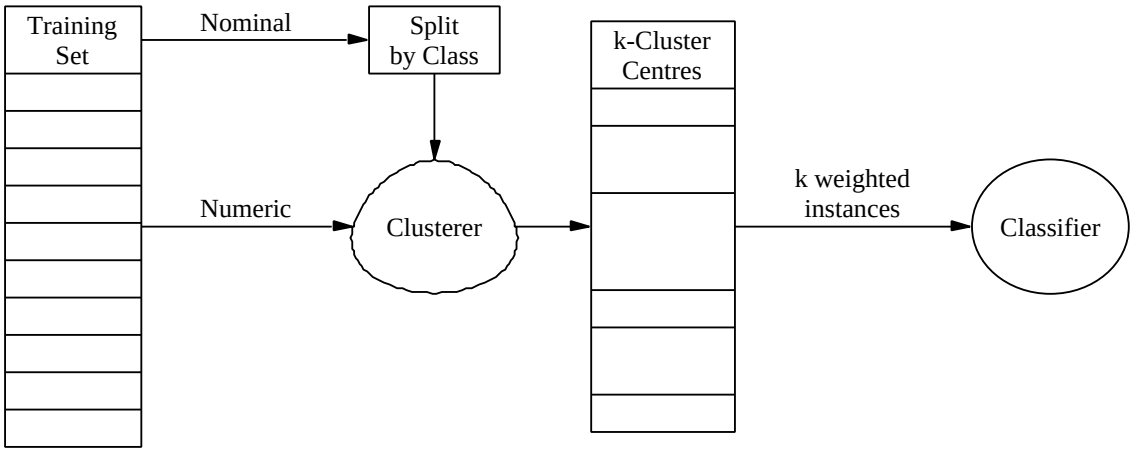

Figure 1.1: **The Training Process**

points then distribution information is lost since the weights encode the support for a centre.

The process for evaluating the effectiveness of a given clusterer and cluster size is similar to evaluating a classifier. This is how the clustered data points would also be used in a production environment. Since the test is evaluating the clusterer for its effectiveness as a means of data summarisation, the measure of success is not overall accuracy but accuracy relative to base classifier performance. A good result for a particular clusterer is maintaining or improving the classification accuracy of the base classifier alone while using fewer clusters than the original number of data points. Testing is done by taking data points held out of the set used for training and passing them through the classifier built from the clustered data points. The process for nominal targets is illustrated in Figure 1.2. Accuracy is measured by how close the class value predicted by the classifier matches the actual class of the data point. For numeric targets the correlation coefficient is computed as a measure of accuracy.

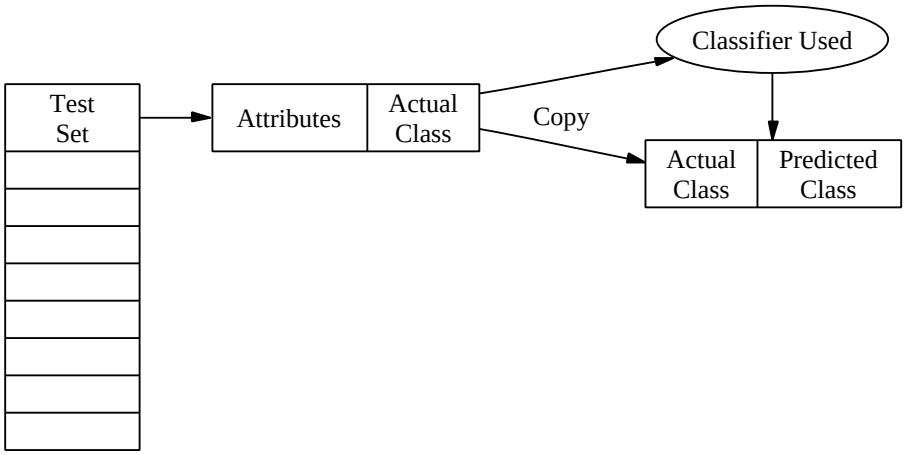

Figure 1.2: **The Testing Process**

Having outlined the framework this thesis then evaluates it's performance on a number of large datasets in order to demonstrate the effectiveness of the method of data summarisation. This is done using multiple clustering methods in order to gauge the potential of the framework independently of any one particular method of clustering.

Chapter 2 describes clustering methods that are used within the framework. Chapter 3 describes an experimental setup which is used to test the framework. Chapter 4 presents the results of experiments, testing the framework on a series of datasets with nominal targets. Chapter 5 presents the experimental results from using this framework on a series of datasets with numeric targets. Chapter 6 presents a summary of the related work in this area. Chapter 7 presents conclusions drawn from the experiments and some future work to extend the method.

Chapter 2

Cluster Methods

The cluster methods described in this chapter all have the key property of forming a user specified number of clusters. While the method does not have to be exact, the clusterer must be able to get close to the requested number. It must also be possible to obtain a meta-data point for each cluster. As mentioned earlier, some clusterers provide a cluster centroid that can be used as the meta-data point, others must have one generated.

2.1 First K

First K is a very naive clusterer focused on speed. This clusterer declares the first K data points encountered to be the cluster centres, each subsequent data point in the dataset is then merged with the closest cluster centre determined by the Euclidean distance. This results in clusters that are certainly not the best clusters that could have been obtained from the data. However, it allows the creation of clusters in a single pass through the data. This gives the clusterer a linear time performance in the number of clusters and data points.

The closeness is determined by measuring the relative squared Euclidean distance between the data point and each cluster centre. The cluster centre is then updated so that each of it's attribute values is the sum of the weight adjusted attribute values of the cluster centre and the data point. For example, if the centre had a weight of three and the data point had a weight of one then the resulting attribute value would be three quarters the value of the centre plus one quarter the value of the data point. The weight of the data point is added to the weight of the centre to create the new centre weight.

2.2 Simple K-Means

This clusterer, described in (Hartigan, 1975), is one of the most basic and commonly used clustering algorithms. Firstly, it requires specification in advance of how many clusters are to be generated. This makes it ideal for use with the Cluster Classifier. The number

of clusters is the parameter K for which the algorithm is named. K points are chosen at random to form the initial cluster centres. All data points are assigned to the nearest cluster by the Euclidean distance. The cluster centre is then recalculated by taking the mean attribute values of all data points in a cluster. The process is then repeated with these new cluster centres until there are no data points reassigned between two iterations. The clusters obtained by this process can be shown to be a local minimum of the sum of the Euclidean distance from each data point to it's cluster centre. There is no guarantee, however, of this result being the best cluster assignment that could have been obtained from this dataset since the result will often fall in a local minimum. The clusters that result from this clusterer are highly dependent on the initial random selection process.

2.3 Farthest First

Farthest first is a variant of K-Means that places each cluster centre in turn at the point farthest from the existing cluster centres (Hochbaum, Dorit S. & Shmoys, David B., 1985). This point must lie within the data area. This greatly speeds up the clustering in most cases since less reassignment and adjustment is needed. In practice this works by taking a data point at random from the dataset as the first cluster centre. Subsequent cluster centres are chosen by taking the data point with the greatest Euclidean distance from all cluster centres already chosen, until K cluster centres have been assigned. The algorithm then proceeds as per K-Means. All data points are assigned to the nearest cluster centre via the Euclidean distance and then the cluster centres are recalculated to put them in the centre of the set of data points that are assigned to their cluster. The data points are then reassigned and if any data point has changed clusters the process is repeated until no further cluster switching occurs. Because of the similarity of this method to K-Means the clusters obtained are often very similar. Placing the centres at the extreme points of the data-space can however result in some clusters that have very little support. This is because noisy datasets will often contain centres that diverge from the main body of the dataset. These data points have a much higher chance of being picked as cluster centres under the Farthest First scheme than under other schemes which would be more likely to filter them out as unreliable.

2.4 Bisecting K-Means

Bisecting K-Means (Steinbach, Karypis & Kumar, 2000) creates bisecting regions in the dataset. It does this by applying the Simple K-Means algorithm with K set to two clusters,

recursively. The operation of this clusterer is very simple, taking the initial set of data points K-Means is used with a K value of two. The larger of the two resulting clusters is then processed the same way. This process is repeated for three clusters and so on until the number of clusters is the same as the K value supplied to Bisecting K-Means. The process of continually dividing by two results in good clusters in relatively short timeframes. The clusters are not guaranteed to be local minima since once a cluster is created by subdividing an existing cluster it is never modified except to divide it into further clusters. In essence there is no way to go back and correct for mis-assignments made early on in the process. Bisecting K-Means, in almost every case, creates vastly different clusters from K-Means. In K-Means and other clusterers it is common to have widely different cluster sizes especially where there are a lot of similar data points in one concentration of a much larger and generally more sparse data space. By contrast, Bisecting K-Means in dividing up this space will create many clusters containing data points that are similar to other data points in nearby clusters. This difference leads to clusters that all have very similar weights and many clusters with centres that are very close to each other whereas the other Euclidean distance based clusterers can often create cluster centres that are more dispersed around the data space with weights that give more significance to those cluster centres in areas with more support. Clusters generated with Bisecting K-Means should be less sensitive to the weights of the data points than other clusterers since the cluster sizes are more uniform.

2.5 Expectation Maximisation (EM)

EM is a probability based clusterer. The method was first described in (Dempster, A. P., Laird, N. M. & Rubin, D. B., 1977), although the methods used had been applied in more specific ways before that. Since for a given dataset the clusterer will not know what the distribution of each cluster is, it takes a similar approach to K-Means generating a random distribution for each cluster and then calculating the probability that each data point comes from that cluster. The cluster models are then recalculated based on the data points assigned to them. The probability of the data points are then reassigned. This process of recalculating first the model and then the probabilities is repeated a set number of times. The calculation of the cluster probabilities is called expectation. The second step in the calculation of the model parameters is called maximisation.

EM is guaranteed to produce a maximum, however like K-Means it will be a local maximum rather than a global maximum. EM also has problems where the number of

clusters is high relative to the number of data points. The chance of a model being randomly generated which has no data points supporting it increases as the number of clusters gets closer to the number of data points. This results in EM producing fewer clusters than specified as these unused models are discarded.

2.6 Clusterer Summary

The clustering methods described above all fit within the requirements set out at the beginning of this section. The resulting clusters from these methods are often dramatically different from each other on the same dataset. Each clusterer has different costs associated with its clustering process. A naive summary of the overall complexity of each method is given in Table 2.1, where n is the number of data points, k is the number of clusters, and i is the number of iterations.

Table 2.1: Clusterer Complexity

Clusterer	Complexity
First K	$O(nk)$
Farthest First	$O(nk^2)$
Bisecting K-Means	$< O(nki)$
Simple K-Means	$O(nki)$
EM	$O(nki)$

First K is the least complex clustering method. This algorithm calculates clusters in $O(nk)$ time. The next most efficient clusterers are the K-Means family of clusterers, Farthest First and Bisecting K-Means which are both optimisations of Simple K-Means designed to reduce its time complexity. Farthest First is the faster of the two, especially for low values of k where $k < i$. Simple K-Means is the pure form of the K-Means algorithm and is more complex, EM is the slowest of the clustering methods due to very large constant factors involved in computing the cluster probabilities. It is less likely that these last two clustering methods would be really useful in the Cluster Classifier as their complexities are very high which could negate the benefits obtained by reducing the dataset.

The complexities in Table 2.1 are very naive. In order to obtain a better bound on the complexities of the more complex methods a way of determining i based on the dataset is needed. This is not important for Bisecting K-Means since i will in not be large for this method. There is some debate as to the overall time complexity of Simple K-Means with (Vassilvitskii & Arthur, 2006) proposing a tighter bound, of $O(n^{O(k)}) * poly(n, \frac{D}{\sigma})$ than the more conservative proposed by (Inaba, Katoh & Imai, 1994) of $O(n^{O(kd)})$. In both of

these proposals d is the distribution of the data. For EM the i value is often fixed at run time since it can otherwise continue for a considerable number of iterations.

Chapter 3

Experiments

This section describes the three sets of experiments conducted to test whether the Cluster Classifier will work with a variety of learners, and the results of those experiments. There are two distinct types of experiments. The first experiment tests a range of clusterers using a variety of classifiers. Testing all possible clusterers and classifiers is beyond the scope of this thesis. The experiments test a reasonable selection of clusterers with a small but representative set of classifiers. Initially smaller datasets and a greater number of clusterers are trialled. The second half of the experiment eliminates the clusterers that were either too slow or too inaccurate on the smaller datasets. The remaining clusterers are then used on larger datasets which provide a better picture of their performance and suitability for use within the Cluster Classifier. The final two experiments test key features of the Cluster Classifier. The first tests the use of unweighted data points or a classifier that does not support weights. This tests the necessity of representing the data points supporting a cluster by adding their weight to the cluster. The second tests the noise reduction effects of clustering. The number of clusters is tested in the first experiment only, as it is core to the Cluster Classifier.

3.1 Experiment Setup

This section describes the setup for all the experiments, the datasets and classifiers used, the experimental method, the gold standard of performance and what would be considered a good result. These things hold for all experiments conducted.

3.1.1 Algorithms

This section describes the four different algorithms we used for testing and why each was chosen. The experiments use two algorithms for each target class type a simple one and a more complex one. These algorithms are listed in Table 3.1. Two algorithms were chosen for each class type because earlier testing had shown that some clusterers worked better with more complex classifiers, while others were better for the simple classifiers.

Algorithms for Nominal Datasets

The experiments using nominal datasets are conducted with two different algorithms: Naive Bayes and Logistic Regression (Witten & Frank, 2005). Naive Bayes was chosen as the simple classifier primarily for its speed and accuracy. Logistic Regression was chosen as the complex classifier because it is relatively parameter free and relatively quick. A key consideration in choosing these classifiers was to have classifiers that operated in a significantly different fashion from each other. These differences enable the classifiers to illustrate the effects of the different styles of clustering used in the experiments. Support Vector Machines were also considered and would work very well within the framework of the Cluster Classifier. However, they require extensive parameter tuning in order to get good performance. Adding this to the long runtimes, even for simple support vector machines, compared to the methods chosen was found to be too expensive for use in these experiments.

Algorithms for Numeric Datasets

The experiments using numeric datasets are conducted with two different algorithms: Linear Regression and M5 Model Trees (Witten & Frank, 2005). Linear Regression was chosen as the simple classifier because it is a fast and well known method of regression that makes use of only simple relations in the data. M5 Model trees were chosen also for their speed and their ability to find more complex relations. M5 Model trees present a problem to the algorithm since they only support instance weights in the leaves not throughout the tree. This could cause problems at higher levels of clustering since more information is encoded in the weights. Support Vector Machines were also considered in the numeric case but rejected for similar reasons to the nominal case, though their support for instance weights makes them a very attractive alternative to M5.

Table 3.1: Algorithms used in testing

	Nominal	Numeric
Simple	Naive Bayes	Linear Regression
Complex	Logistic Regression	M5 Model Trees

3.1.2 Datasets

The experiments use nineteen different datasets, nine nominal and ten numeric. The full list of datasets is given in Table 3.2. These datasets were chosen primarily for their size.

Since the purpose of this thesis is data reduction, running the algorithm on small datasets would not provide any useful results. All datasets are taken from the UCI repository (Blake, Keogh & Merz, 1998). Six of the nominal datasets are real world datasets. The last three are generated datasets. Of these, Agrawal, and both Waveform datasets were generated to forty thousand instances as this was thought to be a reasonable number of instances to process without taking too long. The only difference between the two waveform datasets is nineteen random noise attributes placed at the end of each instance. These attributes are useful in comparing the noise reduction abilities of a clusterer.

Table 3.2: Datasets Used in these experiments

Dataset Name	Size	Attributes	Classes
pendigits	10992	17	10
letter	20000	17	26
mushroom	8124	23	2
opdigits	5620	65	10
hypothyroid	3772	30	4
kr-vs-kp	3196	37	2
Waveform21	40000	21	3
Waveform40	40000	40	3
Agrawal	40000	10	2
2Dplanes	40768	11	Numeric
ailerons	13750	41	Numeric
kin8nm	8192	9	Numeric
house-8L	22784	9	Numeric
mv	40768	11	Numeric
fried	40768	11	Numeric
ColorHistogram	68040	33	Numeric
LayoutHistogram	66615	33	Numeric
CoocTexture	68040	17	Numeric
ColorMoments	68040	10	Numeric

3.1.3 Experimental Base

To provide a baseline for the experiments the classifiers used in the experiments were run over all the datasets. This created a target benchmark against which the performance of the clusterer can be evaluated. These baselines for the nominal datasets are shown in Table 3.3, while Table 3.4 shows the baseline correlation coefficient for the numeric datasets. It is unlikely that the clustered data can get better results than unclustered data. There is an exception in the case where clustering serves as a form of noise reduction. This could occur either by removing poorly supported meta-data points or passively by averaging out erroneous values that would have caused the classifier to misclassify instances. The baseline

and all experiments use ten randomly generated train-test splits of fifty-fifty throughout for consistency, as this was found to be the most practical on the larger datasets. The tables report averages and standard deviations over the ten runs.

Table 3.3: Nominal Baselines

Dataset	Naive Bayes	Logistic
kr-vs-kp	87.716±0.889	97.109±0.365
hypothyroid	95.542±0.306	96.740±0.383
Agrawal	88.451±0.190	67.298±0.002
Waveform21	80.430±0.227	86.607±0.161
Waveform40	80.337±0.130	86.644±0.130
letter	64.077±0.558	77.140±0.244
mushroom	95.362±0.683	99.803±0.597
opdigits	91.085±0.302	93.235±0.845
pendigits	85.640±0.313	95.289±0.187

Table 3.4: Numeric Baselines

Dataset	Linear Regression	M5p
2DPlanes	0.840±0.001	0.974±0.000
fried	0.850 0.002	0.952±0.001
mv	0.902±0.001	1.000±0.000
ailerons	0.903±0.003	0.916±0.002
house8L	0.618±0.007	0.791±0.011
kin8nm	0.640±0.008	0.768±0.020
CorelFeatures-LayoutHistogram	0.161±0.003	0.285±0.011
CorelFeatures-CoocTexture	0.220±0.009	0.301±0.003
CorelFeatures-ColorMoments	0.221±0.005	0.333±0.004
CorelFeatures-ColorHistogram	0.247±0.002	0.287±0.053

Key to Graphs and Tables

Recall that the aim of the Cluster Classifier is to reduce the original dataset size and maintain the accuracy of the classifier. Thus a good result occurs when there is no significant difference between the baseline and the Cluster Classifier. These results are included in bold type throughout the following chapters. A * indicates a result that is statistically significantly better than the baseline. Throughout, we speak of two results for a dataset as being significantly different if the difference is statistically significant at the 5% level according to the corrected resampled t-test (Nadeau & Bengio, 1999).

Chapter 4

Evaluation on Nominal Datasets

This set of experiments evaluates the performance of the Cluster Classifier on nominal datasets. The first two datasets used are hypothyroid and kr-vs-kp. On these smaller datasets all clusterers described in Chapter 2 are used. For the remaining datasets, the more time complex clusterers Simple K-Means and EM are not used since their runtimes are too long to make them effective methods of data summarisation. The purpose is to evaluate the performance of a range of clusterers when used with the Cluster Classifier to reduce a dataset. Since some clusterers perform better with simple classifiers and others are more suited to complex classifiers, the experiments first used a simple classifier, Naive Bayes, and then a more complex one, Logistic Regression.

4.1 Experiments

Each experiment uses a single dataset and one classifier with a variety of clustering methods in order to evaluate the performance of the various clustering methods within the Cluster Classifier framework.

4.1.1 kr-vs-kp-NaiveBayes

Figure 4.1 shows the results for Naive Bayes on the kr-vs-kp dataset. On this dataset without any clustering Naive Bayes produces a result of 85.188±1.157. Figure 4.1 shows that three of the clusterers can summarise the dataset quite significantly but without a statistically significant loss of classification accuracy. Farthest First is the best performing algorithm on this data with no statistically significant difference in performance for a number of clusters greater than 450. As Table 4.1 shows First K and Bisecting K-Means also perform well with 1500 or more clusters. Significantly, Farthest First is the only algorithm to always either match or beat simple Random Selection. EM does very poorly on this dataset.

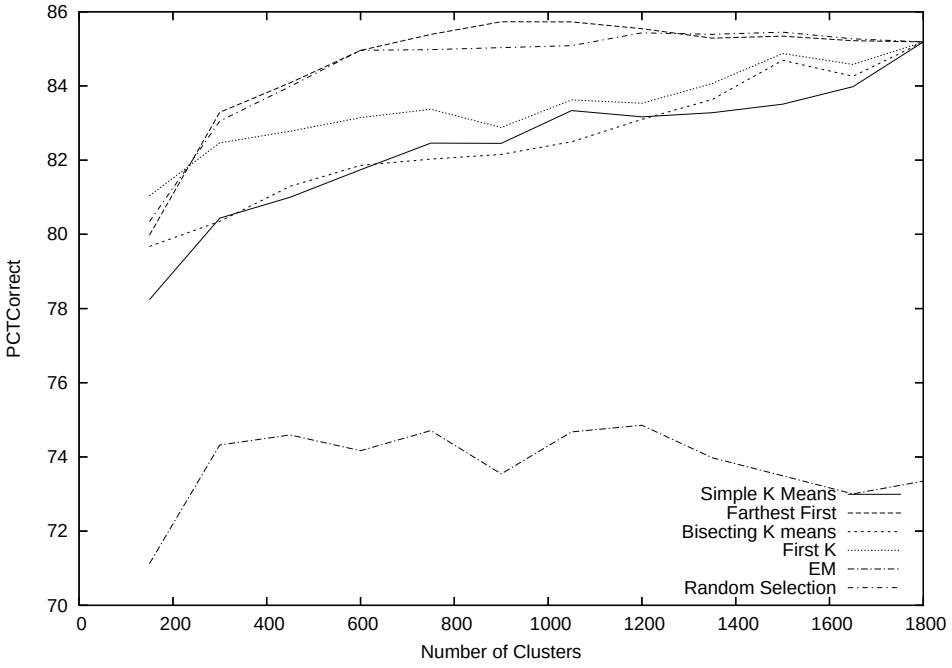

Figure 4.1: **kr-vs-kp-NaiveBayes**

Table 4.1: kr-vs-kp-NaiveBayes

Baseline	Clusters	KM	FF	BI	FK	EM	RN
85.19±1.16	150	78.24±2.59	79.98±1.58	79.67±1.84	81.04±1.65	71.13±2.61	80.34±3.22
85.19±1.16	300	80.44±2.54	83.30±1.20	80.35±2.35	82.47±0.80	74.32±2.28	83.05±2.18
85.19±1.16	450	81.00±2.47	84.09±1.38	81.29±1.94	82.78±1.72	74.59±3.10	**83.99±2.03**
85.19±1.16	600	81.74±3.15	**84.96±1.63**	81.86±1.67	83.15±1.21	74.17±1.29	**84.96±1.33**
85.19±1.16	750	82.46±2.44	**85.39±1.32**	82.03±1.73	83.37±1.59	74.71±1.10	**84.98±1.16**
85.19±1.16	900	82.45±1.80	**85.73±1.01**	82.15±1.62	82.88±1.96	73.54±2.41	**85.03±1.15**
85.19±1.16	1050	83.34±1.28	**85.73±1.14**	82.50±2.59	83.62±1.62	74.67±3.03	**85.09±1.04**
85.19±1.16	1200	83.17±1.65	**85.54±1.12**	83.10±1.50	83.54±1.82	74.86±1.67	**85.43±1.02**
85.19±1.16	1350	83.28±1.75	**85.29±1.03**	83.64±1.30	84.07±1.32	73.98±2.56	**85.39±1.01**
85.19±1.16	1500	83.51±1.58	**85.34±1.01**	**84.69±1.42**	**84.87±1.27**	73.49±3.03	**85.45±1.12**
85.19±1.16	1650	83.98±1.47	**85.22±1.07**	**84.26±1.22**	**84.58±2.53**	73.00±1.46	**85.27±1.04**
85.19±1.16	1800	**85.19±1.16**	**85.19±1.16**	**85.19±1.16**	**85.19±1.16**	73.35±1.35	**85.19±1.16**

4.1.2 kr-vs-kp-Logistic

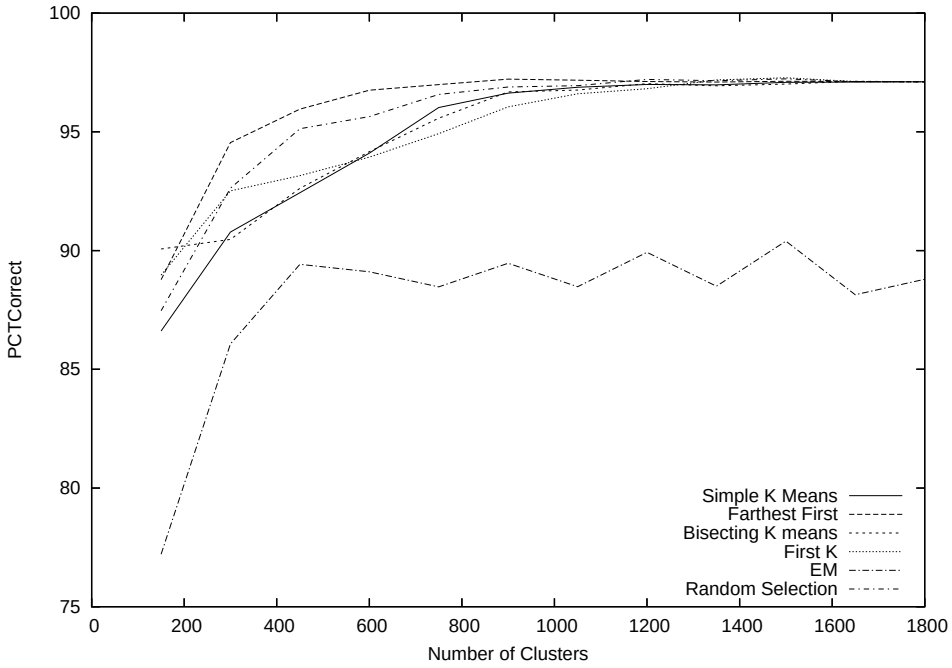

Figure 4.2: **kr-vs-kp-Logistic**

Table 4.2: kr-vs-kp-Logistic

Baseline	Clusters	KM	FF	BI	FK	EM	RN
97.11±0.36	150	86.61±3.01	88.77±2.66	90.07±2.17	88.97±3.26	77.21±5.47	87.47±2.50
97.11±0.36	300	90.78±1.69	94.55±1.38	90.47±1.53	92.52±2.21	86.07±3.15	92.63±1.71
97.11±0.36	450	92.43±1.29	95.96±0.80	92.62±1.15	93.15±1.15	89.42±1.46	95.13±0.86
97.11±0.36	600	94.10±1.04	96.75±0.51	94.17±1.45	93.94±1.05	89.10±1.75	95.64±0.65
97.11±0.36	750	96.03±0.87	**96.99±0.57**	95.58±1.21	94.92±0.96	88.47±2.00	96.58±0.38
97.11±0.36	900	96.63±0.50	**97.22±0.31**	96.68±0.38	96.05±0.68	89.47±2.16	**96.89±0.38**
97.11±0.36	1050	**96.88±0.46**	**97.18±0.37**	96.76±0.40	96.60±0.53	88.48±3.15	**96.95±0.48**
97.11±0.36	1200	**97.00±0.39**	**97.12±0.37**	**97.03±0.32**	96.81±0.29	89.92±2.63	**97.21±0.50**
97.11±0.36	1350	**96.98±0.44**	**97.10±0.40**	**96.94±0.20**	**97.18±0.43**	88.50±2.71	**97.15±0.46**
97.11±0.36	1500	**97.08±0.34**	**97.13±0.43**	**97.01±0.33**	**97.28±0.36**	90.40±1.87	**97.23±0.40**
97.11±0.36	1650	**97.10±0.37**	**97.12±0.35**	**97.13±0.36**	**97.12±0.39**	88.14±1.77	**97.11±0.36**
97.11±0.36	1800	**97.11±0.36**	**97.11±0.36**	**97.11±0.36**	**97.11±0.36**	88.79±1.94	**97.11±0.36**

Figure 4.2 shows the results for Logistic Regression on the kr-vs-kp dataset. For the kr-vs-kp dataset the best performer is Farthest First. Once again both this and Random Selection produce results for all cluster sizes greater than six hundred that are not significantly different than the base classification of 97.109±0.365. All other clusterers with the exception of EM are as accurate as the base classifier after some point, Simple K-Means for cluster sizes greater than nine hundred, Bisecting K-Means at sizes greater

than one thousand and fifty, and First K at thirteen hundred and fifty.

The results for both classifiers are similar to each other, although Simple K-Means and Bisecting K-Means perform markedly better using the more complex classifier. The shape of the graph in Figure 4.2 for less than a thousand clusters, is similar to the overall shape in Figure 4.1

4.1.3 hypothyroid-NaiveBayes

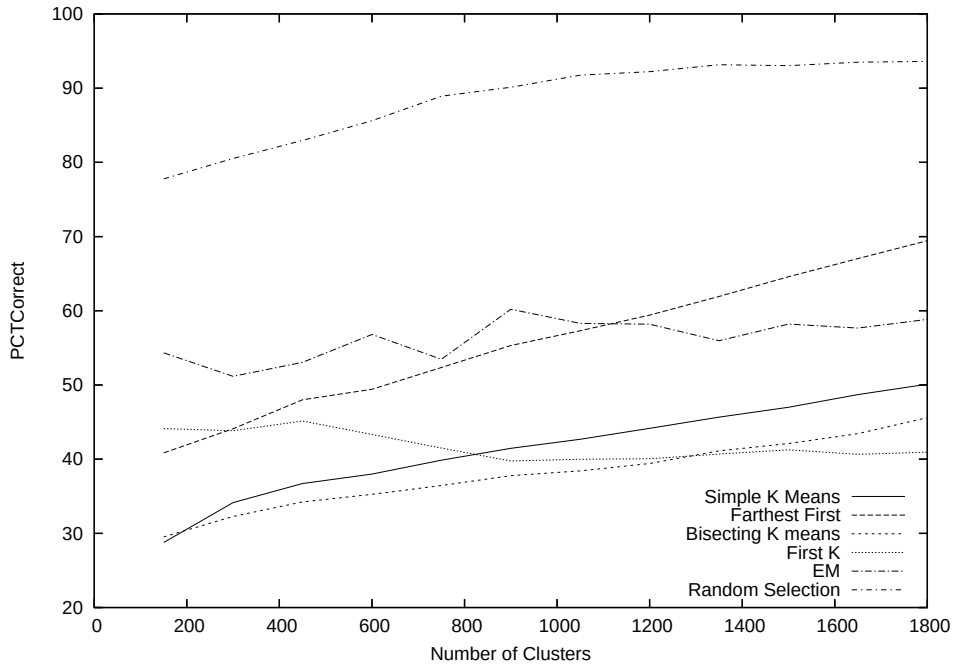

Figure 4.3: **hypothyroid-NaiveBayes**

Table 4.3: hypothyroid-NaiveBayes

Baseline	Clusters	KM	FF	BI	FK	EM	RN
95.27±0.32	150	28.78±4.56	40.85±2.89	29.52±3.26	44.12±8.37	54.32±12.05	77.77±6.68
95.27±0.32	300	34.14±3.35	44.10±2.59	32.27±2.85	43.82±7.19	51.16±10.33	80.52±5.72
95.27±0.32	450	36.70±2.44	48.00±3.43	34.24±3.38	45.13±5.55	53.04±7.48	82.94±4.88
95.27±0.32	600	37.98±2.60	49.41±3.50	35.25±3.27	43.33±6.48	56.80±9.79	85.61±5.00
95.27±0.32	750	39.86±2.62	52.34±3.62	36.44±3.35	41.49±6.50	53.46±10.30	88.90±3.39
95.27±0.32	900	41.46±3.15	55.30±3.51	37.76±3.69	39.75±4.78	60.20±6.70	90.12±2.89
95.27±0.32	1050	42.68±3.14	57.30±3.61	38.41±3.97	39.98±6.00	58.29±8.30	91.75±2.00
95.27±0.32	1200	44.14±3.03	59.40±3.67	39.41±4.04	40.05±6.59	58.18±10.08	92.21±2.23
95.27±0.32	1350	45.67±2.84	61.93±3.84	41.12±4.15	40.68±5.66	55.95±9.04	93.15±1.54
95.27±0.32	1500	47.00±3.29	64.57±3.54	42.11±4.46	41.26±5.30	58.20±6.52	93.02±1.18
95.27±0.32	1650	48.69±3.49	67.02±3.50	43.44±4.53	40.64±5.36	57.67±10.42	93.50±1.01
95.27±0.32	1800	50.10±3.60	69.43±3.09	45.59±4.82	40.93±5.51	58.83±8.28	93.60±1.17

The hypothyroid dataset, as shown in Figure 4.3 reacts very differently to the kr-vs-kp

dataset despite their similar size. This is partially attributable to the unusual structure of the hypothyroid dataset and partially due to the skewed class distribution. Hypothyroid is a four class problem however one class makes up over ninety percent of the data points. This means that for most specified numbers of clusters fewer clusters are generated than intended and the method does not perform as predicted.

The hypothyroid dataset has several attribute pairs consisting of a binary attribute and a numeric one. The binary attribute encodes whether the numeric one was measured or not. If the binary attribute is false then the following numeric attribute will be a missing value. This information is not usable by the clusterers and it is destroyed by even a small amount of clustering. This can be seen in the results in Table 4.3 which shows that with a Naive Bayes classifier an accuracy of 95.542±0.306 can be attained This is closely matched by Random Selection.

Random Selection, which is not a clusterer, does not destroy the relationships of the paired variables. This causes it to degrade slowly as the level of summarisation increases, in a similar fashion to its performance on other datasets. All clusterers do very poorly on this dataset. EM's model fitting approach is least affected although the results are still very poor. Of particular interest for this dataset is the performance of the First K, this clusterer actually performs worst with the least summarisation and improves as summarisation increases.

4.1.4 hypothyroid-Logistic

Table 4.4: hypothyroid-Logistic

Baseline	Clusters	KM	FF	BI	FK	EM	RN
96.72±0.38	150	58.44±8.79	43.58±9.597	63.69±9.19	78.29±5.15	67.06±14.68	81.47±4.14
96.72±0.38	300	88.73±2.59	68.83±18.77	84.29±10.64	78.29±5.59	76.78±7.87	90.66±2.72
96.72±0.38	450	90.73±3.10	70.25±15.49	89.89±5.38	80.82±4.25	73.58±8.37	92.67±2.01
96.72±0.38	600	91.21±2.45	75.63±14.16	92.28±1.93	82.17±4.37	81.32±7.67	93.92±1.43
96.72±0.38	750	91.96±2.65	82.93±10.74	93.68±0.93	85.45±2.22	84.71±4.56	94.61±1.14
96.72±0.38	900	93.24±1.50	85.81±8.59	94.48±1.00	86.97±3.26	83.94±5.76	94.75±1.29
96.72±0.38	1050	93.54±1.29	90.69±6.16	95.24±1.17	90.33±1.95	87.19±3.99	94.88±0.89
96.72±0.38	1200	94.35±1.43	92.50±3.62	95.71±0.99	90.26±1.599	88.34±4.14	95.21±0.92
96.72±0.38	1350	94.59±1.11	94.72±2.11	95.72±0.97	91.45±1.44	88.498±4.95	95.39±0.71
96.72±0.38	1500	95.097±0.93	95.40±1.42	95.82±0.92	91.88±2.47	88.89±3.798	95.59±0.96
96.72±0.38	1650	95.62±1.10	95.67±0.97	96.18±0.78	92.81±1.396	89.04±4.56	95.82±1.05
96.72±0.38	1800	95.97±0.87	**96.28±0.68**	**96.44±0.59**	93.25±1.52	89.54±4.47	95.82±1.07

Figure 4.4 shows the results for Logistic Regression on the hypothyroid dataset. All

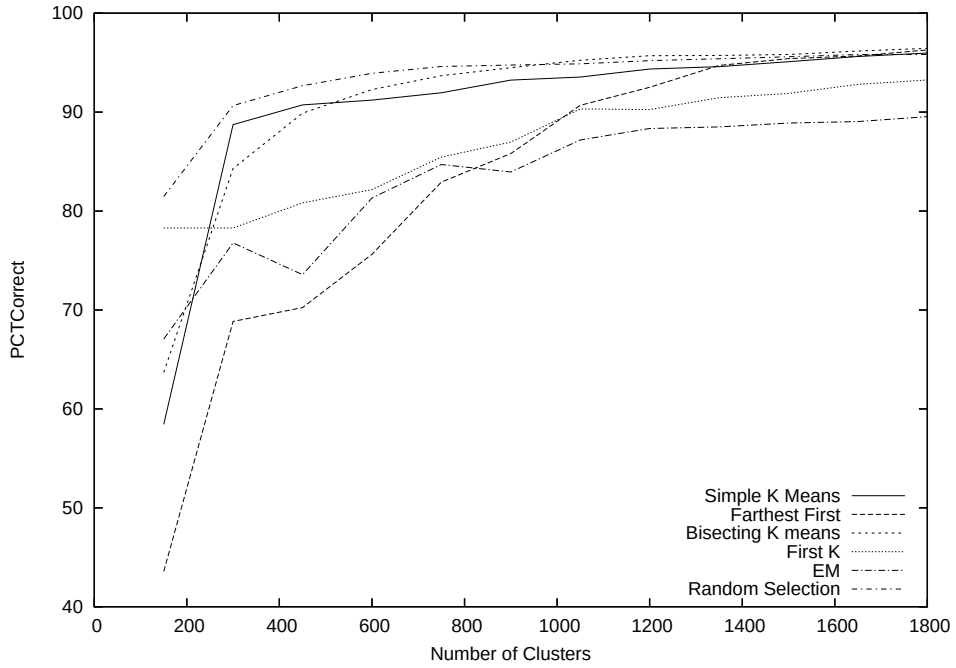

Figure 4.4: **hypothyroid-Logistic**

clusterers perform considerably better than they did using Naive Bayes as the base classifier. However EM shows the least improvement. This experiment shows that random selection because of it's non clustering nature does not suffer as much since it does not destroy the paired variables within the data point.

4.1.5 Agrawal-NaiveBayes

Table 4.5: Agrawal-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
87.780±0.299	2500	**92.249±0.445***	53.229±4.108	**88.985±0.370***	**90.999±1.189***
87.780±0.299	5000	**92.809±0.544***	77.519±1.166	87.297±0.342	**92.576±0.834***
87.780±0.299	7500	**93.019±0.495***	84.683±1.225	86.023±0.403	**93.361±0.758***
87.780±0.299	10000	**92.729±0.300***	**88.714±1.182***	84.966±0.506	**93.687±0.591***
87.780±0.299	12500	**92.278±0.238***	**89.843±1.398***	83.880±0.413	**93.684±0.565***
87.780±0.299	15000	**92.421±0.223***	**90.152±1.307***	83.827±0.559	**93.613±0.472***
87.780±0.299	17500	**92.400±0.179***	**91.278±1.015***	84.112±0.622	**92.863±0.446***
87.780±0.299	20000	**91.767±0.188***	**90.059±0.824***	84.683±0.628	**91.837±0.338***

Naive Bayes on the Agrawal dataset, Figure 4.5, is an interesting example because this is the first experiment that shows a significant performance improvement over the base classifier, as seen in Table 4.5. On this dataset Random Selection actually outperforms the base classifier for any number of selections. First K performs badly but improves to near Random Selection with increased data reduction. This is a reflection of the dependance

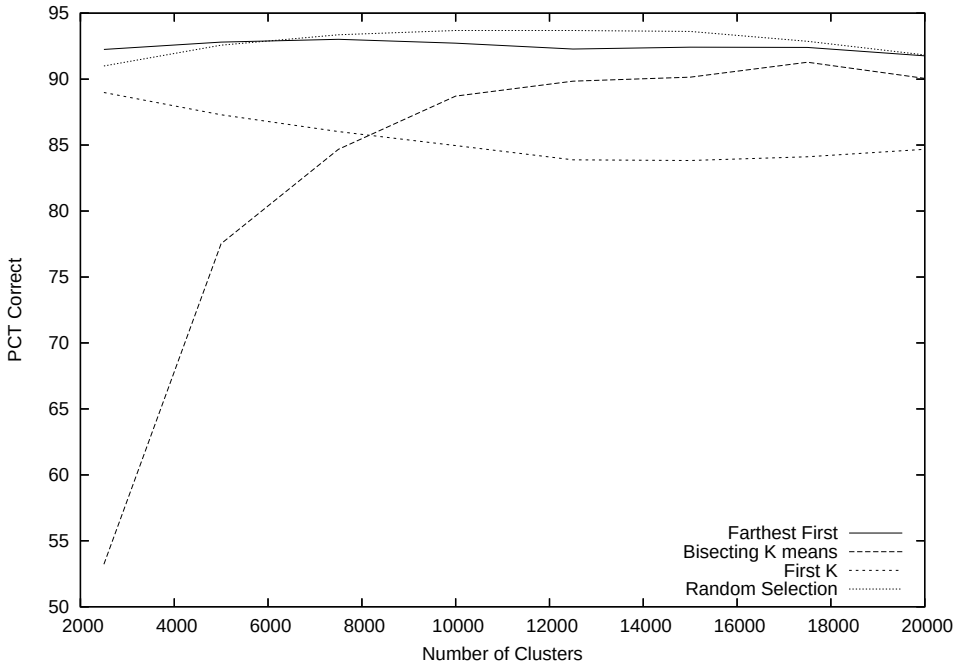

Figure 4.5: **Agrawal-NaiveBayes**

of this particular clusterer on randomisation. Bisecting K-Means has a significantly worse performance for levels of clustering below eight thousand, but for more than ten thousand clusters performs significantly better than the baseline. Farthest First stands out as the most successful method on this data-set, matching Random Selection at most points and always significantly better than the baseline. Of note here are the large standard deviations seen in Table 4.5 relative to the standard deviation of the baseline.

4.1.6 Agrawal-Logistic

Table 4.6: Agrawal-Logistic

Baseline	Clusters	FF	BI	FK	RN
67.298±0.002	2500	48.668±0.737	50.166±1.037	66.814±0.505	49.920±0.725
67.298±0.002	5000	46.824±0.627	50.115±0.488	**67.298±0.002**	49.742±0.503
67.298±0.002	7500	47.218±0.529	50.101±0.425	**67.298±0.002**	49.967±0.544
67.298±0.002	10000	47.383±0.480	50.388±0.435	**67.298±0.002**	50.134±0.607
67.298±0.002	12500	47.203±0.255	50.432±0.403	**67.298±0.002**	50.095±0.480
67.298±0.002	15000	57.635±0.493	61.948±1.430	**67.298±0.002**	62.301±1.568
67.298±0.002	17500	66.530±0.351	67.170±0.149	**67.298±0.002**	67.129±0.222
67.298±0.002	20000	**67.296±0.006**	**67.298±0.002**	**67.298±0.002**	**67.298±0.002**

Logistic Regression on the Agrawal dataset produces very different results from those seen with Naive Bayes. As seen in Table 4.6 the logistic algorithm performs much worse than Naive Bayes on this dataset. However the results are much more consistent for this

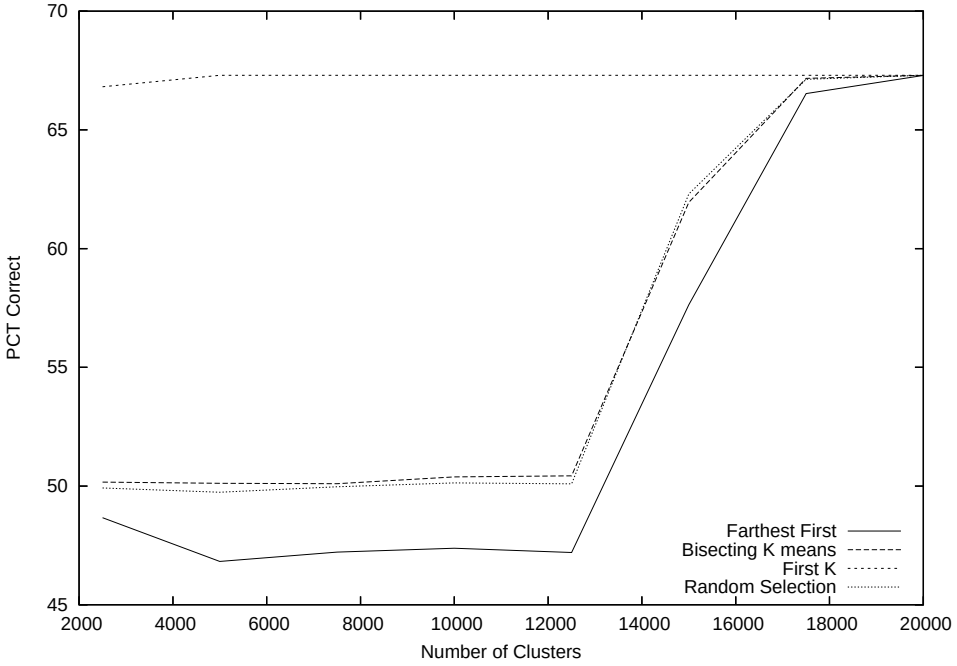

Figure 4.6: **Agrawal-Logistic**

algorithm as seen by the small standard deviations in the Table. Of special note is the almost perfect performance of First K with this algorithm. First K does not deviate significantly from the baseline performance until two thousand clusters, while the other clusterers collapse around 13,000 clusters, and then stabilise again leaving a sixteen percent accuracy difference between themselves and First K as seen in Figure 4.6. Farthest First does noticeably worse than Bisecting K-Means and Random Selection.

Overall First K is the best clustering method on this dataset, achieving ninety percent data reduction on one classifier and seventy five percent on the other. Farthest First and Random Selection are much better for the first classifier but they are very much worse on the second.

4.1.7 Waveform21-NaiveBayes

On the Waveform21 dataset without the extra noise we have a very interesting occurrence, as seen in Figure 4.7. Random Selection performs as well as the base classifier for all numbers of clusters. Both Bisecting K-Means and First K improve significantly on the base classifier getting more accurate with fewer clusters. This leads to a differential of over two percent at two thousand five hundred clusters. These increases also come with more reliability as the standard deviations for both Bisecting K-Means and First K are smaller than those for Random Selection as seen in Table 4.7 . First K is overall the best performer in this experiment. Farthest First trails off from the baseline as the number of

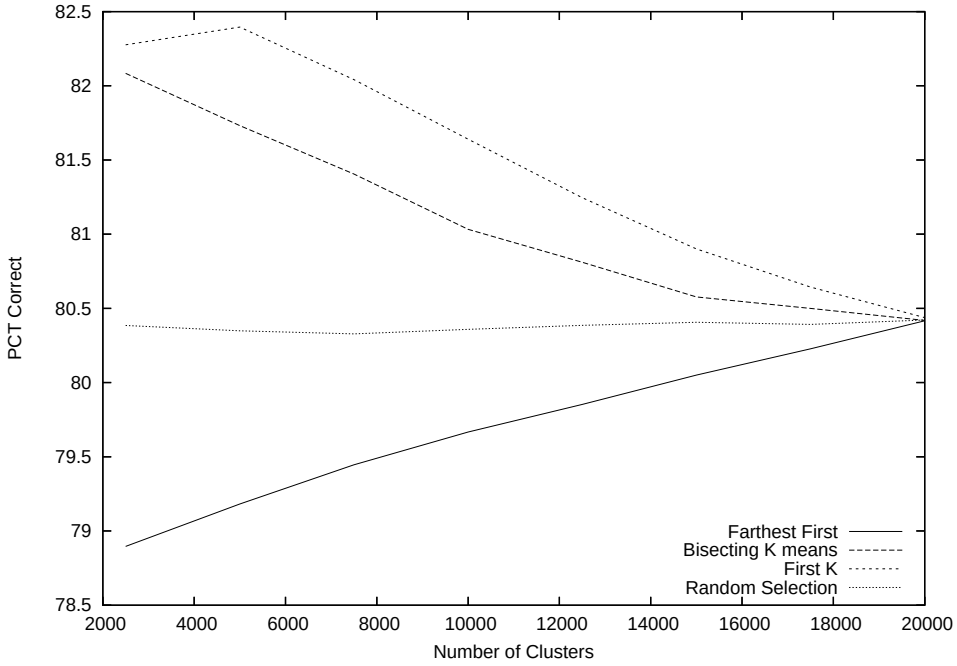

Figure 4.7: **Waveform21-NaiveBayes**

Table 4.7: Waveform21-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
80.430±0.228	2500	78.896±0.253	**82.084±0.223***	**82.276±0.220***	**80.385±0.293**
80.430±0.228	5000	79.182±0.241	**81.732±0.189***	**82.396±0.238***	**80.349±0.258**
80.430±0.228	7500	79.446±0.256	**81.406±0.168***	**82.042±0.244***	**80.328±0.299**
80.430±0.228	10000	79.667±0.241	**81.033±0.222***	**81.640±0.200***	**80.359±0.240**
80.430±0.228	12500	79.853±0.248	**80.811±0.214***	**81.246±0.173***	**80.386±0.247**
80.430±0.228	15000	80.051±0.243	**80.577±0.213**	**80.901±0.219***	**80.407±0.262**
80.430±0.228	17500	80.228±0.235	**80.500±0.221**	**80.644±0.223***	**80.392±0.246**
80.430±0.228	20000	**80.416±0.229**	**80.419±0.224**	**80.438±0.232**	**80.422±0.232**

clusters falls.

4.1.8 Waveform21-Logistic

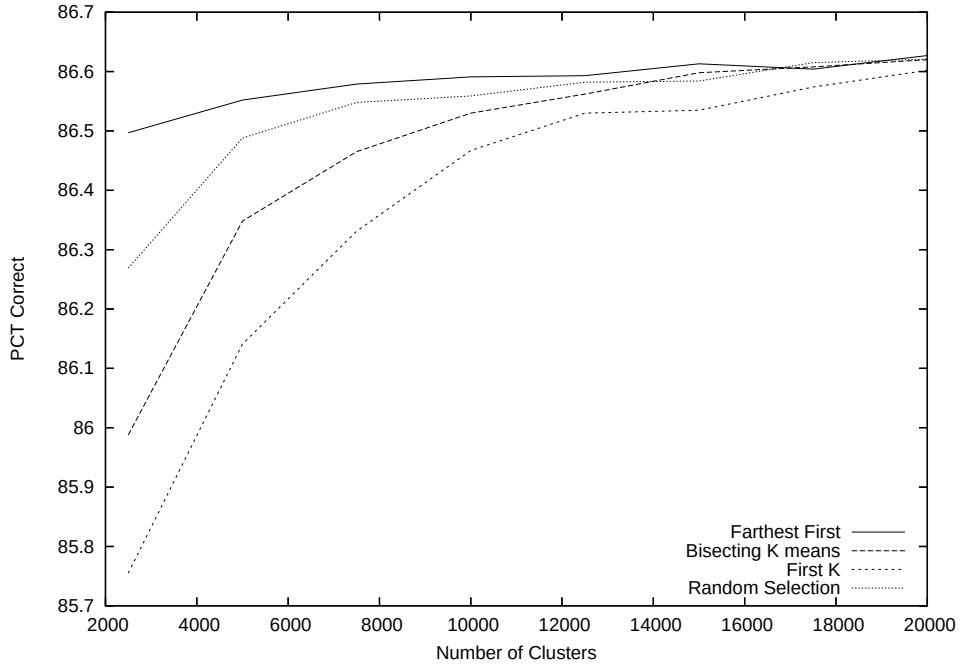

Figure 4.8: **Waveform21-Logistic**

Table 4.8: Waveform21-Logistic

Baseline	Clusters	FF	BI	FK	RN
86.607±0.161	2500	**86.497±0.154**	85.988±0.213	85.755±0.288	86.269±0.203
86.607±0.161	5000	**86.552±0.154**	86.348±0.158	86.141±0.151	**86.488±0.163**
86.607±0.161	7500	**86.579±0.165**	86.465±0.177	86.331±0.181	**86.548±0.164**
86.607±0.161	10000	**86.591±0.187**	**86.530±0.197**	86.467±0.156	**86.559±0.144**
86.607±0.161	12500	**86.593±0.153**	**86.562±0.184**	**86.530±0.160**	**86.582±0.153**
86.607±0.161	15000	**86.613±0.157**	**86.598±0.173**	**86.535±0.165**	**86.584±0.169**
86.607±0.161	17500	**86.604±0.163**	**86.608±0.157**	**86.574±0.169**	**86.615±0.159**
86.607±0.161	20000	**86.627±0.153**	**86.620±0.145**	**86.602±0.162**	**86.621±0.150**

Logistic Regression performs much better than Naive Bayes on the Waveform21 dataset. As seen in Figure 4.8 the rank of the datasets for this classifier is reversed with Farthest First performing best and First K performing worst. However until the 10,000 cluster level no algorithm performs significantly different from any other. Of note is the way all algorithms have similar shape in their performance with Random Selection falling only slightly but still significantly at the two thousand five hundred level. Farthest First is never statistically significantly different from the baseline while the others fall earlier but at a similar rate of change.

In Waveform21 Logistic Regression achieved significantly higher accuracy than Naive Bayes. The different clusterers react very differently to the two methods of classification. Both First K and Bisecting K-Means seem to be averaging out the difference between the classifiers as their accuracy trends towards their accuracy for the other classifier as the number of clusters is reduced. Farthest First shows consistent performance with both classification methods although accuracy is not as badly affected by its clustering when using Logistic Regression, as it is under Naive Bayes.

4.1.9 Waveform40-NaiveBayes

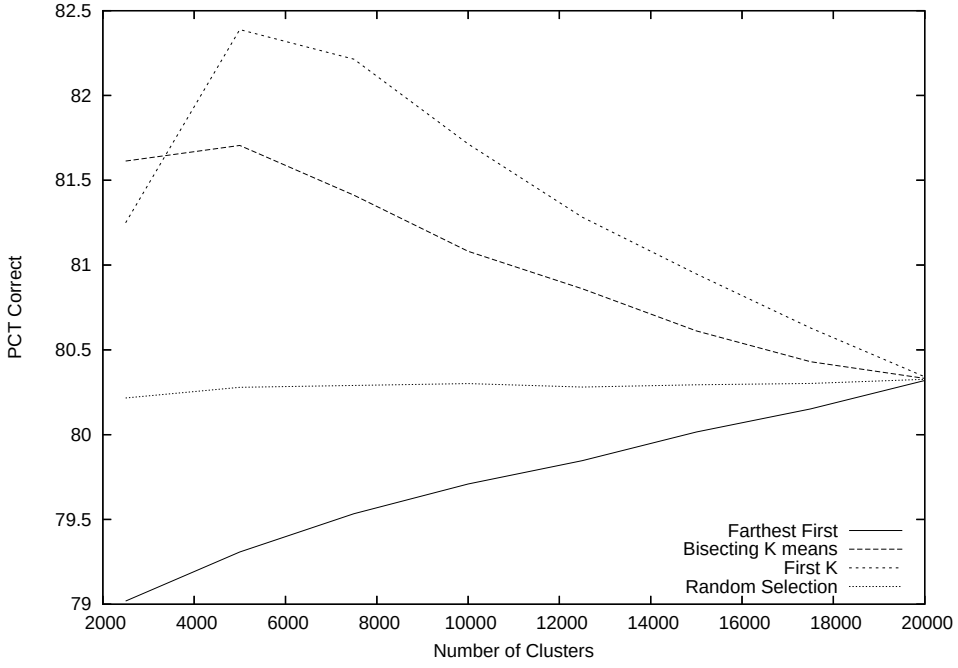

Figure 4.9: **Waveform40-NaiveBayes**

Table 4.9: Waveform40-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
80.337±0.130	2500	79.019±0.238	**81.613±0.308***	**81.249±0.267***	**80.216±0.254**
80.337±0.130	5000	79.308±0.192	**81.705±0.209***	**82.388±0.149***	**80.279±0.156**
80.337±0.130	7500	79.534±0.187	**81.412±0.135***	**82.214±0.115***	**80.290±0.195**
80.337±0.130	10000	79.709±0.169	**81.081±0.101***	**81.714±0.118***	**80.301±0.187**
80.337±0.130	12500	79.847±0.153	**80.860±0.110***	**81.282±0.111***	**80.281±0.161**
80.337±0.130	15000	80.016±0.172	**80.612±0.111***	**80.947±0.107***	**80.294±0.158**
80.337±0.130	17500	80.152±0.149	**80.430±0.114**	**80.629±0.124***	**80.302±0.128**
80.337±0.130	20000	**80.319±0.129**	**80.332±0.130**	**80.341±0.128**	**80.326±0.133**

The shape of the graph shown in Figure 4.9 for the noisy version of the waveform dataset is very similar to the shape seen in the non-noisy form. However the gaps between

the algorithms are more pronounced. Interestingly the standard deviations as seen in Table 4.9 are much smaller for all three clusterers than those shown in Table 4.8 above. With the exception of the two greatest levels of clustering where both Bisecting K-Means and First K produce less reliable predictions. Of note is that these values also reverse the direction of the line from upward to downward which was not seen in the previous example. Also note that Random Selection trends downwards with increased levels of clustering, although by non statistically significant amounts. Farthest First performs similarly on this dataset to the non-noisy version falling away as the number of clusters decreases, only producing a result equivalent to the baseline at 20,000 clusters.

4.1.10 Waveform40-Logistic

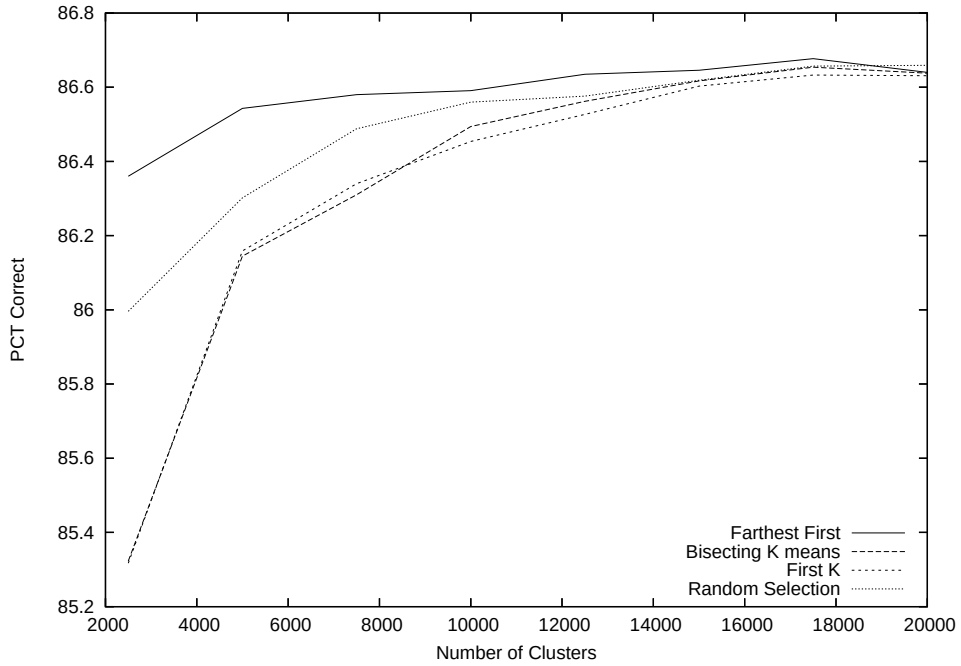

Figure 4.10: **Waveform40-Logistic**

Table 4.10: Waveform40-Logistic

Baseline	Clusters	FF	BI	FK	RN
86.644±0.130	2500	86.360±0.162	85.323±0.249	85.317±0.474	85.996±0.308
86.644±0.130	5000	**86.543±0.148**	86.145±0.196	86.159±0.200	86.302±0.220
86.644±0.130	7500	**86.580±0.144**	86.310±0.158	86.340±0.195	86.488±0.180
86.644±0.130	10000	**86.591±0.183**	86.494±0.149	86.454±0.190	**86.560±0.156**
86.644±0.130	12500	**86.635±0.135**	**86.562±0.159**	**86.527±0.174**	**86.576±0.168**
86.644±0.130	15000	**86.646±0.133**	**86.617±0.147**	**86.603±0.168**	**86.619±0.134**
86.644±0.130	17500	**86.677±0.131**	**86.654±0.142**	**86.633±0.130**	**86.657±0.108**
86.644±0.130	20000	**86.640±0.131**	**86.638±0.124**	**86.631±0.131**	**86.659±0.133**

For the Logistic Regression algorithm the affects of noise on the Waveform40 dataset are more pronounced. Although Random Selection and Bisecting K-Means are more adversely affected than First K. Farthest First only falls significantly at the two thousand cluster level and is the standout performer. Like in the non-noisy version of this dataset all clustered instances do not deviate significantly from the baseline until around 10,000 clusters. At this point the clusterers have the same rank as they maintained for the non-noisy version of the dataset. Beyond this First K and Bisecting K-Means fall together although their accuracy is still never more than one percent worse than Random Selection and never two percent worse than the base classifier.

Waveform40 shows similar results to Waveform21 Again Bisecting K-Means and First K appear to be averaging out differences between classifiers. Farthest First achieves seventy five percent data reduction with Logistic Regression and no reduction at all with Naive Bayes

4.1.11 letter-NaiveBayes

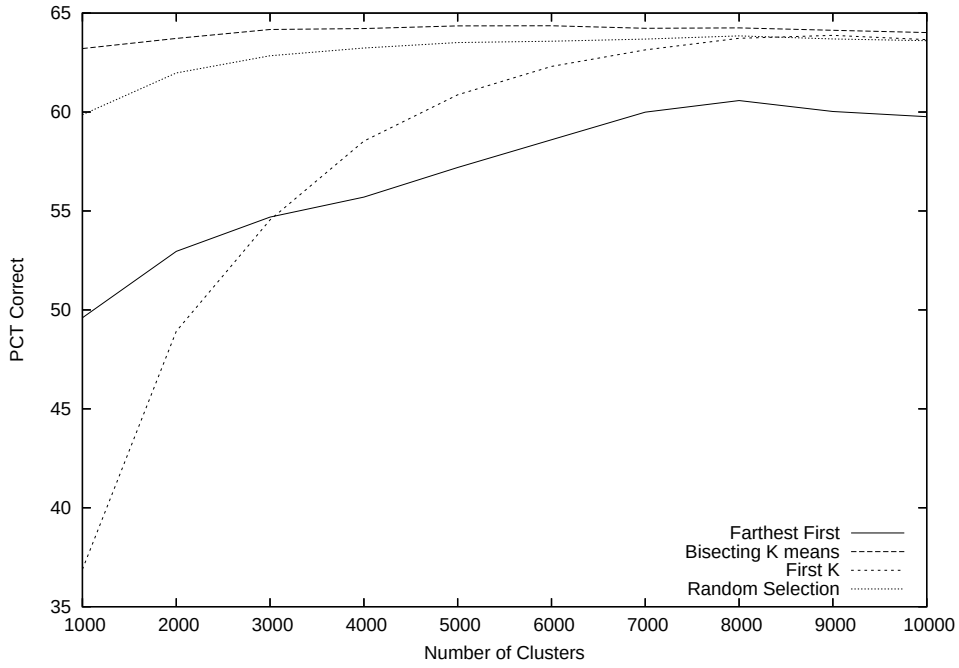

Figure 4.11: **letter-NaiveBayes**

Letter is a twenty six class problem. This is reflected in the poor performance of First K below the seven thousand cluster level. Farthest First does poorly for all cluster sizes on this dataset. Bisecting K-Means performs very well on this dataset, which would seem to indicate that the naive approaches of the other methods have an adverse impact on their performance. In fact, Bisecting K-Means is not significantly different from the base

Table 4.11: letter-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
64.077±0.558	1000	49.605±0.925	63.206±0.660	36.890±2.382	59.865±1.270
64.077±0.558	2000	52.961±0.681	**63.717±0.589**	48.921±1.466	61.977±0.820
64.077±0.558	3000	54.689±0.645	**64.171±0.568**	54.549±1.232	62.842±0.960
64.077±0.558	4000	55.698±0.631	**64.221±0.504**	58.529±1.013	63.234±0.584
64.077±0.558	5000	57.195±0.748	**64.352±0.510**	60.869±0.531	63.514±0.524
64.077±0.558	6000	58.599±0.820	**64.362±0.491**	62.307±0.513	63.579±0.630
64.077±0.558	7000	59.995±0.578	**64.232±0.450**	63.141±0.487	**63.689±0.437**
64.077±0.558	8000	60.581±0.433	**64.250±0.506**	**63.724±0.407**	**63.851±0.425**
64.077±0.558	9000	60.029±0.465	**64.130±0.523**	**63.872±0.512**	**63.691±0.463**
64.077±0.558	10000	59.767±0.497	**64.019±0.510**	**63.662±0.499**	63.608±0.480

classifier performance until the one thousand cluster level. Bisecting K-Means improves on its performance as clustering increases until the six thousand cluster level. This is interesting in that this is the point where First K begins its serious descent. Of note with this problem are the large standard deviations, relative to the baseline, shown in Table 4.11. These standard deviations testify to the difficulties in clustering such a large multi-class problem.

4.1.12 letter-Logistic

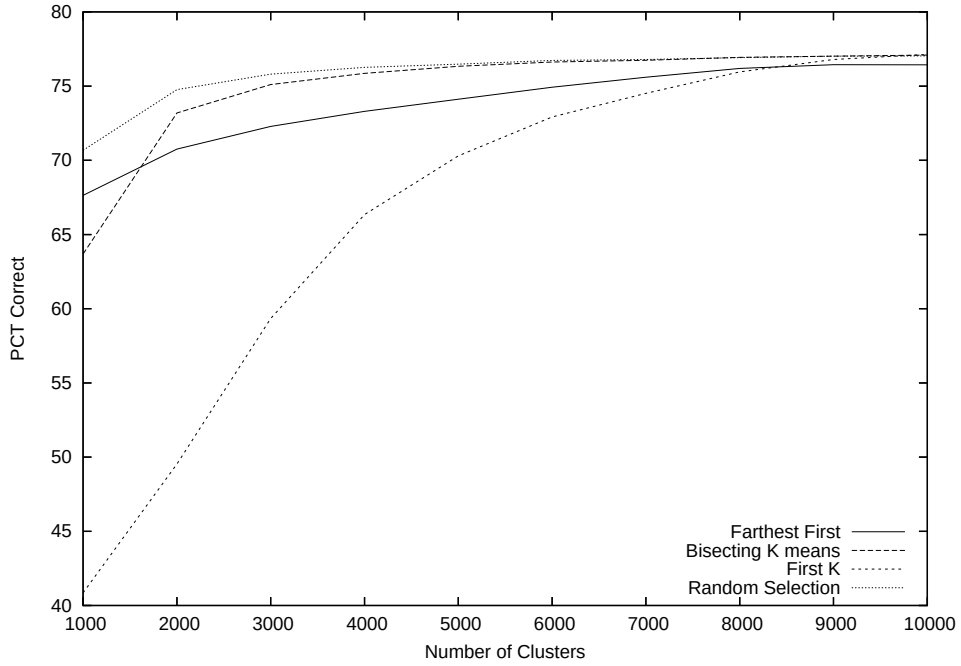

Figure 4.12: **letter-Logistic**

The Logistic Regression algorithm performs much better on a Random Selection of

Table 4.12: letter-Logistic

Baseline	Clusters	FF	BI	FK	RN
77.140±0.244	1000	67.640±0.664	63.684±1.056	40.858±1.210	70.678±0.846
77.140±0.244	2000	70.758±0.236	73.187±0.536	49.533±1.252	74.763±0.461
77.140±0.244	3000	72.286±0.418	75.106±0.378	59.332±1.847	75.807±0.401
77.140±0.244	4000	73.294±0.357	75.860±0.245	66.343±0.975	76.265±0.360
77.140±0.244	5000	74.116±0.299	76.336±0.328	70.294±0.547	76.474±0.354
77.140±0.244	6000	74.921±0.219	76.618±0.299	72.917±0.664	76.731±0.349
77.140±0.244	7000	75.600±0.298	76.739±0.246	74.506±0.605	76.791±0.250
77.140±0.244	8000	76.193±0.323	**76.939±0.264**	75.954±0.474	76.914±0.251
77.140±0.244	9000	76.443±0.241	**77.017±0.271**	76.791±0.231	**77.016±0.309**
77.140±0.244	10000	76.430±0.215	**77.087±0.291**	**77.135±0.235**	**77.035±0.295**

instances from letter than Naive Bayes did. Figure 4.12 shows the that the accuracy of the algorithm using meta-data points generated by Bisecting K-Means is indistinguishable from Random Selection for most cluster sizes. Although with eight thousand clusters it is not statistically significantly different from the baseline while Random Selection is. Bisecting K-Means falls off faster than Random Selection similarly to how it did with Logistic Regression on the Waveform21 and Agrawal datasets. This fall off does not occur until the number of clusters is reduced to two thousand which is indicative of the strength of Bisecting K-Means on this dataset. Farthest First and First K also perform very badly with this classifier and dataset.

Overall Bisecting K Means is the best clustering method for the letter dataset, achieving an average of forty percent reduction in dataset size without significant loss of accuracy. Both Farthest First and First K perform very poorly on this dataset.

4.1.13 mushroom-NaiveBayes

Table 4.13: mushroom-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
92.952±0.396	500	**93.092±0.456**	**93.028±1.281**	**93.929±1.197***	**92.935±0.599**
92.952±0.396	1000	**92.986±0.407**	**92.999±0.981**	**93.227±1.071**	**92.915±0.396**
92.952±0.396	1500	**92.969±0.428**	**93.151±1.094**	**93.247±1.172**	**92.915±0.368**
92.952±0.396	2000	**92.967±0.428**	**93.013±0.467**	**93.326±1.345**	**92.925±0.351**
92.952±0.396	2500	**92.947±0.404**	**92.875±0.555**	**93.075±0.520**	**92.942±0.366**
92.952±0.396	3000	**92.971±0.391**	**92.893±0.529**	**92.922±0.655**	**92.962±0.383**
92.952±0.396	3500	**92.949±0.401**	**92.839±0.538**	**92.868±0.517**	**92.952±0.372**
92.952±0.396	4000	**92.949±0.401**	**92.900±0.412**	**92.942±0.497**	**92.954±0.400**

The mushroom dataset is an interesting challenge since the baseline here is usually

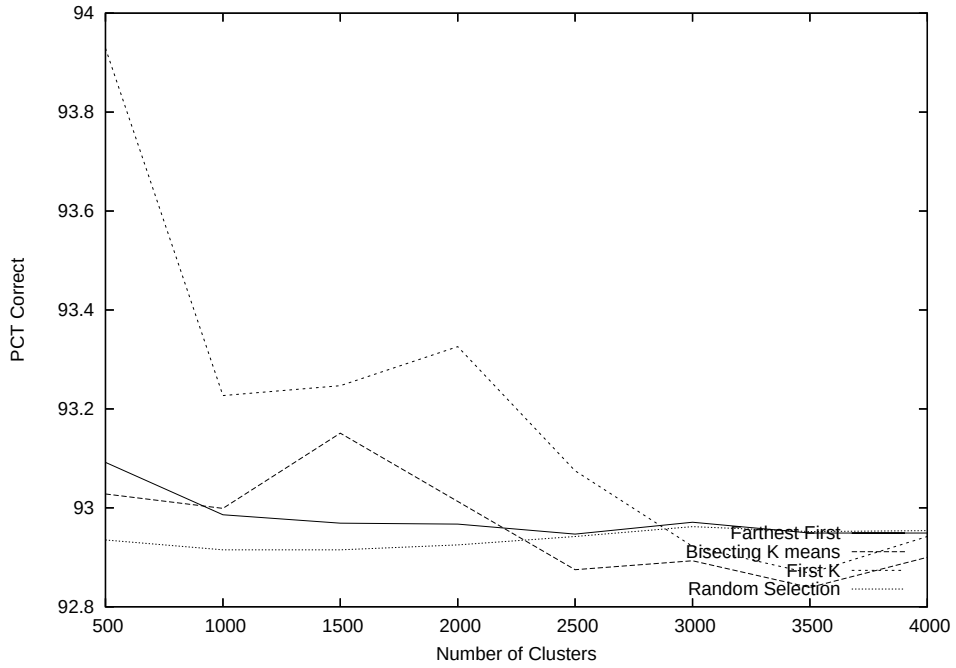

Figure 4.13: **mushroom-NaiveBayes**

very good. Naive Bayes baseline performance from Table 4.13 above is 95.362±0.683. The values for all four methods, as seen in Figure 4.13 and Table 4.13 are not statistically significantly different from the unclustered value, except for First K at five hundred. Farthest First and Random Selection are never significantly different from the baseline. What is interesting about this case is that both clustered sets improve on the base classification when two thousand or less clusters are used. First K in fact is markedly better at the five hundred cluster level. This value was rechecked and retested because it seemed to be too good relative to the other values obtained for the First K clusterer. While these values climb significantly the standard deviations climb as well indicating that these good values are highly dependent on the randomisation used.

4.1.14 mushroom-Logistic

Using the Logistic Regression algorithm with the mushroom dataset is somewhat of a challenge since the baseline performance of this combination is 99.803±0.597 which is close to a one hundred percent correct classification. Figure 4.14 seems to show wildly fluctuating classifications. However, all algorithms for all clusterer sizes are actually not statistically significantly different from each other. Looking at Table 4.14 we see that with the exception of a few points the values are all incredibly similar and in many cases the range covered by the standard deviation extends beyond perfect classification. Also of note is the way the order of the algorithms continually changes throughout the range of

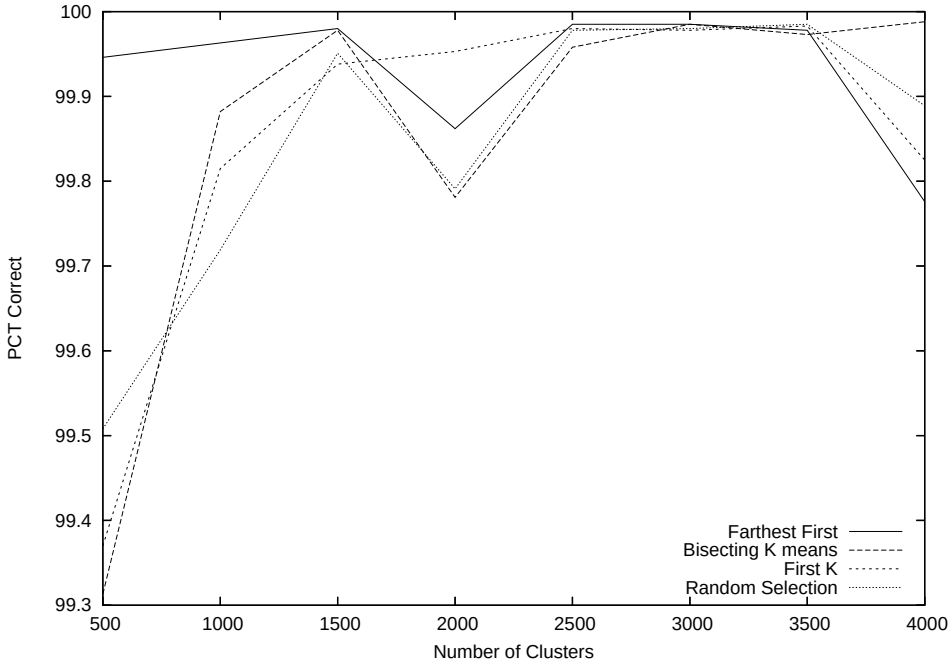

Figure 4.14: **mushroom-Logistic**

Table 4.14: mushroom-Logistic

Baseline	Clusters	FF	BI	FK	RN
99.803±0.597	500	**99.946±0.094**	99.313±0.453	**99.370±0.718**	**99.508±0.350**
99.803±0.597	1000	**99.963±0.072**	**99.882±0.134**	**99.815±0.158**	**99.719±0.435**
99.803±0.597	1500	**99.980±0.030**	**99.978±0.024**	**99.938±0.103**	**99.951±0.051**
99.803±0.597	2000	**99.862±0.290**	**99.781±0.559**	**99.953±0.090**	**99.791±0.310**
99.803±0.597	2500	**99.985±0.024**	**99.958±0.044**	**99.980±0.028**	**99.978±0.038**
99.803±0.597	3000	**99.985±0.024**	**99.985±0.026**	**99.978±0.029**	**99.980±0.032**
99.803±0.597	3500	**99.978±0.036**	**99.973±0.086**	**99.983±0.029**	**99.985±0.026**
99.803±0.597	4000	**99.776±0.633**	**99.988±0.039**	**99.825±0.527**	**99.889±0.325**

cluster sizes. At the points 500, 2000 and 4000 where one algorithm is noticeably but not significantly different from the others a different algorithm leads each time; Farthest First at 500; First K at 2000; Bisecting K-Means at 4000.

The shape in Figure 4.14 is the inverse to that seen in Figure 4.13. This reflects the clusterers making available to Naive Bayes information which otherwise it would not or could not use in its classification, while at the same time the summarisation process causes Logistic to lose information necessary for perfect classification. All clusterers and Random Selection can achieve a ninety percent data reduction on this dataset so most differences are truly minimal as the results for each clusterer are not statistically significantly different from each other.

4.1.15 opdigits-NaiveBayes

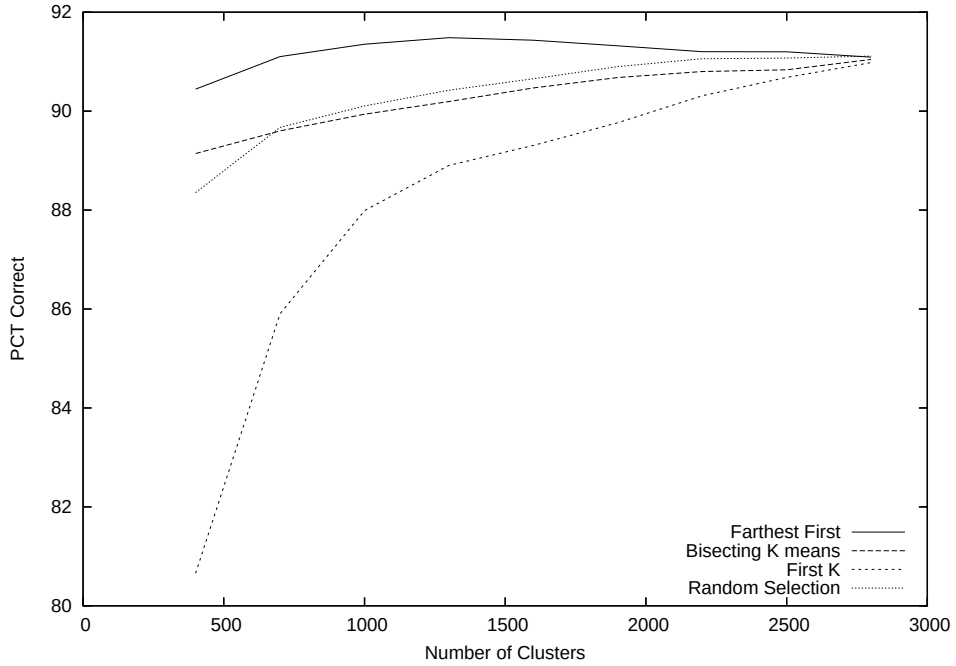

Figure 4.15: **opdigits-NaiveBayes**

With the opdigits dataset we have a similar situation to that seen in the initial experiments. First K in this instance performed significantly worse that the other methods for all clustering levels below two thousand five hundred. Farthest First is the standout performer surpassing the base classification twice and only falling below it with four hundred clusters. Figure 4.15 shows how close the other two methods are to each other and the differential between them. This is initially less than point one and grows to around eight percent. Random Selection is always closely followed by Bisecting K-Means, however below one thousand nine hundred clusters they both do significantly worse than the

Table 4.15: opdigits-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
91.085±0.302	400	90.444±0.629	89.142±0.483	80.658±2.907	88.352±0.707
91.085±0.302	700	**91.103±0.457**	89.601±0.507	85.904±1.069	89.672±0.452
91.085±0.302	1000	**91.352±0.357**	89.939±0.408	87.989±0.781	90.106±0.428
91.085±0.302	1300	**91.484±0.341***	90.195±0.550	88.903±0.444	90.423±0.389
91.085±0.302	1600	**91.434±0.315***	90.469±0.391	89.306±0.462	90.655±0.416
91.085±0.302	1900	**91.320±0.333**	90.679±0.485	89.765±0.430	**90.900±0.339**
91.085±0.302	2200	**91.202±0.293**	**90.800±0.399**	90.309±0.389	**91.060±0.342**
91.085±0.302	2500	**91.199±0.293**	**90.836±0.378**	90.683±0.308	**91.074±0.344**
91.085±0.302	2800	**91.092±0.293**	**91.046±0.286**	**90.982±0.296**	**91.114±0.313**

baseline. Table 4.15 shows just how close these two methods are to each other.

4.1.16 opdigits-Logistic

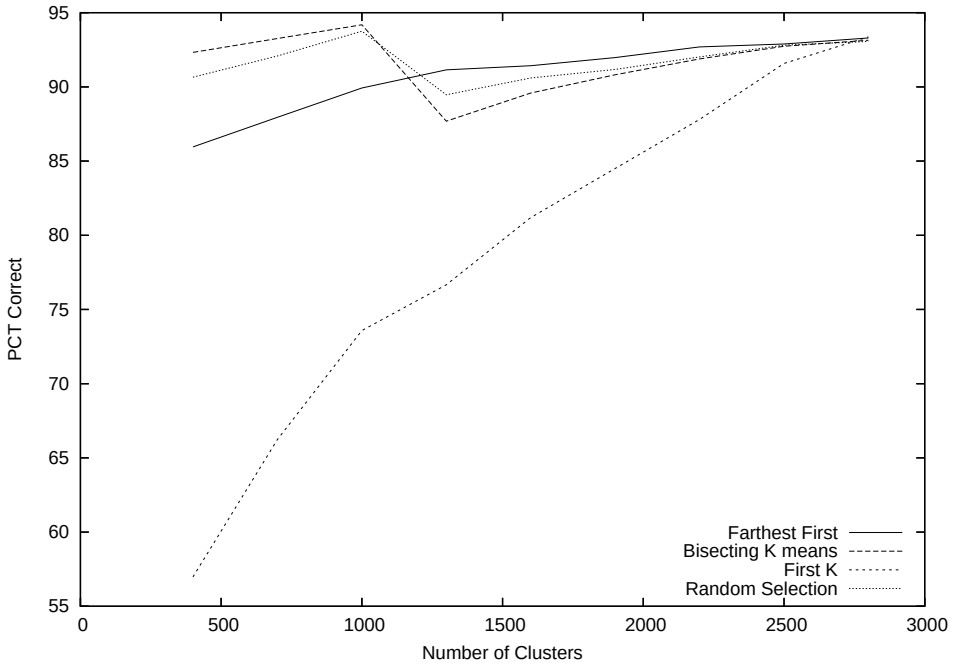

Figure 4.16: **opdigits-Logistic**

With the Logistic Regression algorithm and the opdigits dataset we have a very interesting set of results. At the two thousand eight hundred level all methods are essentially equal. Below this level First K descends to increasingly worse values as the level of summarisation increases. Random Selection and Bisecting K-Means initially follow a similar but much less severe trend. This is dramatically reversed at the one thousand cluster level where the accuracy for both of these methods becomes better than at the two thousand eight hundred level.

Table 4.16: opdigits-Logistic

Baseline	Clusters	FF	BI	FK	RN
93.156±0.910	400	85.950±1.310	92.320±0.797	56.976±9.483	90.647±0.875
93.156±0.910	700	87.946±1.840	**93.260±0.632**	66.230±4.145	92.078±1.112
93.156±0.910	1000	89.928±0.408	**94.181±0.461***	73.576±3.207	**93.754±0.653**
93.156±0.910	1300	91.146±0.732	87.686±1.565	76.658±2.413	89.466±0.660
93.156±0.910	1600	91.427±0.784	89.583±1.171	81.192±3.179	90.601±0.579
93.156±0.910	1900	91.975±0.518	90.811±0.981	84.494±1.780	91.174±0.519
93.156±0.910	2200	**92.694±0.439**	91.886±1.067	87.800±1.367	92.017±0.709
93.156±0.910	2500	**92.889±0.757**	**92.733±1.105**	91.573±1.648	**92.807±1.155**
93.156±0.910	2800	**93.292±0.742**	**93.138±0.921**	**93.366±0.863**	**93.057±0.906**

With Logistic Regression all methods start above their equivalent performance with Naive Bayes but degrade quickly to be significantly worse by the one thousand three hundred cluster level. When Bisecting K-Means and Random Selection improve their performance at the one thousand cluster level their performance moves to a level that is significantly better than that of Naive Bayes passing Farthest First which up to this point was the best performer. Table 4.16 shows interesting results for First K having a much larger variance than the other three methods. This variance is partially due to its extreme sensitivity to the ordering of the base data prior to clustering. Farthest First is the overall best method on this dataset achieving an average fifty percent data reduction, and twice with Naive bayes generating results which are significantly better then the baseline.

4.1.17 pendigits-NaiveBayes

Table 4.17: pendigits-NaiveBayes

Baseline	Clusters	FF	BI	FK	RN
85.640±0.313	500	83.461±1.015	85.298±0.283	73.845±4.200	83.621±1.796
85.640±0.313	1000	84.820±0.690	**85.280±0.587**	82.334±0.996	84.454±1.414
85.640±0.313	1500	**85.662±0.591**	**85.721±0.426**	83.657±0.575	84.756±1.209
85.640±0.313	2000	**86.134±0.462***	**85.508±0.367**	84.727±0.395	**85.206±0.938**
85.640±0.313	2500	**86.265±0.378***	**85.600±0.428**	84.698±0.467	**85.380±0.762**
85.640±0.313	3000	**86.399±0.341***	**85.582±0.345**	85.127±0.455	**85.475±0.564**
85.640±0.313	3500	**86.494±0.371***	**85.619±0.368**	85.382±0.231	**85.551±0.470**
85.640±0.313	4000	**86.385±0.266***	**85.640±0.322**	**85.582±0.284**	**85.600±0.405**
85.640±0.313	4500	**86.296±0.300***	**85.619±0.325**	**85.633±0.285**	**85.675±0.347**
85.640±0.313	5000	**85.919±0.329***	**85.670±0.373**	**85.739±0.296**	**85.644±0.333**

Pendigits appears from Figure 4.17 to be an easier clustering task than most of the others seen in this series of experiments. Farthest First out performs the baseline above two

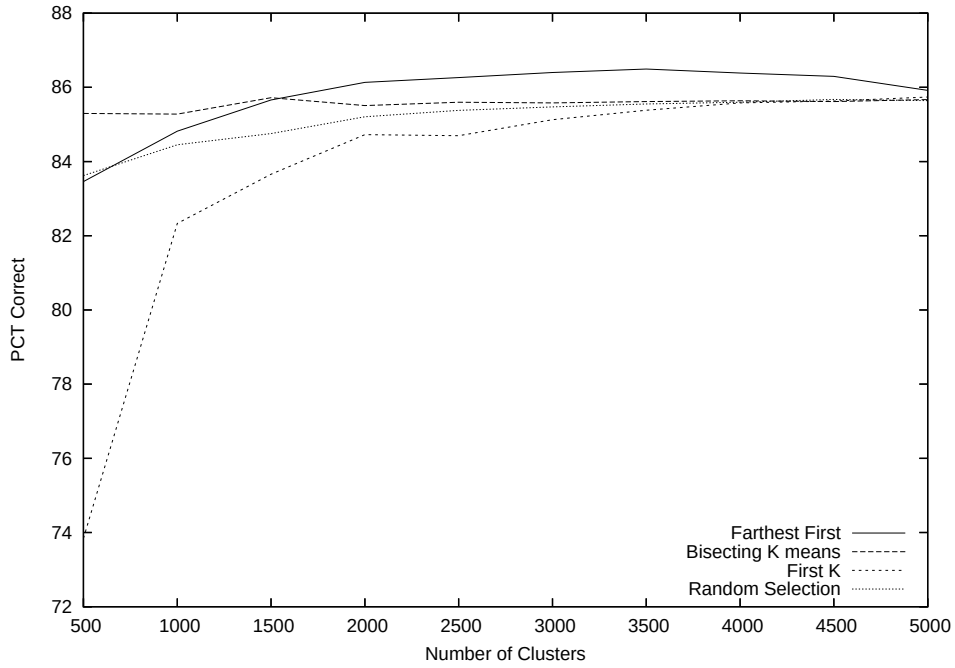

Figure 4.17: **pendigits-NaiveBayes**

thousand clusters. Above three thousand clusters there is no significant difference between any of the other clustering methods, although First K lags slightly. Other clusterers are not statistically significantly worse until two thousand clusters. Beyond this point Figure 4.17 shows the four methods diverging significantly. First K falls relative to the others while Random Selection falls by much smaller amounts. Bisecting K-Means however actually improves its score from here, beating Farthest First, and it even maintains near base classification levels for all numbers of clusters although it is significantly worse than the baseline at five hundred clusters. Table 4.17 shows a large increase in the standard deviations for both Random Selection and First K beyond the two thousand cluster level.

4.1.18 pendigits-Logistic

Using the Logistic Regression algorithm with the pendigits dataset creates a similar picture to ones we have seen before. First K as shown in Figure 4.18, is always significantly worse than the other methods and the baseline. Farthest First is again the standout performer although it is equivalent to both Bisecting K-Means and Random Selection until there are less than three thousand clusters, from there they fall away almost in tandem. The only significant differences between them being at one thousand five hundred. All three of these clustering methods are as good as the base classifier for values above two thousand.

Both Farthest First and Bisecting K-Means achieve over seventy percent summarisation without significant loss on Naive Bayes. This is not repeated with Logistic Regression

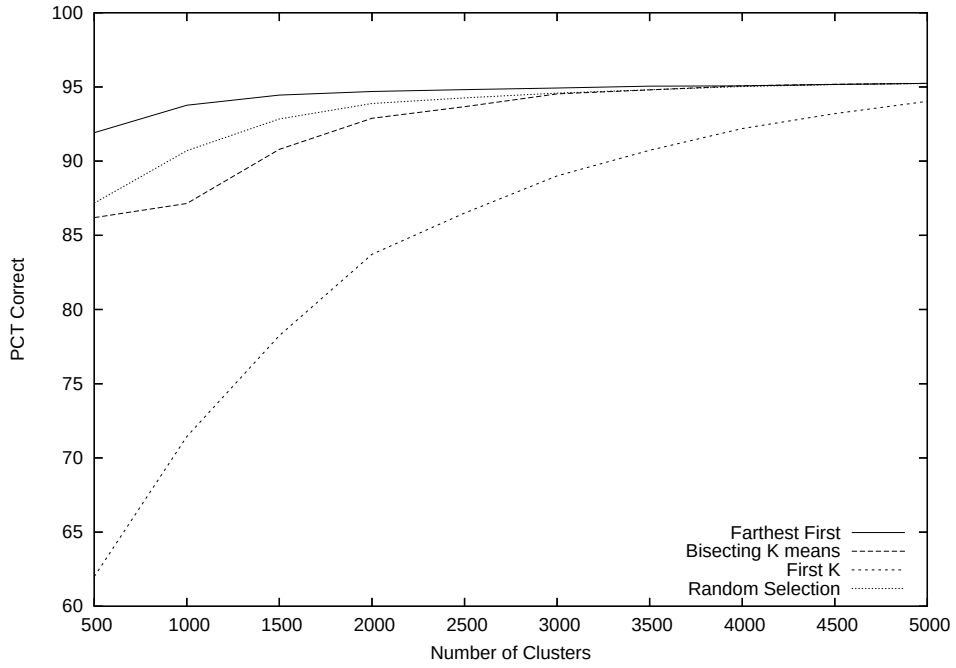

Figure 4.18: **pendigits-Logistic**

Table 4.18: pendigits-Logistic

Baseline	Clusters	FF	BI	FK	RN
95.287±0.188	500	91.914±0.801	86.186±1.555	62.007±2.419	87.171±1.202
95.287±0.188	1000	93.766±0.346	87.143±1.295	71.437±1.882	90.697±0.791
95.287±0.188	1500	94.445±0.305	90.793±0.591	78.237±1.774	92.838±0.686
95.287±0.188	2000	94.691±0.221	92.884±0.598	83.719±1.026	93.879±0.531
95.287±0.188	2500	94.813±0.262	93.663±0.746	86.499±0.919	94.258±0.357
95.287±0.188	3000	94.920±0.200	94.532±0.368	89.003±0.781	94.583±0.281
95.287±0.188	3500	95.049±0.152	94.798±0.321	90.728±0.497	94.809±0.245
95.287±0.188	4000	95.071±0.172	95.080±0.256	92.196±0.578	95.020±0.209
95.287±0.188	4500	**95.169±0.156**	**95.173±0.181**	93.202±0.572	**95.153±0.186**
95.287±0.188	5000	**95.242±0.189**	**95.240±0.150**	94.028±0.378	**95.231±0.208**

where both achieve only ten percent reduction before they loose accuracy. Of these two Farthest First out performs the baseline with up to sixty percent reductions on Naive Bayes. This difference makes Farthest First the standout performer on this dataset.

4.2 Summary

The key measure of usefulness of a particular clustering method on a dataset is the amount of summarisation the clusterer can achieve before classification accuracy is reduced. Table 4.19 shows the maximum summarisation attained by each clustering method for each dataset, when using the simple classifier, Naive Bayes. The clustering methods are shown in rank order with the method achieving most summarisation to the left and that achieving least summarisation to the right. The results for each classifier were summarised and tabulated separately as an overall average would present somewhat of an unbalanced picture for many datasets since in most cases a clustering method that performs well with one classifier will perform poorly with the other. A good example of this is the waveform datasets. On both of these datasets under Naive Bayes, First K and Bisecting K-Means both improve on the base classification for all low numbers of clusters, however they perform poorly under Logistic Regression. This would give them an overall summarisation score which is worse than that of Random Selection, even though they are significantly better under Naive Bayes.

Datasets	Clusterer Ranking	Reduction Achieved
kr-vs-kp	(FF, RN, BI, FK, KM, EM)	(60%, 60%, 6%, 6%, 0%, -)
hypothyroid	(All)	(-)
Agrawal	(FF, RN, FK, BI)	(87%, 87%, 87%, 50%)
waveform21	(FK, BI, RN, FF)	(87%, 87%, 87%, 0%)
waveform40	(FK, BI, RN, FF)	(87%, 87%, 87%, 0%)
letter	(BI, RN, FK, FF)	(90%, 65%, 60%, -)
mushroom	(FK, FF, BI, RN)	(87%, 87%, 87%, 87%)
opdigits	(FF, RN, BI, FK)	(75%, 32%, 21%, 0%)
pendigits	(BI, FF, RN, FK)	(80%, 70%, 60%, 20%)

Table 4.19: Naive Bayes: Maximum summarisation achieved for each dataset

There is no overall best clustering method with the simple classifier. Farthest First and First K both obtain the best results on three datasets, and Bisecting K-Means on two. The only method that is never the best with the simple classifier is Random Selection which is not a clusterer. Overall the levels of summarisation attained with the simple classifier are excellent. They clearly demonstrate the benefits of clustering over Random Selection for data reduction. However Table 4.19 does show how important it is to select

the right clustering method for the dataset being clustered.

Datasets	Ranking	Reduction Achieved
kr-vs-kp	(FF, RN, KM, BI, FK, EM)	(60%, 40%, 30%, 20%, 10%, -)
hypothyroid	(FF-BI, FK-KM-EM-RN)	(0%, -)
Agrawal	(FK, RN-FF-BI)	(75%, 0%)
waveform21	(FF, RN, BI, FK)	(87%, 75%, 50%, 37%)
waveform40	(FF, RN, BI, FK)	(75%, 50%, 37%, 37%)
letter	(BI, RN, FK, FF)	(20%, 10%, 0%, -)
mushroom	(FF, FK, RN, BI)	(87%, 87%, 87%, 75%)
opdigits	(BI, RN, FF, FK)	(75%, 64%, 21%, 0%)
pendigits	(BI, FF, RN, FK)	(10%, 10%, 15%-)

Table 4.20: Logistic Regression: Maximum summarisation achieved for each dataset

When using a more complex classifier, Logistic Regression, two clustering methods dominate, as shown in Table 4.20. Farthest First has the best summarisation on five datasets and Bisecting K-Means is the best on four. First K is the best summarisation method on one dataset, Agrawal. The result it achieved on this dataset was rather spectacular, compared to it's performance on other datasets. Random Selection again performs in the middle of the range never the best method and never the worst.

Overall, on all datasets, for each classification method, some clusterer outperforms or equals Random Selection. In no case is Random Selection equal to the better clustering method. This shows the Cluster Classifier to be a worthwhile method of data summarisation on nominal datasets, provided an appropriate clustering method is chosen for the dataset being used.

Finally, preliminary experiments were run to evaluate the need for weights in the classifier and to attempt data summarisation by removing clusters with little support. Preliminary experiments were also run to evaluate the effectiveness of the Cluster Classifier as a means of outlier detection for clusterers where isolated instances are likely to form clusters with low support. The result of these experiments are outlined in Appendix A and B. In both cases the study generated more questions than answers.

Chapter 5

Evaluation on Numeric Datasets

This set of experiments evaluates the performance of the Cluster Classifier on numeric datasets. The first two datasets used are Kin8nm and ailerons. On these smaller datasets all clusterers described in Chapter 2 are used. For the remaining datasets, the more complex clusterers Simple K-Means and EM are not used since their runtimes are too long to make them effective methods of data summarisation. The purpose is to evaluate the performance of a range of clusterers when used with the Cluster Classifier to reduce a dataset. Since some clusterers perform better with simple classifiers and others are more suited to complex classifiers, the experiments first used a simple classifier, Linear Regression, and then a more complex one, M5.

5.1 Experiments

Each experiment uses a single dataset and one classifier with a variety of clustering methods in order to evaluate the performance of the various clustering methods within the Cluster Classifier framework.

5.1.1 ailerons-LinearRegression

Table 5.1: ailerons-LinearRegression

Baseline	Clusters	KM	FF	BI	FK	EM	RN
0.90±0.00	1000	0.85±0.08	**0.90±0.00**	**0.86±0.09**	0.41±0.31	**0.88±0.05**	**0.90±0.00**
0.90±0.00	2000	**0.87±0.10**	**0.90±0.00**	**0.89±0.04**	0.79±0.14	**0.84±0.17**	0.88±0.03
0.90±0.00	3000	**0.90±0.00**	**0.90±0.00**	**0.90±0.01**	0.82±0.12	**0.84±0.18**	**0.77±0.24**
0.90±0.00	4000	**0.90±0.00**	**0.90±0.00**	**0.90±0.01**	**0.86±0.08**	**0.84±0.18**	**0.78±0.21**
0.90±0.00	5000	**0.90±0.00**	**0.90±0.00**	**0.90±0.01**	**0.90±0.00**	**0.85±0.15**	**0.90±0.01**
0.90±0.00	6000	**0.90±0.00**	**0.90±0.00**	**0.90±0.00**	**0.89±0.02**	**0.90±0.00**	**0.89±0.05**
0.90±0.00	7000	**0.90±0.00**	**0.90±0.00**	**0.90±0.00**	**0.90±0.00**	**0.90±0.00**	**0.90±0.00**

Figure 5.1 shows the performance of all clusterers on the ailerons dataset. The best performer here is Farthest First which never deviates significantly from the performance

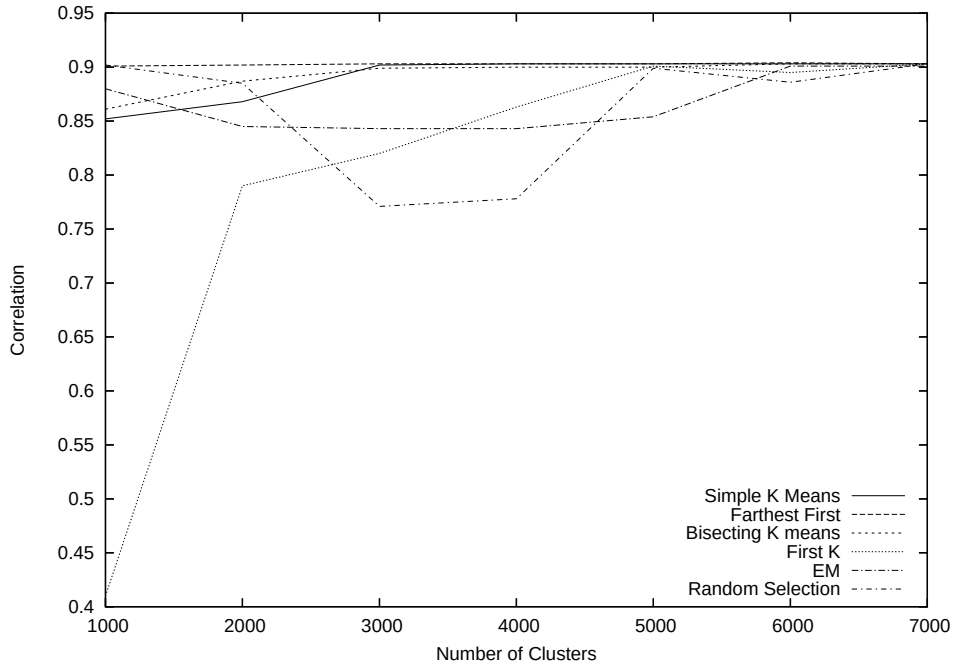

Figure 5.1: **ailerons-LinearRegression**

of Linear Regression on the full dataset. Bisecting K-Means and EM are also not statistically significantly different at any number of clusters. All clusterers perform well as does Random Selection. First K and Random Selection both perform significantly worse than the base classifier for a significant proportion of cluster values. Interestingly both EM and Random Selection improve their correlation markedly at the highest level of summarisation while First K has its worst performance there.

5.1.2 ailerons-M5

Table 5.2: ailerons-M5

Baseline	Clusters	KM	FF	BI	FK	EM	RN
0.92±0.00	1000	0.91±0.00	0.91±0.00	0.91±0.00	0.90±0.00	0.91±0.00	0.91±0.00
0.92±0.00	2000	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00
0.92±0.00	3000	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00
0.92±0.00	4000	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00
0.92±0.00	5000	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00
0.92±0.00	6000	0.91±0.00	**0.92±0.00**	0.91±0.00	0.91±0.00	0.91±0.00	0.91±0.00
0.92±0.00	7000	**0.92±0.00**	**0.92±0.00**	**0.92±0.00**	**0.92±0.00**	0.91±0.00	**0.92±0.00**

The M5 classifier on the ailerons dataset produces a very high correlation. This is closely matched by all clustered values as seen in Table 5.2. When the least number of clusters are used the difference from the baseline is only 0.01. This is however a fairly

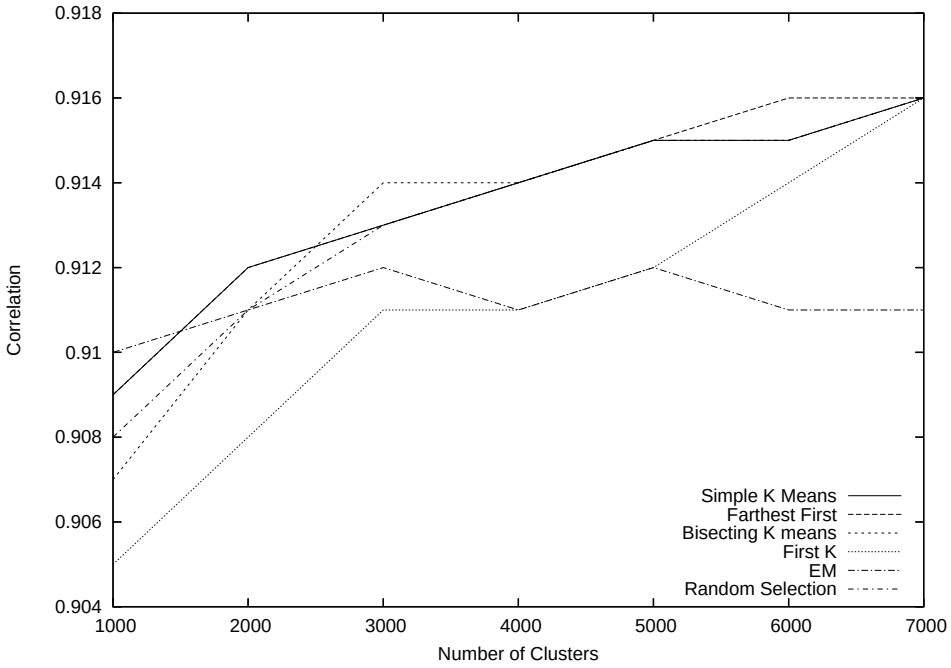

Figure 5.2: **ailerons-M5**

statistically significant result owing to the small variation in the measurements. Table 5.2 shows three clusterers, Simple K-Means, Bisecting K-Means and Farthest First, producing nearly identical results which are not statistically significantly different from the baseline with five thousand or more clusters. This is further illustrated in Figure 5.2 where these clusterers are consistently together, obscuring each other.

Overall the results of this experiment are very good even for EM which gave a reasonable performance on this dataset. However while all clusterers except First K manage eighty percent data reduction with Linear Regression those same clusterers only manage twenty five percent with M5

5.1.3 kin8nm-LinearRegression

Table 5.3: kin8nm-LinearRegression

Baseline	Clusters	KM	FF	BI	FK	EM	RN
0.64±0.01	500	**0.64±0.01**	0.63±0.01	**0.64±0.01**	0.61±0.01	**0.64±0.01**	0.63±0.01
0.64±0.01	1000	**0.64±0.01**	0.63±0.01	**0.64±0.01**	0.61±0.01	**0.64±0.01**	**0.64±0.01**
0.64±0.01	1500	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	0.62±0.01	**0.64±0.01**	**0.64±0.01**
0.64±0.01	2000	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	0.62±0.01	**0.64±0.01**	**0.64±0.01**
0.64±0.01	2500	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	0.63±0.01	**0.64±0.01**	**0.64±0.01**
0.64±0.01	3000	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	0.63±0.01	**0.64±0.01**	**0.64±0.01**
0.64±0.01	3500	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**
0.64±0.01	4000	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**	**0.64±0.01**

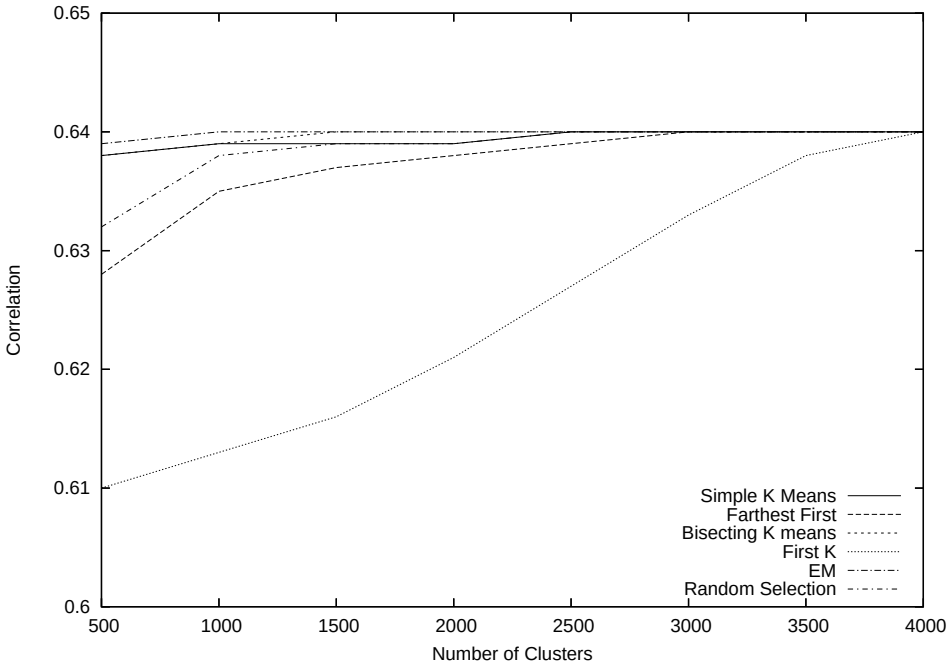

Figure 5.3: **kin8nm-LinearRegression**

On the kin8nm dataset the baseline for Linear Regression is 0.640±0.008. Figure 5.3 shows that the performance of the clustered data on this dataset is almost indistinguishable for most cluster sizes. With the exception of the First K clusterer all other methods produce results that are not statistically significantly different from the baseline performance. This can be seen in greater detail in Table 5.3 where most measurements are in bold. Both Farthest First and Random Selection dip significantly at five hundred clusters. Of note here is that none of the clustered datasets ever exceed the unclustered value. Everything except First K betters Random Selection.

5.1.4 kin8nm-M5

Table 5.4: kin8nm-M5

Baseline	Clusters	KM	FF	BI	FK	EM	RN
0.77±0.02	500	0.66±0.03	0.63±0.01	0.67±0.03	0.61±0.01	0.64±0.02	0.64±0.02
0.77±0.02	1000	0.68±0.04	0.64±0.01	0.68±0.04	0.63±0.02	0.69±0.03	0.67±0.03
0.77±0.02	1500	0.69±0.03	0.70±0.03	0.72±0.02	0.65±0.02	0.70±0.03	0.69±0.03
0.77±0.02	2000	0.72±0.03	0.68±0.03	0.72±0.02	0.69±0.03	0.72±0.03	0.72±0.02
0.77±0.02	2500	0.74±0.02	0.71±0.05	0.73±0.02	0.71±0.02	0.72±0.04	0.73±0.02
0.77±0.02	3000	0.75±0.02	**0.75±0.03**	0.75±0.02	0.73±0.02	0.74±0.01	0.74±0.02
0.77±0.02	3500	**0.76±0.02**	**0.76±0.02**	0.74±0.02	0.75±0.02	0.74±0.02	0.75±0.01
0.77±0.02	4000	**0.76±0.02**	**0.76±0.02**	**0.76±0.02**	**0.77±0.02**	0.74±0.03	**0.77±0.02**

As Figure 5.4 shows clustering the kin8nm dataset for use with M5 loses accuracy

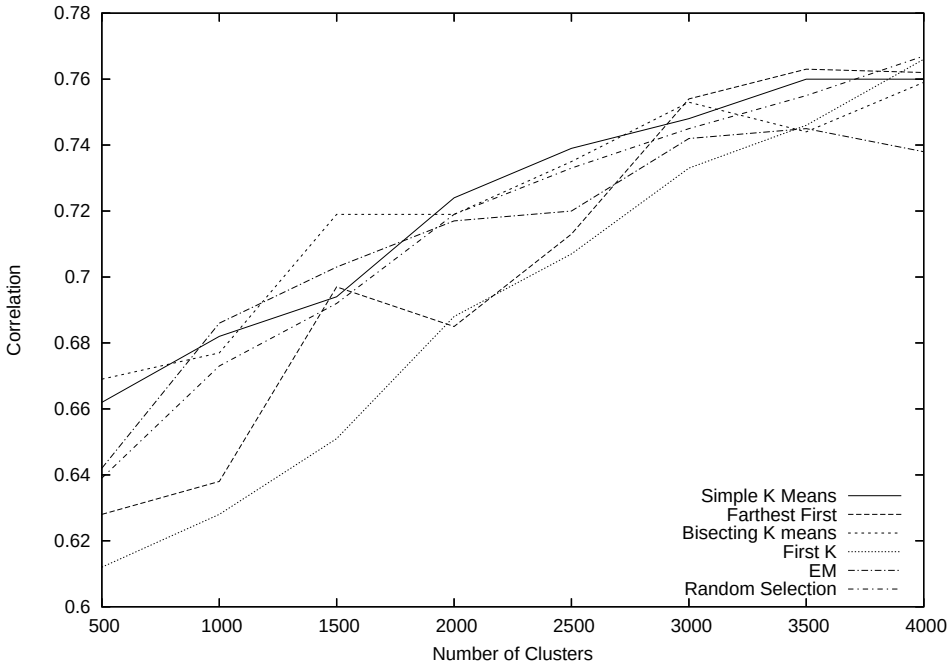

Figure 5.4: **kin8nm-M5**

almost immediately. The total accuracy drop is not severe, at only sixteen percent in the worst case. Bisecting K-Means is overall the most consistent performer, as seen in Table 5.4, however only two of it's points are not statistically significantly worse than the baseline. Farthest First is not statistically significantly different from the baseline for more than three thousand clusters but with fewer clusters it is among the worst methods.

The variance on all measurements is also much higher in this case than under Linear Regression. This is reflected in the erratic swings in the values measured compared with the smooth degradation seen in Table 5.3.

At this point some explanation of the poor performance of EM is needed. As EM is a model based approach, which does not work by assigning data-points to centres as the other clusterer do, when asked for a given number of clusters EM will try and explain the data using normal distributions. EM does not generate enough clusters. It typically generates only half the requested number. These missing clusters are in fact generated however as EM is not able to fit a model for them they become phantom clusters with no support in data. These clusters are removed since they have zero weight.

From these initial datasets a few trends can be seen. It appears that Farthest First and First K are much better on nominal data while K-Means works better in the numeric case. Simple K-Means is better than Bisecting K-Means in most cases but also much slower.

The remaining experiments remove two of the clusterers, EM and Simple K-Means from the set of clusterers used. These clusterers are much slower than the other methods

and their results are not sufficient to warrant their inclusion in further testing. Since one of the goals in clustering a dataset is to reduce its size in order for more complex classification algorithms to be applied in reasonable time it follows that the process of reducing the dataset size must also be a fast process since it should take less time for both the clustering and the subsequent classification than to run the classifier on the original dataset.

EM was simply not accurate enough on any dataset to warrant the time taken. Simple K-Means while a good performer is not significantly better than Bisecting K-Means which is much faster.

5.1.5 2DPlanes-LinearRegression

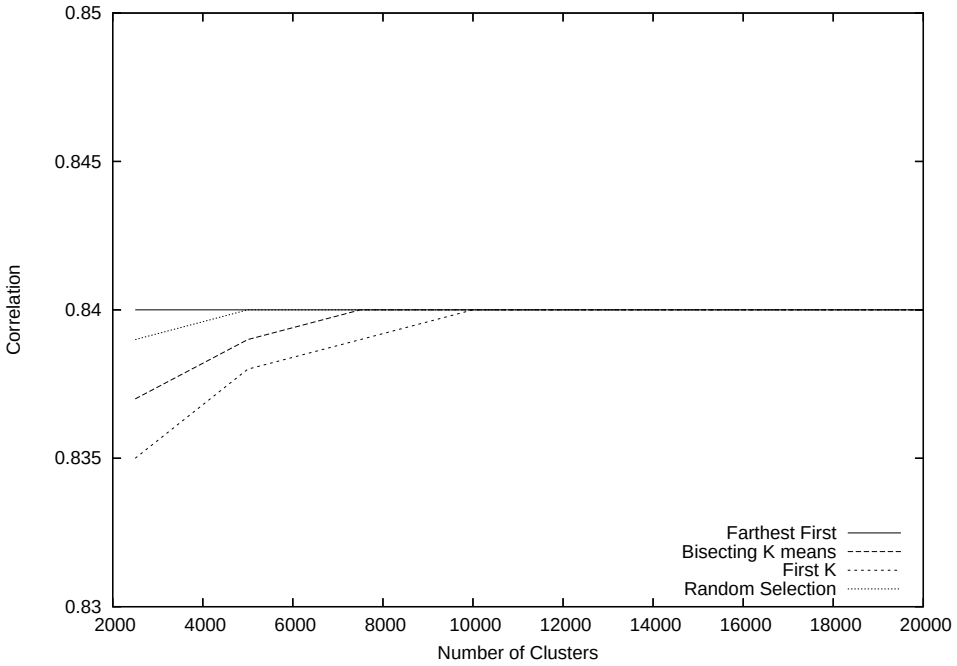

Figure 5.5: **2DPlanes-LinearRegression**

Table 5.5: 2DPlanes-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.840±0.001	2500	**0.840±0.001**	0.837±0.001	0.835±0.002	0.839±0.001
0.840±0.001	5000	**0.840±0.001**	0.839±0.001	0.838±0.001	**0.840±0.001**
0.840±0.001	7500	**0.840±0.001**	**0.840±0.001**	0.839±0.001	**0.840±0.001**
0.840±0.001	10000	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**
0.840±0.001	12500	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**
0.840±0.001	15000	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**
0.840±0.001	17500	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**
0.840±0.001	20000	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**	**0.840±0.001**

2DPlanes is a very simple clustering task. In fact, Figure 5.5 at first glance appears similar to Figure 5.3 and Figure 4.18. However the entire range for the correlations is less than zero point zero zero five percent which is much less than these other figures. This is because of the small standard deviations in these values. All results that are not exactly the same as the baseline are in fact statistically significantly different. Table 5.5 shows that all values above ten thousand are completely identical to each other and to the correlation of the classifier on the unclustered dataset.

5.1.6 2DPlanes-M5

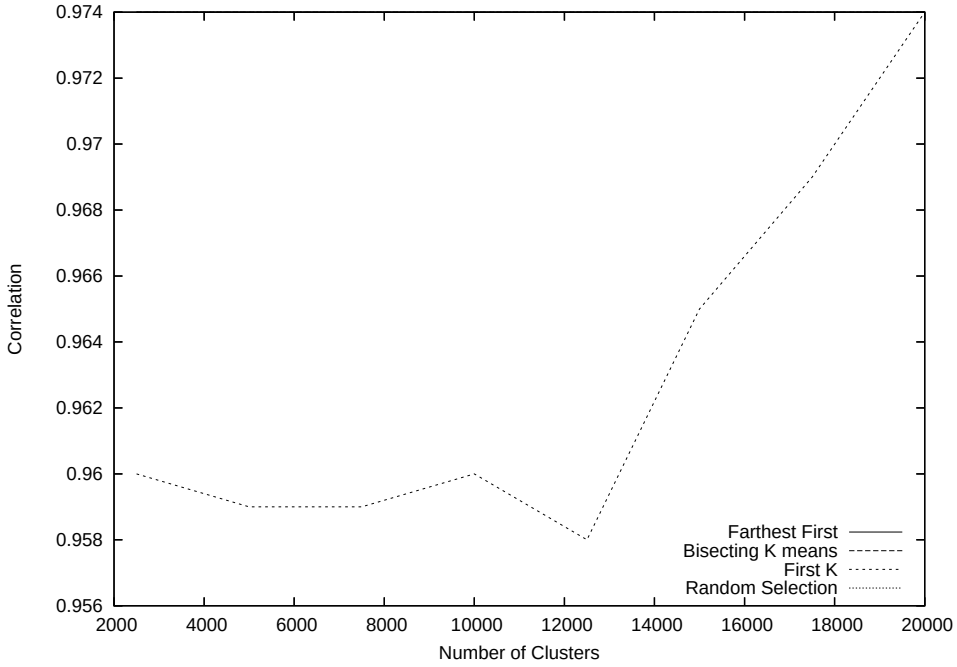

Figure 5.6: **2DPlanes-M5**

Table 5.6: 2DPlanes-M5

Baseline	Clusters	FF	BI	FK	RN
0.974±0.000	2500	**0.974±0.000**	**0.974±0.000**	0.960±0.005	**0.974±0.000**
0.974±0.000	5000	**0.974±0.000**	**0.974±0.000**	0.959±0.003	**0.974±0.000**
0.974±0.000	7500	**0.974±0.000**	**0.974±0.000**	0.959±0.002	**0.974±0.000**
0.974±0.000	10000	**0.974±0.000**	**0.974±0.000**	0.960±0.001	**0.974±0.000**
0.974±0.000	12500	**0.974±0.000**	**0.974±0.001**	0.958±0.003	**0.974±0.001**
0.974±0.000	15000	**0.974±0.000**	**0.974±0.000**	0.965±0.001	**0.974±0.000**
0.974±0.000	17500	**0.974±0.000**	**0.974±0.000**	0.969±0.000	**0.974±0.000**
0.974±0.000	20000	**0.974±0.000**	**0.974±0.000**	**0.974±0.000**	**0.974±0.000**

M5 performs significantly better than Linear Regression on the 2DPlanes dataset. This accounts for the entire difference between the scores for Bisecting K-Means and Random

Selection in Figure 5.6 and Figure 5.5. Farthest First, Bisecting K-Means and Random Selection perform as well as the unclustered M5 tree for all numbers of clusters. First K on the other hand, which is more susceptible to randomisation effects, falls a total of 0.016 percent between 20,000 and 12,500 before leveling off at the lower value. This difference in performance is not hugely significant. First K's worse performance with M5 can partially be explained by M5's only partial support for weighted instances, which are much more critical to the First K algorithm than to the other clusterers.

Farthest First is the best performing method on this dataset, achieving close to ninety percent data reduction with both classification methods. Bisecting K-Means also comes close achieving sixty percent reduction with Linear Regression and ninety percent with M5.

5.1.7 fried-LinearRegression

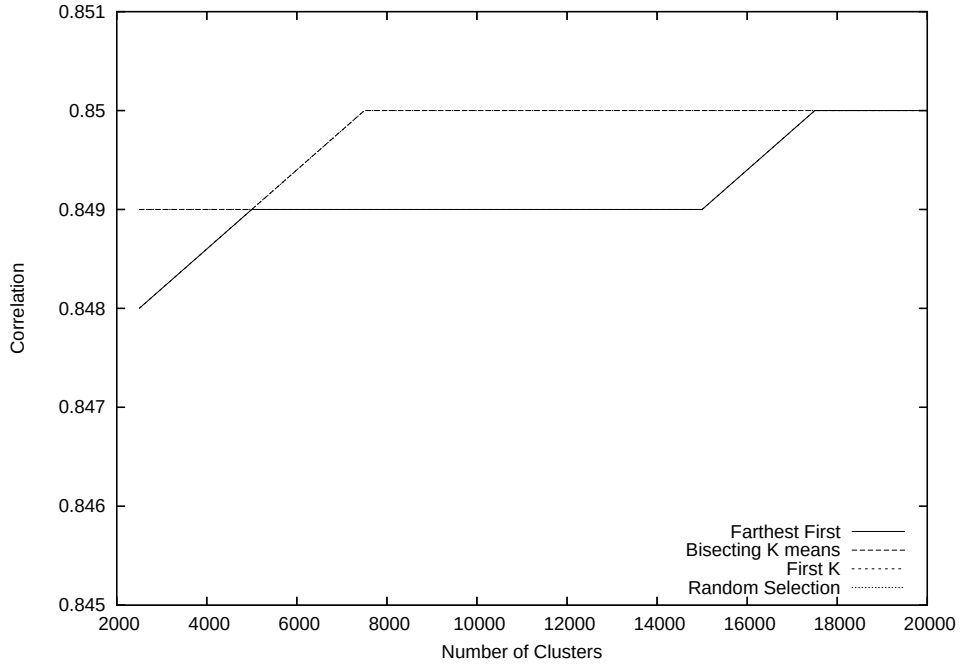

Figure 5.7: **fried-LinearRegression**

Linear Regression with the fried dataset is another good but uninteresting result. Figure 5.7 shows minute variations in classifier performance for each different clusterer and level of clustering. Of note all values in Table 5.7 are within the standard deviation of all other values, although both First K and Farthest First produce results that are statistically significantly worse than the baseline when they produce only two thousand five hundred clusters. This dataset was originally a generated dataset like 2DPlanes and it is very easy to cluster.

Table 5.7: fried-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.850±0.002	2500	0.848±0.001	**0.849±0.002**	0.848±0.002	**0.849±0.002**
0.850±0.002	5000	**0.849±0.002**	**0.849±0.002**	**0.849±0.002**	**0.849±0.002**
0.850±0.002	7500	**0.849±0.002**	**0.850±0.002**	**0.849±0.002**	**0.850±0.002**
0.850±0.002	10000	**0.849±0.002**	**0.850±0.002**	**0.849±0.002**	**0.850±0.002**
0.850±0.002	12500	**0.849±0.002**	**0.850±0.002**	**0.849±0.002**	**0.850±0.002**
0.850±0.002	15000	**0.849±0.002**	**0.850±0.002**	**0.849±0.002**	**0.850±0.002**
0.850±0.002	17500	**0.850±0.002**	**0.850±0.002**	**0.850±0.002**	**0.850±0.002**
0.850±0.002	20000	**0.850±0.002**	**0.850±0.002**	**0.850±0.002**	**0.850±0.002**

5.1.8 fried-M5

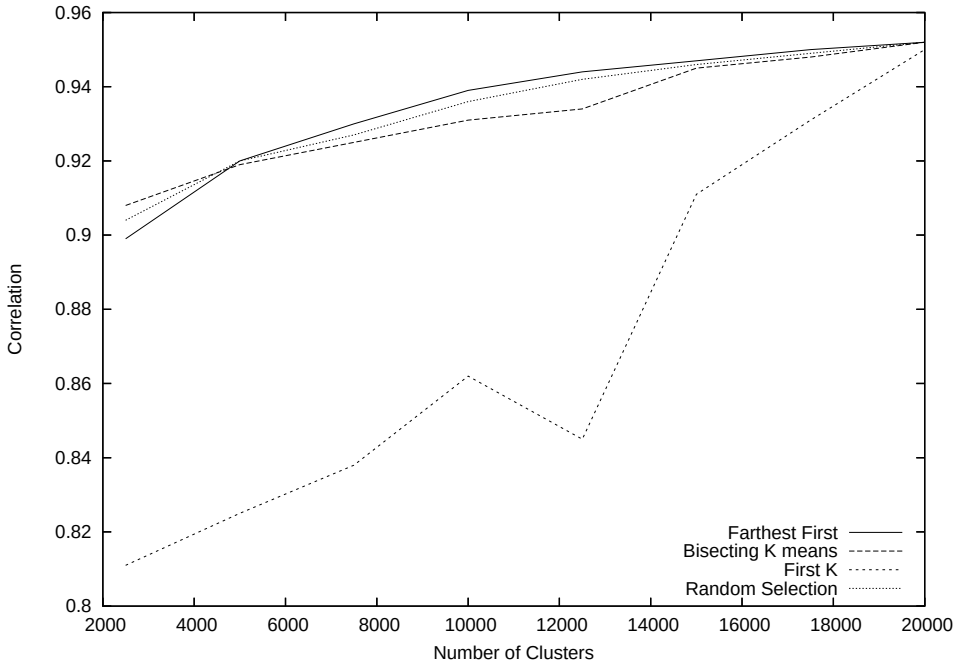

Figure 5.8: **fried-M5**

Using M5 with the fried dataset provides similar results to 2DPlanes. As Figure 5.8 shows First K runs significantly worse than its counterparts for all numbers of clusters. In this case the values for Bisecting K-Means, Farthest First and Random Selection also fall as the number of clusters decreases. Of note here is the dip in accuracy experienced by First K when 12,500 clusters are generated. This was also the point at which First K performed worst on the 2DPlanes dataset. Bisecting K-Means also shows a significant fall in performance at the 12,500 clusters level. This is perhaps due to the structure of these datasets.

The fried dataset shows significant differences between clusterer performances between

Table 5.8: fried-M5

Baseline	Clusters	FF	BI	FK	RN
0.952±0.001	2500	0.899±0.006	0.908±0.005	0.811±0.007	0.904±0.005
0.952±0.001	5000	0.920±0.003	0.919±0.004	0.825±0.006	0.920±0.002
0.952±0.001	7500	0.930±0.003	0.925±0.004	0.838±0.007	0.927±0.003
0.952±0.001	10000	0.939±0.002	0.931±0.003	0.862±0.004	0.936±0.004
0.952±0.001	12500	0.944±0.002	0.934±0.003	0.845±0.008	0.942±0.003
0.952±0.001	15000	0.947±0.001	0.945±0.002	0.911±0.003	0.946±0.002
0.952±0.001	17500	0.950±0.002	0.948±0.001	0.931±0.003	0.949±0.002
0.952±0.001	20000	**0.952±0.001**	**0.952±0.001**	0.950±0.002	**0.952±0.002**

the two classification methods. With Linear Regression all methods achieve seventy five percent or better summarisation without loss of accuracy. Under M5 no significant summarisation can be achieved by any clustering method as the clustered datasets only match the base classification at twenty thousand clusters which is the level of no summarisation.

5.1.9 mv-LinearRegression

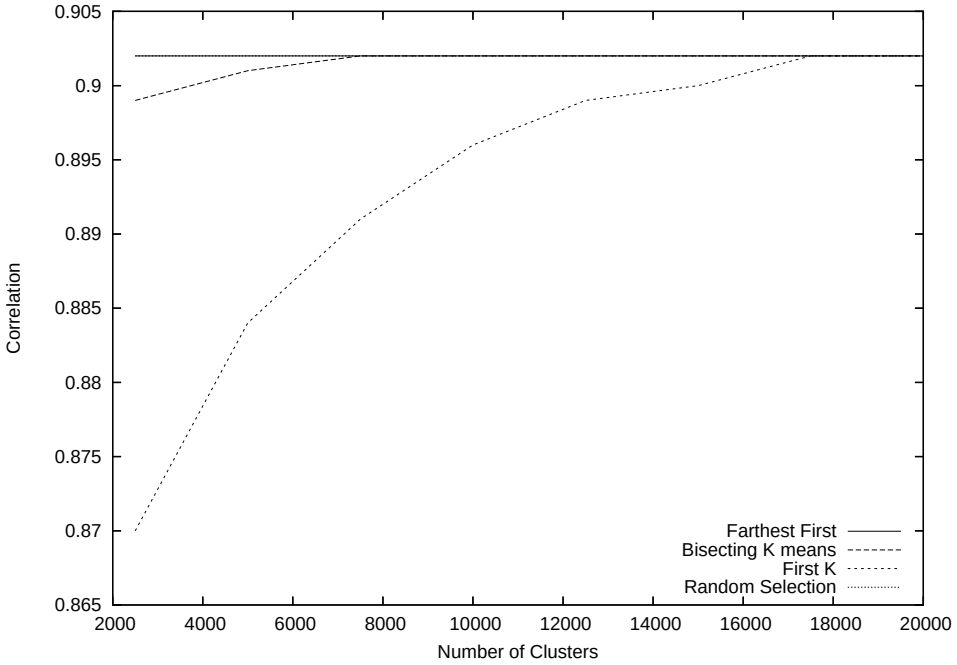

Figure 5.9: **mv-LinearRegression**

Mv is another dataset similar to the two we described immediately above. The Linear Regression results are also similar. Mv however provides a more difficult problem for First K. Figure 5.9 shows this performance degradation which is significantly more pronounced than for the two previous datasets. This fall is only three percent from the correlation seen for the base classifier in Table 5.9 above. Bisecting K-Means falls off on the two

Table 5.9: mv-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.902±0.001	2500	**0.902±0.001**	0.899±0.001	0.870±0.003	**0.902±0.001**
0.902±0.001	5000	**0.902±0.001**	0.901±0.001	0.884±0.002	**0.902±0.001**
0.902±0.001	7500	**0.902±0.001**	**0.902±0.001**	0.891±0.001	**0.902±0.001**
0.902±0.001	10000	**0.902±0.001**	**0.902±0.001**	0.896±0.001	**0.902±0.001**
0.902±0.001	12500	**0.902±0.001**	**0.902±0.001**	0.899±0.001	**0.902±0.001**
0.902±0.001	15000	**0.902±0.001**	**0.902±0.001**	0.900±0.001	**0.902±0.001**
0.902±0.001	17500	**0.902±0.001**	**0.902±0.001**	**0.902±0.001**	**0.902±0.001**
0.902±0.001	20000	**0.902±0.001**	**0.902±0.001**	**0.902±0.001**	**0.902±0.001**

smallest numbers of clusters although the accuracy loss is not large, it is significant due to the small standard deviations. Farthest First and Random Selection are both identical to the baseline on this dataset.

5.1.10 mv-M5

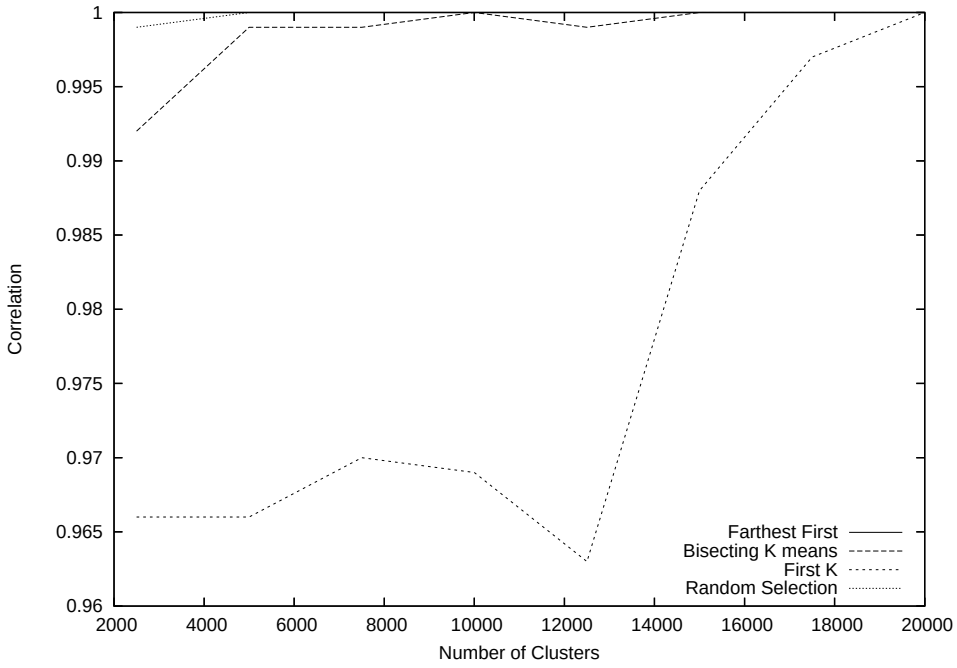

Figure 5.10: **mv-M5**

M5 when used with the mv dataset produces a perfect correlation. This makes it a harder problem than the two similar datasets encountered earlier. Farthest First rises to the challenge matching the baseline classification with any number of clusters, as seen in Table 5.10. Both Bisecting K-Means and Random Selection perform close to perfect for all cluster sizes. First K is only as good as the rest at 40,000 clusters where not real clustering occurs. Figure 5.10 shows how badly it actually performs, the shape here is very similar

Table 5.10: mv-M5

Baseline	Clusters	FF	BI	FK	RN
1.000±0.000	2500	**1.000±0.000**	0.992±0.005	0.966±0.014	0.999±0.000
1.000±0.000	5000	**1.000±0.000**	0.999±0.001	0.966±0.013	**1.000±0.000**
1.000±0.000	7500	**1.000±0.000**	0.999±0.000	0.970±0.007	**1.000±0.000**
1.000±0.000	10000	**1.000±0.000**	**1.000±0.000**	0.969±0.007	**1.000±0.000**
1.000±0.000	12500	**1.000±0.000**	0.999±0.001	0.963±0.010	**1.000±0.000**
1.000±0.000	15000	**1.000±0.000**	**1.000±0.000**	0.988±0.005	**1.000±0.000**
1.000±0.000	17500	**1.000±0.000**	**1.000±0.000**	0.997±0.002	**1.000±0.000**
1.000±0.000	20000	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**

to that seen with 2DPlanes. Of note is the minimum value which occurs at 12,500, the same place as the minimum in Figure 5.6. Bisecting K-Means also performs noticeably worse at this point before recovering to perfection. This is similar to the relationship seen in both fried and 2DPlanes for this number of clusters.

Farthest First is the overall best clustering method on this dataset, producing results identical to the baseline even with close to ninety percent summarisation. However Random Selection also performs well with seventy five percent or more summarisation for both classification methods.

5.1.11 house8L-LinearRegression

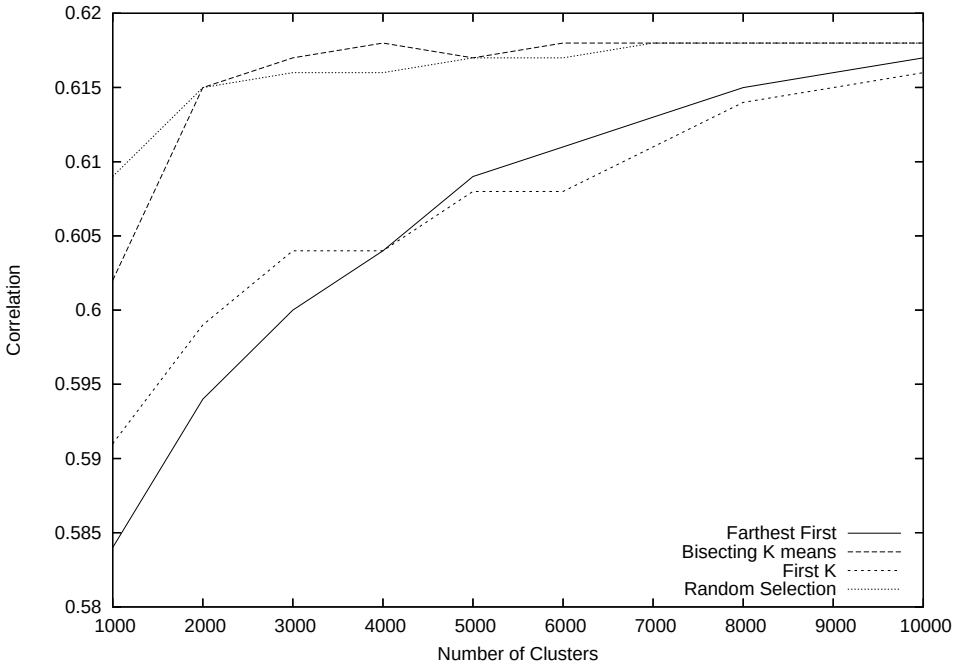

Figure 5.11: **house8L-LinearRegression**

Linear Regression on the house8L dataset is not very accurate. Both Bisecting K-

Table 5.11: house8L-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.618±0.007	1000	0.584±0.008	0.602±0.014	0.591±0.019	**0.609±0.016**
0.618±0.007	2000	0.594±0.008	**0.615±0.006**	0.599±0.019	**0.615±0.007**
0.618±0.007	3000	0.600±0.008	**0.617±0.006**	0.604±0.015	**0.616±0.008**
0.618±0.007	4000	0.604±0.007	**0.618±0.007**	0.604±0.018	**0.616±0.009**
0.618±0.007	5000	0.609±0.008	**0.617±0.007**	0.608±0.011	**0.617±0.007**
0.618±0.007	6000	0.611±0.008	**0.618±0.007**	0.608±0.010	**0.617±0.007**
0.618±0.007	7000	**0.613±0.007**	**0.618±0.007**	0.611±0.010	**0.618±0.007**
0.618±0.007	8000	**0.615±0.007**	**0.618±0.007**	**0.614±0.010**	**0.618±0.007**
0.618±0.007	9000	**0.616±0.007**	**0.618±0.007**	**0.615±0.009**	**0.618±0.007**
0.618±0.007	10000	**0.617±0.007**	**0.618±0.007**	**0.616±0.008**	**0.618±0.007**

Means and Random Selection perform well on this dataset as shown by Figure 5.11 neither showing a statistically significant difference from the performance of the Linear Regression on this dataset except Bisecting K-Means at one thousand clusters. First K gets worse as the amount of clustering increases. The variance in the prediction of First K also increases as the number of clusters falls as can be seen from the increasing standard deviations seen in Table 5.11. This is consistent with First K's poor performance on numeric datasets as seen throughout this experiment. Farthest First also does poorly, becoming statistically significantly worse than the baseline at six thousand. Its fall follows a similar path to First K but it actually passes First K for the smaller numbers of clusters.

5.1.12 house8L-M5

Table 5.12: house8L-M5

Baseline	Clusters	FF	BI	FK	RN
0.791±0.011	1000	0.730±0.022	0.699±0.061	0.474±0.046	0.733±0.015
0.791±0.011	2000	0.759±0.010	0.744±0.023	0.475±0.067	0.740±0.024
0.791±0.011	3000	0.767±0.012	0.757±0.012	0.496±0.059	0.752±0.015
0.791±0.011	4000	0.778±0.008	0.765±0.025	0.515±0.023	0.761±0.011
0.791±0.011	5000	0.781±0.005	0.765±0.029	0.530±0.014	0.775±0.010
0.791±0.011	6000	**0.785±0.005**	0.770±0.034	0.549±0.016	0.778±0.006
0.791±0.011	7000	**0.786±0.007**	**0.786±0.005**	0.567±0.019	**0.787±0.011**
0.791±0.011	8000	**0.791±0.007**	0.781±0.009	0.601±0.013	**0.786±0.007**
0.791±0.011	9000	**0.791±0.010**	**0.786±0.009**	0.625±0.011	**0.787±0.008**
0.791±0.011	10000	**0.790±0.007**	**0.776±0.033**	0.669±0.010	**0.790±0.009**

Using M5 with the house8L dataset results in much higher correlations than Linear-Regression. Farthest First is the standout clusterer, achieving statistically similar results to the baseline when there are more than 6000 clusters. The overall shape produced by

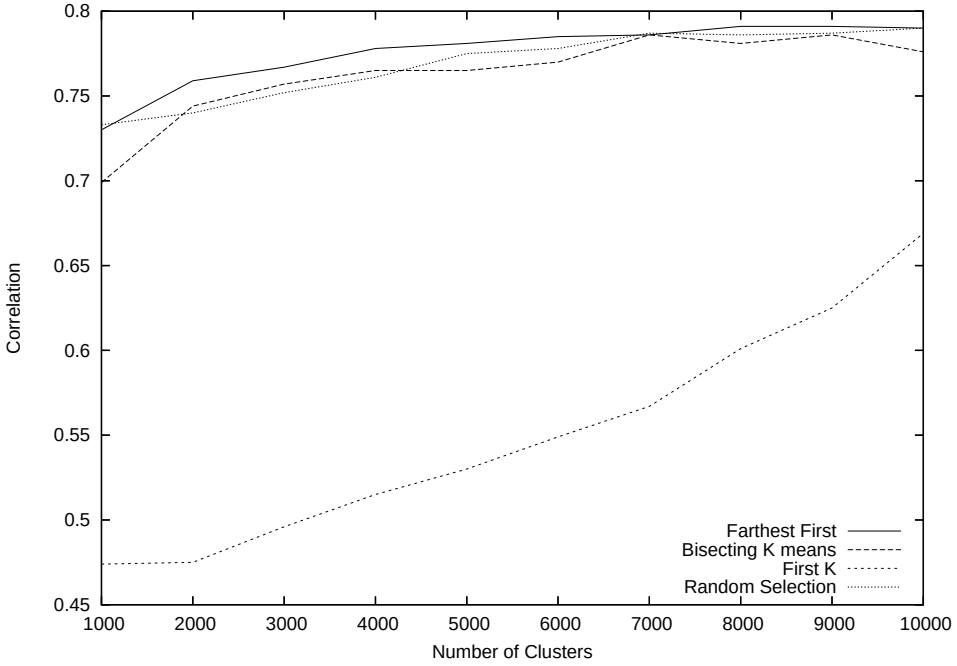

Figure 5.12: **house8L-M5**

Farthest First is similar to those of Bisecting K-Means and Random Selection both of which perform well above seven thousand clusters, although Bisecting K-Means is slightly worse. First K performs significantly worse on this dataset even with minimal clustering. This performance as seen in Figure 5.12 is actually worse than its performance under Linear Regression.

This dataset again shows the effects of different clustering methods upon the classifier. Bisecting K-Means along with Random Selection can achieve over eighty percent data reduction without loss of classification accuracy under Linear Regression. However using M5 both of these clustering methods cannot achieve equivalent results to the baseline with more than thirty percent summarisation. Farthest First is much more stable achieving thirty percent with Linear Regression and forty percent summarisation with M5.

5.1.13 CorelFeatures-LayoutHistogram-LinearRegression

Layout Histogram is a difficult dataset to classify using Linear Regression, with the low baseline performance of 0.161± 0.003 there is little room for the clustered data to perform worse. First K does very poorly on this dataset. Figure 5.13 shows the correlations obtained with different numbers of clusters. Below 25,000 First K is significantly worse than both the base classification and the classifications obtained with Bisecting K-Means and Random Selection. Farthest First is for all numbers of clusters significantly worse than the baseline. However it is also the most consistent, with a much smaller range of values

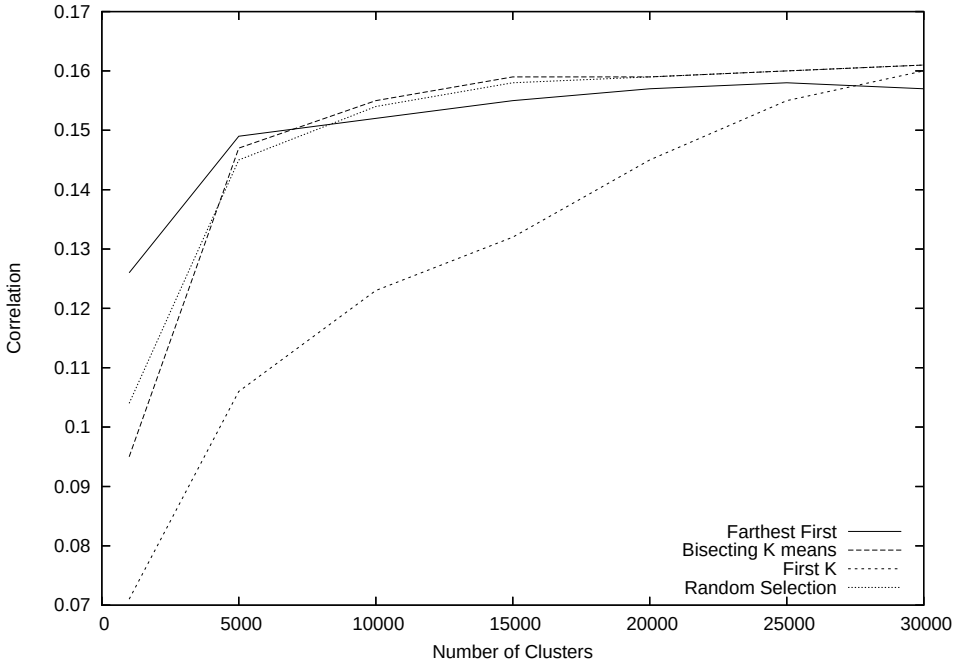

Figure 5.13: **CorelFeatures-LayoutHistogram-LinearRegression**

Table 5.13: CorelFeatures-LayoutHistogram-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.161±0.003	1000	0.126±0.012	0.095±0.019	0.071±0.023	0.104±0.009
0.161±0.003	5000	0.149±0.005	0.147±0.005	0.106±0.019	0.145±0.007
0.161±0.003	10000	0.152±0.003	0.155±0.004	0.123±0.014	0.154±0.004
0.161±0.003	15000	0.155±0.003	**0.159±0.004**	0.132±0.010	0.158±0.004
0.161±0.003	20000	0.157±0.003	**0.159±0.004**	0.145±0.007	**0.159±0.004**
0.161±0.003	25000	0.158±0.003	**0.160±0.004**	0.155±0.004	**0.160±0.004**
0.161±0.003	30000	0.157±0.003	**0.161±0.003**	**0.160±0.003**	**0.161±0.003**

than the other clustering methods. Bisecting K-Means is essentially equivalent to Random Selection for all cluster sizes. Above 15,000, which represents fifty percent summarization, they are both not significantly different from the base classification. Below this point both fall off to some degree. The only significant difference between the two is at the one thousand cluster level, where Bisecting K-Means does slightly worse, as seen in Table 5.13.

5.1.14 CorelFeatures-LayoutHistogram-M5

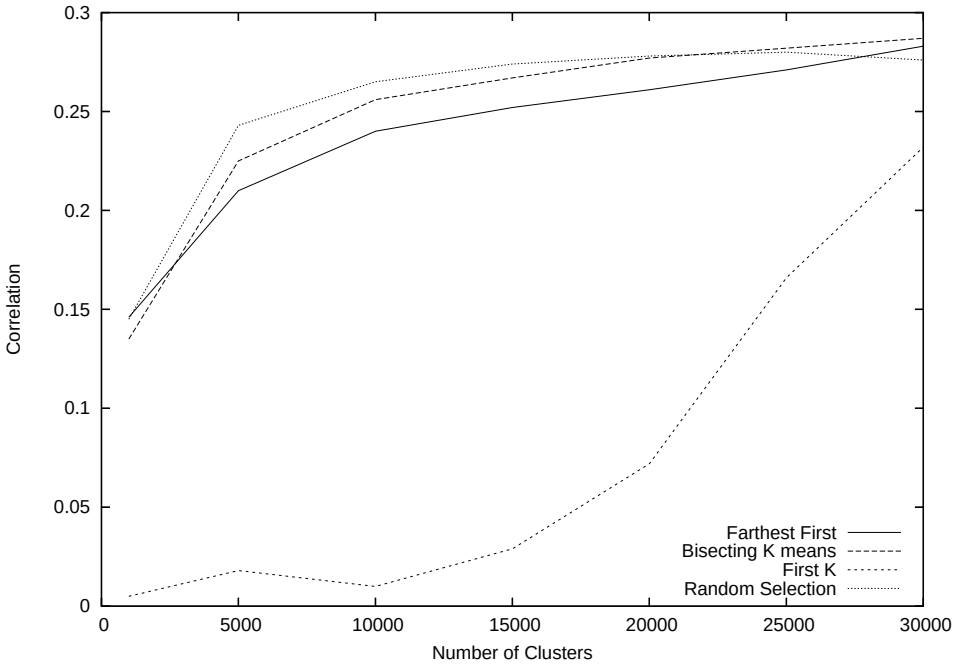

Figure 5.14: **CorelFeatures-LayoutHistogram-M5**

Table 5.14: CorelFeatures-LayoutHistogram-M5

Baseline	Clusters	FF	BI	FK	RN
0.285±0.011	1000	0.146±0.037	0.135±0.026	0.005±0.032	0.145±0.040
0.285±0.011	5000	0.210±0.026	0.225±0.013	0.018±0.011	0.243±0.009
0.285±0.011	10000	0.240±0.008	0.256±0.010	0.010±0.013	0.265±0.008
0.285±0.011	15000	0.252±0.006	0.267±0.011	0.029±0.012	0.274±0.006
0.285±0.011	20000	0.261±0.008	**0.277±0.011**	0.072±0.013	**0.278±0.010**
0.285±0.011	25000	0.271±0.013	**0.282±0.009**	0.166±0.042	**0.280±0.010**
0.285±0.011	30000	**0.283±0.012**	**0.287±0.008**	0.232±0.082	**0.276±0.034**

M5 produces a much more accurate classification for the Layout Histogram dataset than Linear Regression. Figure 5.14 shows a similar picture to Figure 5.13. With more than 20,000 clusters Bisecting K-Means is not significantly different from the base classi-

fication. It is also not significantly different from Random Selection. Although Table 5.14 shows that the variance on Bisecting K-Means is much less than Random Selection above 20,000. First K does very poorly with M5, as it did on previous numeric datasets. In fact for a cluster size of one thousand the range encompassed within a standard deviation of the First K value includes negative correlations. Farthest First is not as good as Random Selection or Bisecting K-Means anywhere except with 30,000 clusters.

The only clustering method to achieve reasonable results on this dataset is Bisecting K-Means which achieves between thirty three and fifty percent summarisation depending on which classification method is employed. Both Farthest First and First K perform very poorly on this dataset. Bisecting Means is only just better than Random Selection.

5.1.15 CorelFeatures-CoocTexture-LinearRegression

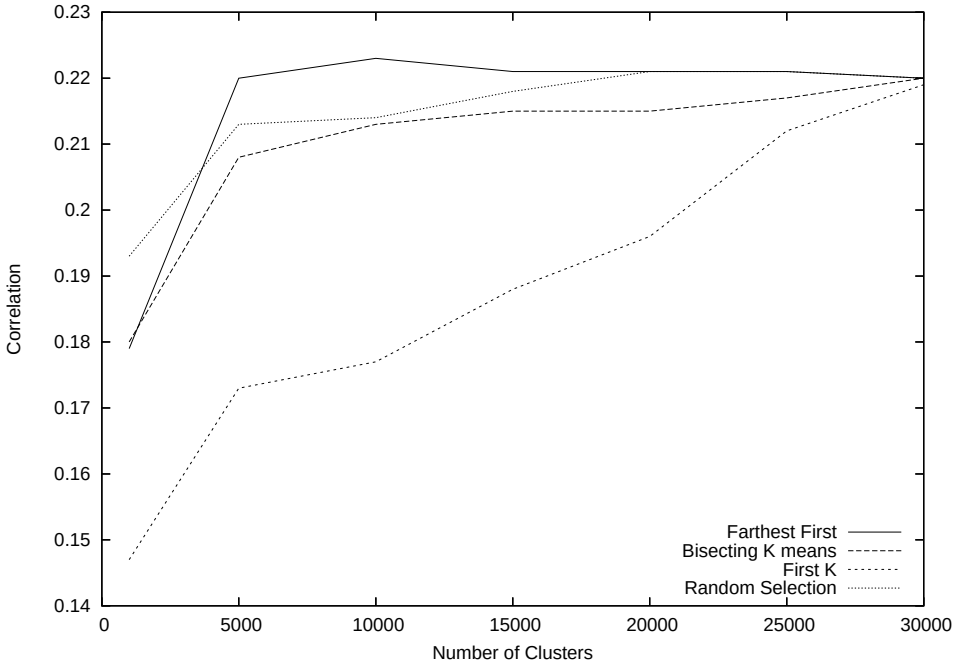

Figure 5.15: **CorelFeatures-CoocTexture-LinearRegression**

CoocTexture produces better correlations under Linear Regression than Layout Histogram. Farthest First clustering produces the best clusters overall, not statistically significantly different from the baseline anywhere except with one thousand clusters. Figure 5.15 shows that Random Selection matches Farthest First almost exactly with more than 15,000 clusters, overall Random Selection is not significantly different from Bisecting K-Means. First K performs better on this dataset than for the Layout Histogram dataset although Table 5.15 shows that the variance on First K's predictions is significantly greater than for the other methods on this dataset. First K is never more than three percent worse

Table 5.15: CorelFeatures-CoocTexture-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.220±0.009	1000	0.179±0.016	0.180±0.012	0.147±0.025	0.193±0.010
0.220±0.009	5000	**0.220±0.004**	0.208±0.004	0.173±0.013	0.213±0.006
0.220±0.009	10000	**0.223±0.008**	0.213±0.003	0.177±0.014	0.214±0.005
0.220±0.009	15000	**0.221±0.008**	**0.215±0.005**	0.188±0.012	**0.218±0.008**
0.220±0.009	20000	**0.221±0.009**	**0.215±0.007**	0.196±0.009	**0.221±0.010**
0.220±0.009	25000	**0.221±0.009**	**0.217±0.006**	**0.212±0.012**	**0.221±0.009**
0.220±0.009	30000	**0.220±0.009**	**0.220±0.009**	**0.219±0.010**	**0.220±0.009**

than any other method.

5.1.16 CorelFeatures-CoocTexture-M5

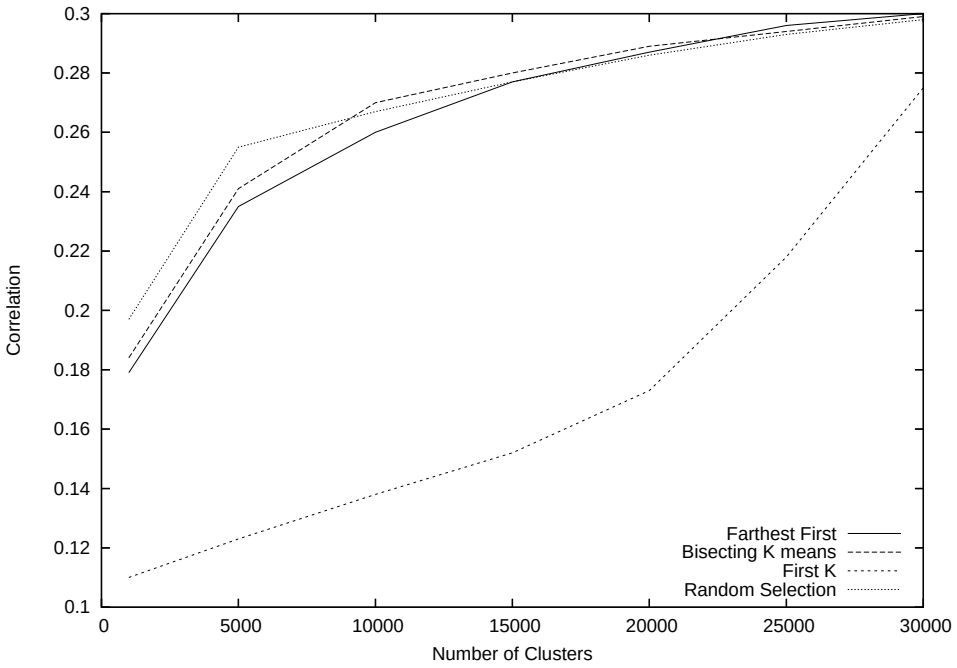

Figure 5.16: **CorelFeatures-CoocTexture-M5**

The improvement achieved by M5 is much less marked on this dataset. As Figure 5.16 shows, the classification accuracy suffers when any amount of clustering is done. Farthest First, Bisecting K-Means and Random Selection are only statistically similar to the baseline at 30,000 clusters. The correlation achieved from their clusters descends smoothly to 10,000 clusters losing only four percent. These three methods are statistically indistinguishable with this classifier. The classification accuracy of M5 suffers heavily with First K clustering, although First K performs markedly better than on the previous dataset and still maintains eleven percent correlation with one thousand clusters.

Table 5.16: CorelFeatures-CoocTexture-M5

Baseline	Clusters	FF	BI	FK	RN
0.301±0.003	1000	0.179±0.016	0.184±0.018	0.110±0.012	0.197±0.015
0.301±0.003	5000	0.235±0.006	0.241±0.008	0.123±0.012	0.255±0.003
0.301±0.003	10000	0.260±0.006	0.270±0.006	0.138±0.008	0.267±0.006
0.301±0.003	15000	0.277±0.007	0.280±0.008	0.152±0.007	0.277±0.004
0.301±0.003	20000	0.287±0.005	0.289±0.004	0.173±0.008	0.286±0.006
0.301±0.003	25000	0.296±0.007	0.294±0.006	0.218±0.006	0.293±0.002
0.301±0.003	30000	**0.300±0.005**	**0.299±0.002**	0.275±0.007	**0.298±0.005**

Farthest First is the overall best clustering method on this dataset, achieving over eighty percent summarisation under Linear Regression. It is not however able to achieve any summarisation when used with M5. Bisecting K-Means and Random Selection perform similarly under Linear Regression as they did on the Layout Histogram dataset. However they are unable to continue this accuracy with M5

5.1.17 CorelFeatures-ColorMoments-LinearRegression

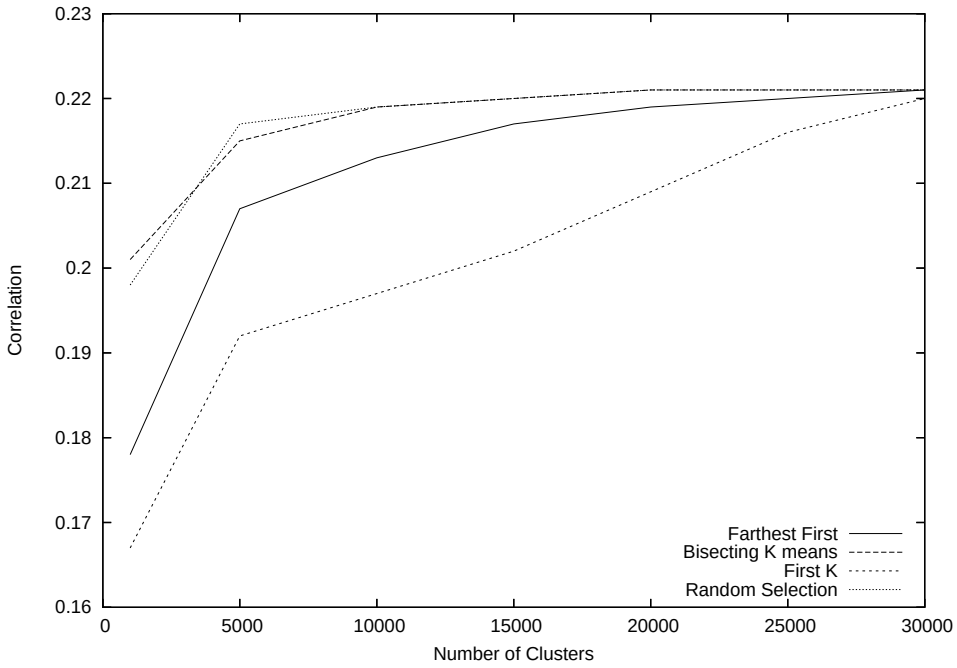

Figure 5.17: **CorelFeatures-ColorMoments-LinearRegression**

Color Moments under Linear Regression also clusters well. Figure 5.17 is very similar in shape to Figure 5.15. In this case however there is no clear winner with Bisecting K-Means and Random Selection being totally indistinguishable with more than 5000 clusters. Farthest First does worse than these two. Random Selection comes out best with no

Table 5.17: CorelFeatures-ColorMoments-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.221±0.005	1000	0.178±0.013	0.201±0.006	0.167±0.019	0.198±0.011
0.221±0.005	5000	0.207±0.007	0.215±0.006	0.192±0.010	**0.217±0.005**
0.221±0.005	10000	0.213±0.006	**0.219±0.006**	0.197±0.009	**0.219±0.005**
0.221±0.005	15000	**0.217±0.006**	**0.220±0.006**	0.202±0.006	**0.220±0.005**
0.221±0.005	20000	**0.219±0.005**	**0.221±0.005**	0.209±0.005	**0.221±0.005**
0.221±0.005	25000	**0.220±0.005**	**0.221±0.006**	0.216±0.005	**0.221±0.005**
0.221±0.005	30000	**0.221±0.005**	**0.221±0.005**	**0.220±0.005**	**0.221±0.005**

statistically significantly difference from the baseline when 5000 or more instances are selected. Bisecting K-Means reaches this point at 10,000 clusters and Farthest First at 15,000. First K while not statistically significantly different at 30,000 clusters falls away quickly as the number of clusters decreases.

5.1.18 CorelFeatures-ColorMoments-M5

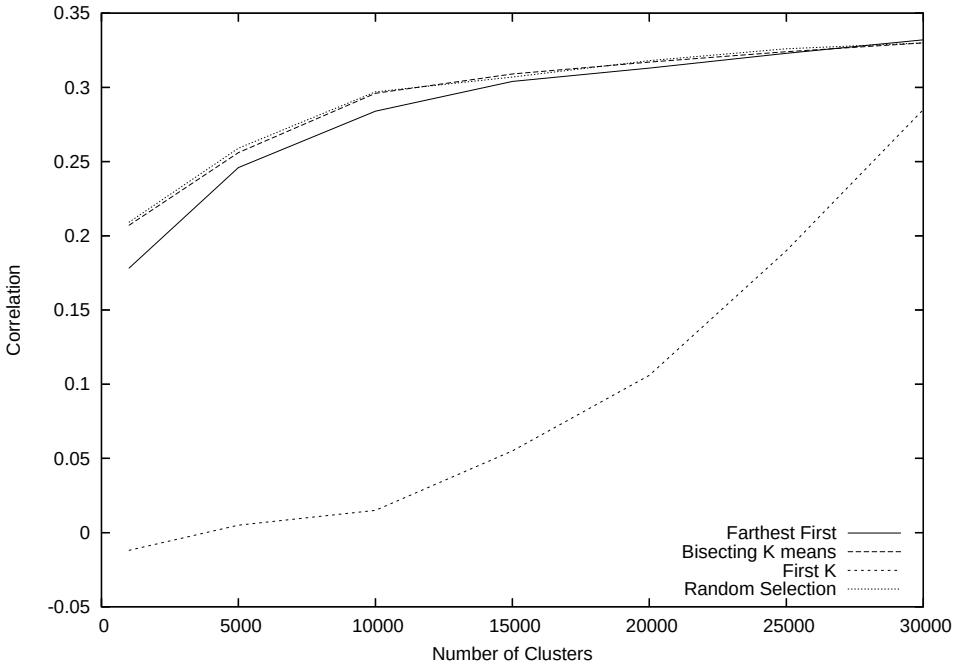

Figure 5.18: **CorelFeatures-ColorMoments-M5**

M5 when used with the Color Moments dataset produces results similar to those for the Cooc Texture dataset. At 30,000 clusters the correlation for Farthest First, Bisecting K-Means and Random Selection are not statistically different from the base classification, see Table 5.18. Bisecting K-Means in this case mirrors Random Selection through the entire range of clusters, Farthest First trails off slightly. First K performs very poorly on

Table 5.18: CorelFeatures-ColorMoments-M5

Baseline	Clusters	FF	BI	FK	RN
0.333±0.004	1000	0.178±0.013	0.207±0.014	-0.012±0.021	0.209±0.018
0.333±0.004	5000	0.246±0.024	0.256±0.010	0.005±0.013	0.259±0.017
0.333±0.004	10000	0.284±0.015	0.296±0.008	0.015±0.018	0.297±0.005
0.333±0.004	15000	0.304±0.006	0.309±0.008	0.055±0.014	0.307±0.006
0.333±0.004	20000	0.313±0.010	0.317±0.006	0.106±0.015	0.318±0.006
0.333±0.004	25000	0.323±0.005	0.324±0.007	0.190±0.007	0.326±0.004
0.333±0.004	30000	**0.332±0.005**	**0.330±0.005**	0.285±0.006	**0.330±0.004**

this dataset with this clusterer. It falls from twenty eight per cent correlation to a negative one percent correlation as the amount of clustering increases.

First K with Linear Regression performed seventeen per cent better than with M5. The poor results are probably due to interactions with the classifier rather than the dataset. The greatest of these interactions is likely M5's lack of support for weighted instances. Random Selection is the overall winner on this dataset because it can reduce the dataset by over eighty percent without significant loss of classification accuracy using Linear Regression. Bisecting K-Means and Farthest First also perform well with Linear Regression, achieving over fifty percent summarisation No significant summarisation can be achieved with the M5 classifier.

5.1.19 CorelFeatures-ColorHistogram-LinearRegression

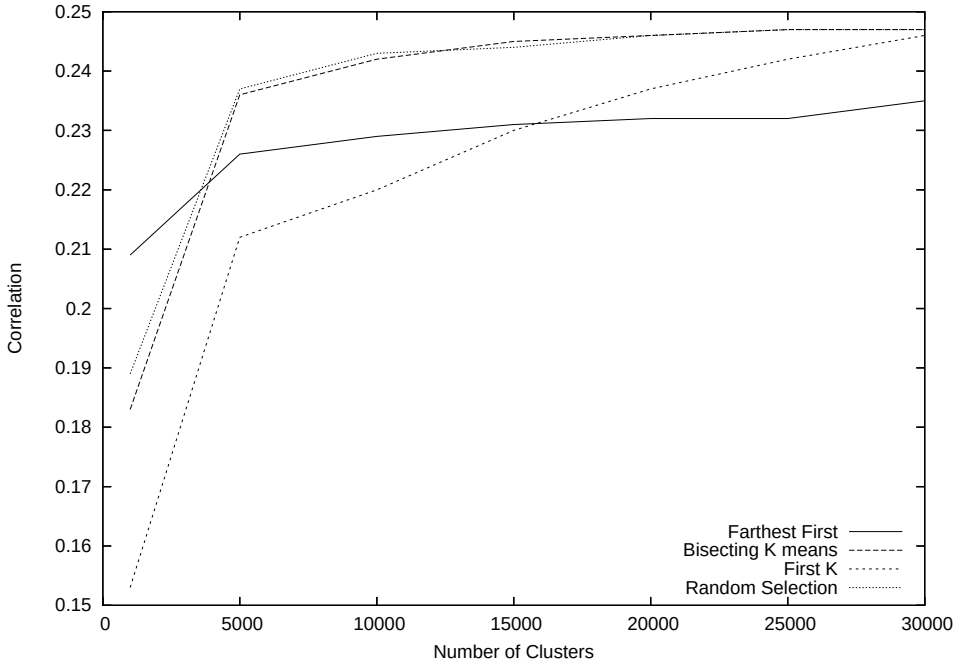

Figure 5.19: **CorelFeatures-ColorHistogram-LinearRegression**

Table 5.19: CorelFeatures-ColorHistogram-LinearRegression

Baseline	Clusters	FF	BI	FK	RN
0.247±0.002	1000	0.209±0.013	0.183±0.020	0.153±0.025	0.189±0.017
0.247±0.002	5000	0.226±0.005	0.236±0.004	0.212±0.010	0.237±0.003
0.247±0.002	10000	0.229±0.005	0.242±0.003	0.220±0.009	0.243±0.002
0.247±0.002	15000	0.231±0.004	0.245±0.002	0.230±0.004	0.244±0.002
0.247±0.002	20000	0.232±0.006	**0.246±0.002**	0.237±0.003	**0.246±0.001**
0.247±0.002	25000	0.232±0.004	**0.247±0.002**	0.242±0.002	**0.247±0.002**
0.247±0.002	30000	0.235±0.003	**0.247±0.002**	**0.246±0.002**	**0.247±0.002**

Linear Regression performs better on the Color Histogram dataset than on any of the other Corel datasets. Figure 5.19 shows Bisecting K-Means again tracking Random Selection. Both of these datasets are not statistically different from the base correlations with more than 20,000 clusters. Below 10,000 both fall away rapidly. Farthest First performs consistently poorly on this dataset, even being beaten by First K above 15,000. First K does better on this dataset than on other Corel datasets. At 30,000 its values are not statistically worse than the base classification. Below this point they descend gradually tracking a similar but lower path to the other clusterers. This pattern is embodied in Table 5.19 , where the values of First K fall more rapidly the variance also increases markedly. This indicates the influence of the randomisation order on the clusters generated by First K.

5.1.20 CorelFeatures-ColorHistogram-M5

Table 5.20: CorelFeatures-ColorHistogram-M5

Baseline	Clusters	FF	BI	FK	RN
0.287±0.053	1000	0.230±0.024	0.158±0.051	0.002±0.011	0.172±0.042
0.287±0.053	5000	**0.274±0.023**	**0.268±0.016**	0.014±0.016	**0.246±0.068**
0.287±0.053	10000	**0.288±0.039**	**0.261±0.076**	0.041±0.016	**0.228±0.112**
0.287±0.053	15000	**0.275±0.052**	**0.241±0.076**	0.076±0.009	**0.222±0.118**
0.287±0.053	20000	**0.309±0.049**	**0.254±0.056**	0.122±0.018	**0.273±0.069**
0.287±0.053	25000	**0.334±0.011***	**0.270±0.078**	0.193±0.062	**0.294±0.044**
0.287±0.053	30000	**0.308±0.038**	**0.264±0.053**	**0.274±0.036**	**0.233±0.112**

M5 with the Color Histogram dataset produces results which are incredibly variable. Farthest First is the overall best performing method. A striking feature of these results is the seemingly large standard deviations and variable means seen in Figure 5.20. These are notable when compared to the smooth curves for Linear Regression. Both Random Selection and Bisecting K-Means change direction every second cluster size. Of interest

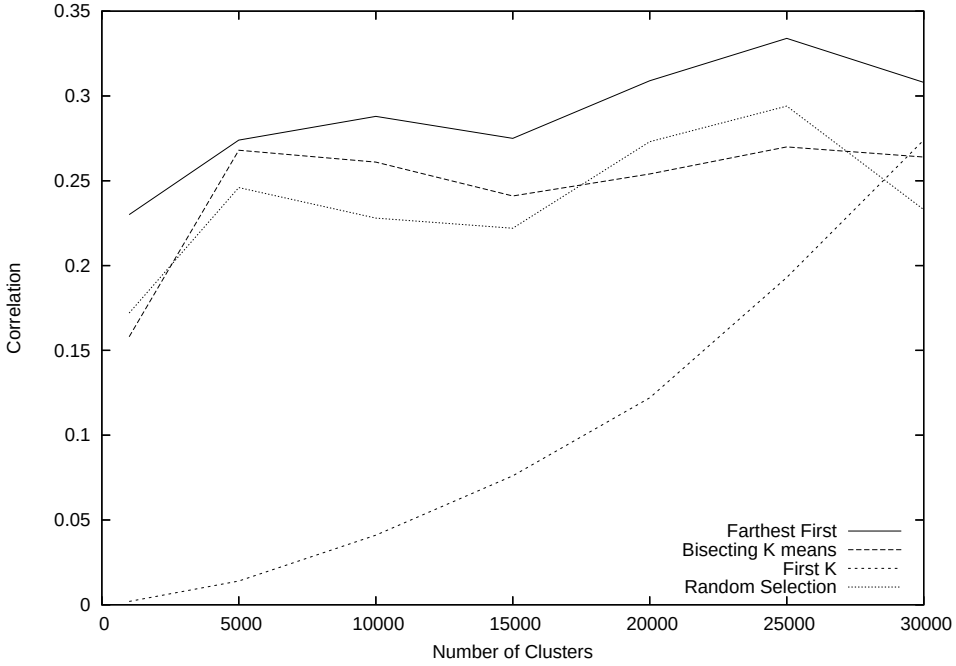

Figure 5.20: **CorelFeatures-ColorHistogram-M5**

here are the local minimum at 15,000 and the two maxima at 5000 and 25,000. The difference from 25,000 to 15,000 is three percent for Bisecting K-Means and seven per cent for Random Selection, both much larger than the standard deviations within this range. Similarly between five thousand and 15,000 there is a two per cent difference which exceeds the standard deviation. First K performs similarly with this classifier as it did with the same classifier on other datasets. However its performance exceeds Bisecting K-Means at 30,000 by one percent, at one thousand it is essentially zero.

Unlike the other Corel datasets this dataset is highly summarise-able under M5 and only marginally so under Linear Regression. Farthest First is the clear winner among the clustering methods, achieving over eighty percent summarisation and with 25,000 clusters actually improving on the base classification. Both Bisecting K-Means and Random Selection achieve over eighty percent summarisation as well, however these methods never produce results which are better than the base classification. They do both however achieve thirty percent summarisation on Linear Regression where Farthest First cannot even match the baseline when no clustering is required.

5.2 Summary

With numeric datasets the same measures of success apply as with the nominal datasets. The key measure of the usefulness of a clustering method in these experiments is the

amount of summarisation the clusterer can achieve on the dataset before the correlation obtained by the classifier used for testing is reduced. Table 5.21 shows the maximum summarisation attained by each clustering method when using the simple classifier Linear Regression. The clustering methods are ranked with the method achieving most summarisation to the left and the method achieving least summarisation on the right the third column shows the actual summarisation scores achieved by each clusterer before accuracy was reduced by a statistically significant amount. Improving on the base classification was not taken into account in this evaluation. Table 5.22 shows the same information for the more complex classifier M5. This information was tabulated in the same fashion.

Datasets	Clusterer Ranking	Reduction Achieved
ailerons	(FF, RN, EM, BI, KM, FK)	(85%, 85%, 85%, 85%, 71%, 42%)
kim8nm	(EM, BI-KM, RN, FF, FK)	(87%, 87%, 75%, 75%, 12.5%)
2Dplanes	(FF, RN, BI, FK)	(87%, 75%, 62%, 50%)
fried	(BI-RN, FK-FF)	(87%, 75%)
mv	(FF-RN, BI, FK)	(87%, 62%, 12.5%)
house8L	(RN, BI, FF, FK)	(90%, 81%, 36%, 27%)
Layout Histogram	(BI, RN, FK, FF)	(55%, 41%, 11%, -)
Cooc Texture	(FF, RN-BI, FK)	(84%, 69%, 24%)
Color Moments	(RN, BI, FF, FK)	(85%, 70%, 55%, 11%)
Color Histogram	(BI-RN, FK, FF)	(41%, 11%, -)

Table 5.21: Linear Regression: Maximum summarisation for each dataset

Under Linear Regression there are several clusterers that can achieve very high levels of summarisation. Farthest First achieves the best summarisation on four datasets, Bisecting K-Means on three, Random Selection on two, and EM on one, as shown by Table 5.21. In all but two of these datasets the summarisation achieved is over eighty percent. On the two datasets where no clusterers could achieve over fifty percent, Farthest First failed to match the base correlation even when asked for as many clusters as instances. First K only achieves more than fifty percent summarisation on the fried dataset. Its average performance of less than thirty percent summarisation makes it the worst performing clustering method. With the exception of the poor performance of First K the results in Table 5.21 are very similar to those in Table 4.19.

On the complex classifier, M5, summarised in Table 5.22, the best clustering method is very clear. Farthest First is the best performing clustering method in all datasets except Layout Histogram. Overall it can achieve an average of thirty-nine percent summarisation on these datasets. The values shown in Table 5.22 are much lower than those in Table 5.21 above. This could be due to a number of factors such as M5's only partial support for weights. The better results achieved by M5 as a baseline make summarisation more

Datasets	Clusterer Ranking	Reduction Achieved
ailerons	(FF-BI-RN-KM, FK, EM)	(23%, 0%, -)
kim8nm	(FF, BI, RN-KM, FK, EM)	(25%, 25%, 12%, 0%, -)
2Dplanes	(FF-RN-BI, FK)	(87%, -)
fried	(FF-BI-RN, FK)	(0%, -)
mv	(FF, RN, BI, FK)	(87%, 75%, 50%, 0%)
house8L	(FF, BI, RN, FK)	(45%, 36%, 36%, -)
Layout Histogram	(BI-RN, FF, FK)	(39%, 10%, -)
Cooc Texture	(FF-BI-RN, FK)	(10%, -)
Color Moments	(FF-BI-RN, FK)	(10%, -)
Color Histogram	(FF, BI, RN, FK)	(85%, 85%, 85%, 10%)

Table 5.22: M5: Maximum summarisation for each dataset

difficult since it is using more features of the data than Linear Regression.

The same preliminary experiments, as reported in Chapter 4, evaluating the unweighted classifier and the effectiveness of clustering as a method for outlier detection were run as well. As in the nominal case further investigation is required for numeric targets.

Overall on numeric datasets it can be seen that the cluster classifier can obtain better results than Random Selection in most cases, since in only two cases was Random Selection found to be the most effective method of data summarisation.

Chapter 6
Related Work

There are two related research areas that can broadly be described as Data Summarisation. The older of the two is Instance Selection, where the goal it to pick out from the dataset the most representative data points. The other is Data Squishing where, rather than select from existing data points, new meta-data points are created. The goal it to ensure that the smaller dataset has the same statistical properties as the original larger dataset. A selection of the papers in each of these two branches is illustrated in Figure 6.1. Figure 6.1 shows the relation of the Cluster Classifier to other data summarisation techniques. The Cluster Classifier falls under the data squishing branch as it creates meta-data points rather than selecting from the original dataset.

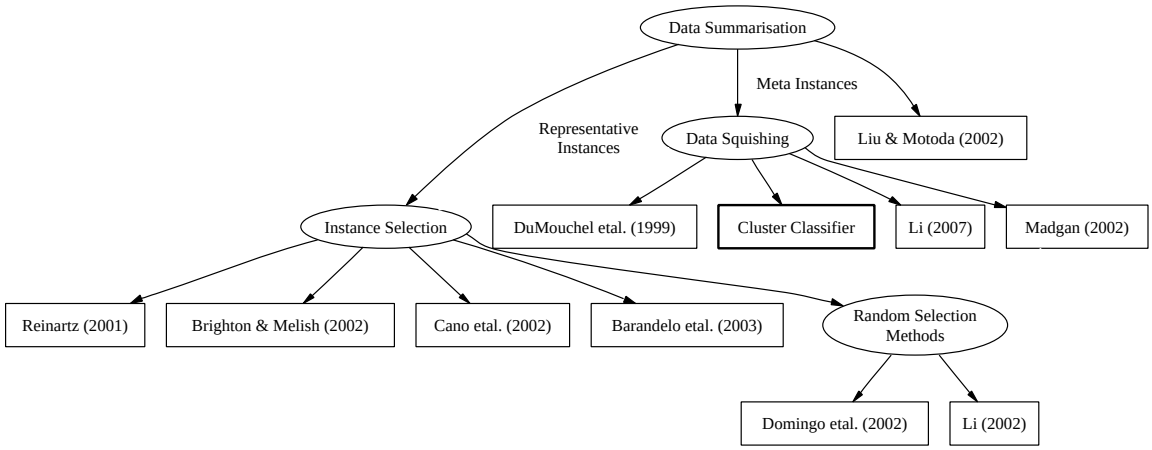

Figure 6.1: **Related Work**

"On Issues of Instance Selection" (Liu & Motoda, 2002) provides a broad overview of the tasks involved in data summarisation. They identify three purposes for data summarisation: Enabling, Focusing, and Cleaning. Enabling is reducing a dataset to a point where, irrespective of time or memory, expensive algorithms can be feasibly applied to it. Focusing is removing extraneous data points from the dataset, leaving only those that apply to the classification problem. Cleaning is removing outliers and misclassified data points (noise) from the data to improve the classification accuracy. These goals are the

same as those that motivated the Cluster Classifier in Chapter 2. The Cluster Classifier explicitly set out to achieve the enabling functions and as (Liu & Motoda, 2002) suggest it should achieves the other functions at the same time.

6.1 Data Squishing

The major work in the data Squishing area is "Squishing Flat Files Flatter" (DuMouchel, Volinsky, Johnson, Cortes & Pregibon, 1999). In this paper the authors propose a three stage framework for data squishing, Group, Moments and Generate (GMG). The first of these stages consists of grouping data points. They accomplish this by using bins to separate the data on categorical(nominal) attributes. In order to be able to separate data on continuous(numeric) attributes they employ two methods hyper rectangles and data spheres. The moments stage uses a Taylor series approximation to generate calculate the moments on the data in each bin. Finally they generate a new smaller dataset with the same moments as the original. To do this they have to create several data points from most bins. Each data point created is weighted by the support it has from instances in the bin from which it was generated. In their future work they suggest that clustering algorithms as used in the Cluster Classifier could be useful in the first stage of the pipeline however they insist that this would need to be followed up by the other two stages. In the Cluster Classifier we do without these complex calculation stages by taking only one meta-data point from each cluster and still maintain reasonable classification accuracy in most cases.

"Support Cluster Machine" (Li, Chi, Fan & Xue, 2007) addresses a similar problem to the Cluster Classifier. The authors concentrate on providing a way for clustering of the dataset to reduce it's size for use with a complex classifier, in their case the Support Vector Machine. They extract more information from each cluster to produce a covariance matrix in addition to the mean and weights, generated from the cluster support, used by the Cluster Classifier. Their work has a much narrower application since they use the clusters directly in a Support Vector Machine. The only clustering method they employed that was also used with the Cluster Classifier was EM. They also found that it was not a satisfactory clustering method. The main contribution of this paper is the use of a probability product kernel for computing the distance between two distributions (clusters).

"Likelihood based Data Squishing" (Madigan, Raghavan, Dumouchel, Nason, Posse & Ridgeway, 2002) takes a similar approach. Instead of grouping the data and then analysing the groups the authors first build a bayesian statistical model of the dataset and

then partition afterwards based on the model. The partitioning is done using likelihood profiles. Like above the clusters created are then analysed and one or more meta-data points are created from them. The meta-data points are weighted according to support. A key requirement of the new smaller set of data points is that it fits the same model built from the larger set at the beginning of the process. This ensures statistical analysis of the smaller dataset will yield equivalent results.

6.2 Instance Selection

Instance Selection uses real instances rather than meta instances. These papers describe techniques designed to select the best representatives from amongst the original dataset and discard the rest. There are many methods used to accomplish this. (Reinartz, 2002) presents a framework for the evaluation of instance selection as a focusing task, with the idea of defining the goodness of a solution. (Brighton & Mellish, 2002) looks at instance selection from a nearest Neighbour perspective. From this standpoint the key reasons for selecting instances is to remove noise and data points that are not useful in the classification process in order to reduce the number of data points that the nearest neighbour classifier must consider in making a classification. They use two methods of instance selection, the BIRCH (Zhang, Ramakrishnan & Livny, 1996) algorithm and hierarchical clustering using OPTICS (Ankerst, Breunig, Kriegel & Sander, 1999). To overcome the problems these methods have with loss of information, such as structural and size distortion, they employ Data Bubbles. These allow them to select the appropriate instance as a representative of the cluster rather than using what they term the naive approach of selecting the cluster centre or data point closest to the centre. This is computationally more expensive than the Cluster Classifier which selects the cluster centre. Their method is also only applicable to hierarchical clustering making it much less generic than the Cluster Classifier framework.

Both (Cano, Herrera & Lozano, 2003) and (Barandela, Valdovinos & Sánchez, 2003) apply classification algorithms to determine which data points are important. Cano uses evolutionary algorithms in order to generate the best set of data points for a particular task. They look specifically at two tasks, prototype selection for a nearest Neighbour classifier and training set selection. They conclude that using evolutionary algorithms on both of these tasks does in fact result in a small dataset that is a more accurate sample from the original dataset than those produced by other algorithms. (Barandela, Valdovinos & Sánchez, 2003) use an ensemble of simple classifiers to determine which data points are the key members of the dataset. These members are then used as the members of the

reduced dataset. The authors motivation was the same as the Cluster Classifier although this is only a minor section of their work.

Random sampling is a subset of instance selection methods. Apart from a simple random selection from the dataset, which is what various clusterers are compared against in Chapter 4 and 5 above there are several methods that attempt to make a more intelligent selection from the dataset. (LI, 2002) takes an initially random selection from the dataset. The authors then take additional data points as candidates for exchange with data points already in the smaller dataset. A chi-square criterion is used to compare the goodness of the dataset as it stands or with the data point replaced with the candidate. If this goodness measure produces a better result using the candidate then the two data points are swapped. (Domingo, Gavaldà & Watanabe, 2002) also begin with random samples rather than using a fixed size they contend that better reduction is achieved by continuing selection until a stopping criterion is met, which should in theory provide the optimal sample size. Their method is designed to be a fast online approach to adaptive random sampling. Each data point is considered and then either retained or not retained as the goodness characteristic dictates. The stopping criterion is the Hoeffding bound, when this is reached the selection stops and no further data points are examined. This makes this algorithm very good on extremely large datasets. However in the worst case it is very poor since it can select almost an entire dataset.

Chapter 7
Conclusions

The aim of this thesis was to demonstrate that it is possible to use clustering algorithms to summarise large datasets by using the cluster centroids to create a more compact representation of the dataset. Since clustering is an inherently more complex process than random selection it needed to be demonstrated that the meta-data points generated by clustering had the potential to be more effective when used in classification than an equivalent size random selection.

The results from Chapters 4 and 5 show that the Cluster Classifier is an effective method of data summarisation. In many cases clustering achieves over fifty percent summarisation which is a very significant reduction on the large datasets used for the experiments. Even more significantly, in only two of the thirty eight experiment runs, was the amount of summarisation achieved by the clustered data worse than the summarisation achieved by random selection. This means that clustering is better than random selection in almost ninety five percent of datasets. This is strong evidence for the effectiveness of this method.

The results do show, however, that when using the Cluster Classifier it is important to select a clusterer carefully. Three clusterers in the experiments above have demonstrated their usefulness within this framework. Farthest First, Bisecting K Means and First K all provide satisfactory performance on some datasets. However none of these clustering methods stands out as always better than the others. The best clustering method varies depending on the structure of the dataset being clustered. An obvious example of this, which can be observed from Chapter 5, is First K's inability to provide effective results on numeric datasets. This very naive clustering method achieves all its better results with nominal datasets.

The Cluster Classifier also shows promise as a method of noise detection and reduction. However this was only briefly examined and should be investigated further.

7.1 Future Work

Further experiments to test the Cluster Classifier's effectiveness as a noise detection and /or reduction technique should be conducted. In addition there are a number of things that could be tried to improve the performance of the Cluster Classifier. Currently the Cluster Classifier only looks at partitioning the nominal data. Numeric data is clustered directly. This is reflected in the poorer summarisation achieved in Chapter 5 relative to that achieved in Chapter 4. To counter this the Cluster Classifier should include a method for partitioning numeric class values prior to clustering similarly to the methods used by (DuMouchel, Volinsky, Johnson, Cortes & Pregibon, 1999) to partition numeric attributes prior to binning.

The Cluster Classifier could also perform its work more effectively and cluster much larger datasets if it were modified to work with streaming clusterers. These clusterers cluster a data point and then discard it immediately after modifying their internal cluster structure. This allows them to handle arbitrarily large datasets. By combining streaming clusterers with the Cluster Classifier it should be possible to summarise arbitrarily large datasets down to a sufficient size to enable complicated batch classification algorithms to be run effectively on them.

Appendix A
Unweighted Clusters

A.1 Bisecting K-Means

The removal of weights has little influence on Bisecting K-Means. Weights do not seem to be an important factor with this algorithm. This is probably due to the even sizing of the clusters which means that weights are already fairly well encoded by the meta-data points.

Table A.1: Agrawal-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
87.780±0.299	2500	53.229±4.108	53.229±4.108
87.780±0.299	5000	77.519±1.166	77.519±1.166
87.780±0.299	7500	84.683±1.225	84.683±1.225
87.780±0.299	10000	**88.714±1.182***	**88.714±1.182***
87.780±0.299	12500	**89.843±1.398***	**89.843±1.398***
87.780±0.299	15000	**90.152±1.307***	**90.152±1.307***
87.780±0.299	17500	**91.278±1.015***	**91.278±1.015***
87.780±0.299	20000	**90.059±0.824***	**90.059±0.824***

Table A.2: Agrawal-Logistic

Baseline	Clusters	Weighted	Unweighted
67.298±0.002	2500	50.166±1.037	50.166±1.037
67.298±0.002	5000	50.115±0.488	50.115±0.488
67.298±0.002	7500	50.101±0.425	50.101±0.425
67.298±0.002	10000	50.388±0.435	50.388±0.435
67.298±0.002	12500	50.432±0.403	50.432±0.403
67.298±0.002	15000	61.948±1.430	61.948±1.430
67.298±0.002	17500	67.170±0.149	67.170±0.149
67.298±0.002	20000	**67.298±0.002**	**67.298±0.002**

Table A.3: Waveform21-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
80.430±0.228	2500	**82.084±0.223***	**82.084±0.223***
80.430±0.228	5000	**81.732±0.189***	**81.732±0.189***
80.430±0.228	7500	**81.406±0.168***	**81.406±0.168***
80.430±0.228	10000	**81.033±0.222***	**81.033±0.222***
80.430±0.228	12500	**80.811±0.214***	**80.811±0.214***
80.430±0.228	15000	**80.577±0.213**	**80.577±0.213**
80.430±0.228	17500	**80.500±0.221**	**80.500±0.221**
80.430±0.228	20000	**80.419±0.224**	**80.419±0.224**

Table A.4: Waveform21-Logistic

Baseline	Clusters	Weighted	Unweighted
86.607±0.161	2500	85.988±0.213	85.988±0.213
86.607±0.161	5000	86.348±0.158	86.348±0.158
86.607±0.161	7500	86.465±0.177	86.465±0.177
86.607±0.161	10000	**86.530±0.197**	**86.530±0.197**
86.607±0.161	12500	**86.562±0.184**	**86.562±0.184**
86.607±0.161	15000	**86.598±0.173**	**86.598±0.173**
86.607±0.161	17500	**86.608±0.157**	**86.608±0.157**
86.607±0.161	20000	**86.620±0.145**	**86.620±0.145**

Table A.5: Waveform40-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
80.337±0.130	2500	**81.613±0.308***	**81.613±0.308***
80.337±0.130	5000	**81.705±0.209***	**81.705±0.209***
80.337±0.130	7500	**81.412±0.135***	**81.412±0.135***
80.337±0.130	10000	**81.081±0.101***	**81.081±0.101***
80.337±0.130	12500	**80.860±0.110***	**80.860±0.110***
80.337±0.130	15000	**80.612±0.111***	**80.612±0.111***
80.337±0.130	17500	**80.430±0.114**	**80.430±0.114**
80.337±0.130	20000	**80.332±0.130**	**80.332±0.130**

Table A.6: Waveform40-Logistic

Baseline	Clusters	Weighted	Unweighted
86.644±0.130	2500	85.323±0.249	85.323±0.249
86.644±0.130	5000	86.145±0.196	86.145±0.196
86.644±0.130	7500	86.310±0.158	86.310±0.158
86.644±0.130	10000	86.494±0.149	86.494±0.149
86.644±0.130	12500	**86.562±0.159**	**86.562±0.159**
86.644±0.130	15000	**86.617±0.147**	**86.617±0.147**
86.644±0.130	17500	**86.654±0.142**	**86.654±0.142**
86.644±0.130	20000	**86.638±0.124**	**86.638±0.124**

Table A.7: CorelFeatures-LayoutHistogram-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.161±0.003	1000	0.095±0.019	0.095±0.019
0.161±0.003	5000	0.147±0.005	0.147±0.005
0.161±0.003	10000	0.155±0.004	0.155±0.004
0.161±0.003	15000	**0.159±0.004**	**0.159±0.004**
0.161±0.003	20000	**0.159±0.004**	**0.159±0.004**
0.161±0.003	25000	**0.160±0.004**	**0.160±0.004**
0.161±0.003	30000	**0.161±0.003**	**0.161±0.003**

Table A.8: CorelFeatures-LayoutHistogram-M5

Baseline	Clusters	Weighted	Unweighted
0.285±0.011	1000	0.135±0.026	0.135±0.026
0.285±0.011	5000	0.225±0.013	0.225±0.013
0.285±0.011	10000	0.256±0.010	0.256±0.010
0.285±0.011	15000	0.267±0.011	0.267±0.011
0.285±0.011	20000	**0.277±0.011**	**0.277±0.011**
0.285±0.011	25000	**0.282±0.009**	**0.282±0.009**
0.285±0.011	30000	**0.287±0.008**	**0.287±0.008**

Table A.9: CorelFeatures-CoocTexture-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.220±0.009	1000	0.180±0.012	0.180±0.012
0.220±0.009	5000	0.208±0.004	0.208±0.004
0.220±0.009	10000	0.213±0.003	0.213±0.003
0.220±0.009	15000	**0.215±0.005**	**0.215±0.005**
0.220±0.009	20000	**0.215±0.007**	**0.215±0.007**
0.220±0.009	25000	**0.217±0.006**	**0.217±0.006**
0.220±0.009	30000	**0.220±0.009**	**0.220±0.009**

Table A.10: CorelFeatures-CoocTexture-M5

Baseline	Clusters	Weighted	Unweighted
0.301±0.003	1000	0.184±0.018	0.184±0.018
0.301±0.003	5000	0.241±0.008	0.241±0.008
0.301±0.003	10000	0.270±0.006	0.270±0.006
0.301±0.003	15000	0.280±0.008	0.280±0.008
0.301±0.003	20000	0.289±0.004	0.289±0.004
0.301±0.003	25000	0.294±0.006	0.294±0.006
0.301±0.003	30000	**0.299±0.002**	**0.299±0.002**

Table A.11: CorelFeatures-ColorMoments-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.221±0.005	1000	0.201±0.006	0.201±0.006
0.221±0.005	5000	0.215±0.006	0.215±0.006
0.221±0.005	10000	**0.219±0.006**	**0.219±0.006**
0.221±0.005	15000	**0.220±0.006**	**0.220±0.006**
0.221±0.005	20000	**0.221±0.005**	**0.221±0.005**
0.221±0.005	25000	**0.221±0.006**	**0.221±0.006**
0.221±0.005	30000	**0.221±0.005**	**0.221±0.005**

Table A.12: CorelFeatures-ColorMoments-M5

Baseline	Clusters	Weighted	Unweighted
0.333±0.004	1000	0.207±0.014	0.207±0.014
0.333±0.004	5000	0.256±0.010	0.256±0.010
0.333±0.004	10000	0.296±0.008	0.296±0.008
0.333±0.004	15000	0.309±0.008	0.309±0.008
0.333±0.004	20000	0.317±0.006	0.317±0.006
0.333±0.004	25000	0.324±0.007	0.324±0.007
0.333±0.004	30000	**0.330±0.005**	**0.330±0.005**

Table A.13: CorelFeatures-ColorHistogram-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.247±0.002	1000	0.183±0.020	0.183±0.020
0.247±0.002	5000	0.236±0.004	0.236±0.004
0.247±0.002	10000	0.242±0.003	0.242±0.003
0.247±0.002	15000	0.245±0.002	0.245±0.002
0.247±0.002	20000	**0.246±0.002**	**0.246±0.002**
0.247±0.002	25000	**0.247±0.002**	**0.247±0.002**
0.247±0.002	30000	**0.247±0.002**	**0.247±0.002**

Table A.14: CorelFeatures-ColorHistogram-M5

Baseline	Clusters	Weighted	Unweighted
0.287±0.053	1000	0.158±0.051	0.158±0.051
0.287±0.053	5000	**0.268±0.016**	**0.268±0.016**
0.287±0.053	10000	**0.261±0.076**	**0.261±0.076**
0.287±0.053	15000	**0.241±0.076**	**0.241±0.076**
0.287±0.053	20000	**0.254±0.056**	**0.254±0.056**
0.287±0.053	25000	**0.270±0.078**	**0.270±0.078**
0.287±0.053	30000	**0.264±0.053**	**0.264±0.053**

A.2 First K

First K shows more effects of removing weights however these are not as bad as expected. More work needs to be done looking into the causes of these unexpectedly good values.

Table A.15: Agrawal-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
87.780±0.299	2500	**88.985±0.370***	**88.190±0.757**
87.780±0.299	5000	87.297±0.342	**88.052±1.420**
87.780±0.299	7500	86.023±0.403	**88.644±1.678**
87.780±0.299	10000	84.966±0.506	**88.974±1.497***
87.780±0.299	12500	83.880±0.413	**89.128±1.506***
87.780±0.299	15000	83.827±0.559	**89.403±1.229***
87.780±0.299	17500	84.112±0.622	**89.404±0.929***
87.780±0.299	20000	84.683±0.628	**89.117±0.733***

Table A.16: Agrawal-Logistic

Baseline	Clusters	Weighted	Unweighted
67.298±0.002	2500	66.814±0.505	49.593±0.705
67.298±0.002	5000	**67.298±0.002**	51.941±0.517
67.298±0.002	7500	**67.298±0.002**	51.603±0.559
67.298±0.002	10000	**67.298±0.002**	49.883±0.492
67.298±0.002	12500	**67.298±0.002**	48.914±0.260
67.298±0.002	15000	**67.298±0.002**	52.238±0.484
67.298±0.002	17500	**67.298±0.002**	59.351±0.827
67.298±0.002	20000	**67.298±0.002**	66.110±0.431

Table A.17: Waveform21-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
80.430±0.228	2500	**82.276±0.220***	**82.794±0.379***
80.430±0.228	5000	**82.396±0.238***	**82.225±0.324***
80.430±0.228	7500	**82.042±0.244***	**81.820±0.302***
80.430±0.228	10000	**81.640±0.200***	**81.494±0.251***
80.430±0.228	12500	**81.246±0.173***	**81.188±0.259***
80.430±0.228	15000	**80.901±0.219***	**80.938±0.263***
80.430±0.228	17500	**80.644±0.223***	**80.666±0.245***
80.430±0.228	20000	**80.438±0.232**	**80.408±0.232**

Table A.18: Waveform21-Logistic

Baseline	Clusters	Weighted	Unweighted
86.607±0.161	2500	85.755±0.288	85.973±0.165
86.607±0.161	5000	86.141±0.151	86.413±0.114
86.607±0.161	7500	86.331±0.181	86.476±0.099
86.607±0.161	10000	86.467±0.156	**86.528±0.129**
86.607±0.161	12500	**86.530±0.160**	**86.567±0.146**
86.607±0.161	15000	**86.535±0.165**	**86.559±0.167**
86.607±0.161	17500	**86.574±0.169**	**86.569±0.160**
86.607±0.161	20000	**86.602±0.162**	**86.606±0.168**

Table A.19: Waveform40-NaiveBayes

Baseline	Clusters	Weighted	Unweighted
80.337±0.130	2500	**81.249±0.267***	**82.615±0.222***
80.337±0.130	5000	**82.388±0.149***	**82.043±0.138***
80.337±0.130	7500	**82.214±0.115***	**81.653±0.196***
80.337±0.130	10000	**81.714±0.118***	**81.345±0.159***
80.337±0.130	12500	**81.282±0.111***	**81.094±0.130***
80.337±0.130	15000	**80.947±0.107***	**80.838±0.131***
80.337±0.130	17500	**80.629±0.124***	**80.578±0.131***
80.337±0.130	20000	**80.341±0.128**	**80.328±0.133**

Table A.20: Waveform40-Logistic

Baseline	Clusters	Weighted	Unweighted
86.644±0.130	2500	85.317±0.474	85.484±0.465
86.644±0.130	5000	86.159±0.200	86.191±0.156
86.644±0.130	7500	86.340±0.195	86.425±0.141
86.644±0.130	10000	86.454±0.190	86.456±0.149
86.644±0.130	12500	**86.527±0.174**	**86.545±0.166**
86.644±0.130	15000	**86.603±0.168**	**86.584±0.122**
86.644±0.130	17500	**86.633±0.130**	**86.630±0.145**
86.644±0.130	20000	**86.631±0.131**	**86.635±0.129**

Table A.21: CorelFeatures-LayoutHistogram-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.161±0.003	1000	0.071±0.023	0.030±0.020
0.161±0.003	5000	0.106±0.019	0.051±0.022
0.161±0.003	10000	0.123±0.014	0.058±0.018
0.161±0.003	15000	0.132±0.010	0.087±0.011
0.161±0.003	20000	0.145±0.007	0.118±0.013
0.161±0.003	25000	0.155±0.004	0.149±0.005
0.161±0.003	30000	**0.160±0.003**	**0.159±0.003**

Table A.22: CorelFeatures-LayoutHistogram-M5

Baseline	Clusters	Weighted	Unweighted
0.285±0.011	1000	0.005±0.032	-0.002
0.285±0.011	5000	0.018±0.011	0.027±0.008
0.285±0.011	10000	0.010±0.013	0.019
0.285±0.011	15000	0.029±0.012	-0.005
0.285±0.011	20000	0.072±0.013	0.020±0.011
0.285±0.011	25000	0.166±0.042	0.010±0.063
0.285±0.011	30000	0.232±0.082	0.015±0.182

Table A.23: CorelFeatures-CoocTexture-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.220±0.009	1000	0.147±0.025	0.102±0.031
0.220±0.009	5000	0.173±0.013	0.117±0.012
0.220±0.009	10000	0.177±0.014	0.119±0.013
0.220±0.009	15000	0.188±0.012	0.132±0.013
0.220±0.009	20000	0.196±0.009	0.148±0.015
0.220±0.009	25000	**0.212±0.012**	0.190±0.012
0.220±0.009	30000	**0.219±0.010**	**0.217±0.010**

Table A.24: CorelFeatures-CoocTexture-M5

Baseline	Clusters	Weighted	Unweighted
0.301±0.003	1000	0.110±0.012	0.117±0.014
0.301±0.003	5000	0.123±0.012	0.127±0.011
0.301±0.003	10000	0.138±0.008	0.141±0.010
0.301±0.003	15000	0.152±0.007	0.158±0.007
0.301±0.003	20000	0.173±0.008	0.183±0.006
0.301±0.003	25000	0.218±0.006	0.234±0.006
0.301±0.003	30000	0.275±0.007	0.284±0.006

Table A.25: CorelFeatures-ColorMoments-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.221±0.005	1000	0.167±0.019	0.088±0.035
0.221±0.005	5000	0.192±0.010	0.126±0.016
0.221±0.005	10000	0.197±0.009	0.136±0.009
0.221±0.005	15000	0.202±0.006	0.153±0.007
0.221±0.005	20000	0.209±0.005	0.180±0.007
0.221±0.005	25000	0.216±0.005	0.207±0.004
0.221±0.005	30000	**0.220±0.005**	**0.219±0.005**

Table A.26: CorelFeatures-ColorMoments-M5

Baseline	Clusters	Weighted	Unweighted
0.333±0.004	1000	-0.012±0.021	-0.003
0.333±0.004	5000	0.005±0.013	0.022±0.017
0.333±0.004	10000	0.015±0.018	0.013±0.025
0.333±0.004	15000	0.055±0.014	0.015±0.073
0.333±0.004	20000	0.106±0.015	0.015±0.135
0.333±0.004	25000	0.190±0.007	0.013±0.227
0.333±0.004	30000	0.285±0.006	0.006±0.303

Table A.27: CorelFeatures-ColorHistogram-LinearRegression

Baseline	Clusters	Weighted	Unweighted
0.247±0.002	1000	0.153±0.025	0.089±0.027
0.247±0.002	5000	0.212±0.010	0.153±0.015
0.247±0.002	10000	0.220±0.009	0.173±0.013
0.247±0.002	15000	0.230±0.004	0.196±0.007
0.247±0.002	20000	0.237±0.003	0.216±0.006
0.247±0.002	25000	0.242±0.002	0.234±0.003
0.247±0.002	30000	**0.246±0.002**	0.244±0.002

Appendix B
Noise Reduction

The aim of these experiments was to determine whether it might be possible to use the Cluster Classifier as a method of noise detection and removal. This is accomplished by removing clusters that have low support. Thresholds of 2.5 and 5 data points were tried. Since Bisecting K-Means creates even sized clusters it is not suitable for use with this method. Only large datasets were used since smaller datasets do not have as much difference in the support for each meta-data point.

B.1 Farthest First

All the following experiments are run with the Naive Bayes classifier. Logistic Regression failed to produce results on these datasets due to long runtimes.

Table B.1: Agrawal

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
87.780±0.299	2500	**92.249±0.445***	67.298±0.002	67.298±0.002
87.780±0.299	5000	**92.809±0.544***	67.298±0.002	67.298±0.002
87.780±0.299	7500	**93.019±0.495***	67.298±0.002	67.298±0.002
87.780±0.299	10000	**92.729±0.300***	67.298±0.002	67.298±0.002
87.780±0.299	12500	**92.278±0.238***	67.298±0.002	67.298±0.002
87.780±0.299	15000	**92.421±0.223***	67.298±0.002	67.298±0.002
87.780±0.299	17500	**92.400±0.179***	67.298±0.002	67.298±0.002
87.780±0.299	20000	**91.767±0.188***	67.298±0.002	67.298±0.002

Farthest First was also run on the Corel Datasets. However both Linear Regression and M5 did not produce consistent results. More investigation is needed to determine the cause of this.

Table B.2: Waveform21

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
80.430±0.228	2500	78.896±0.253	33.745±0.001	33.745±0.001
80.430±0.228	5000	79.182±0.241	33.745±0.001	33.745±0.001
80.430±0.228	7500	79.446±0.256	33.745±0.001	33.745±0.001
80.430±0.228	10000	79.667±0.241	33.745±0.001	33.745±0.001
80.430±0.228	12500	79.853±0.248	33.745±0.001	33.745±0.001
80.430±0.228	15000	80.051±0.243	33.745±0.001	33.745±0.001
80.430±0.228	17500	80.228±0.235	33.745±0.001	33.745±0.001
80.430±0.228	20000	**80.416±0.229**	33.745±0.001	33.745±0.001

Table B.3: Waveform40

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
86.607±0.161	2500	79.019±0.238	33.695±0.001	33.695±0.001
86.607±0.161	5000	79.308±0.192	33.695±0.001	33.695±0.001
86.607±0.161	7500	79.534±0.187	33.695±0.001	33.695±0.001
86.607±0.161	10000	79.709±0.169	33.695±0.001	33.695±0.001
86.607±0.161	12500	79.847±0.153	33.695±0.001	33.695±0.001
86.607±0.161	15000	80.016±0.172	33.695±0.001	33.695±0.001
86.607±0.161	17500	80.152±0.149	33.695±0.001	33.695±0.001
86.607±0.161	20000	80.319±0.129	33.695±0.001	33.695±0.001

B.2 First K

First K unlike Farthest First managed to complete on all the datasets and with all classifiers. For the Waveform datasets under Logistic Regression there were not enough data points to produce a result with twenty thousand clusters, NA was entered in place of these missing results.

Table B.4: Agrawal-NaiveBayes

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
87.780±0.299	2500	**88.985±0.370***	**87.512±0.600**	83.499±1.333
87.780±0.299	5000	87.297±0.342	82.201±0.472	75.150±1.075
87.780±0.299	7500	86.023±0.403	78.050±0.920	76.256±2.283
87.780±0.299	10000	84.966±0.506	74.422±1.453	67.597±0.382
87.780±0.299	12500	83.880±0.413	72.731±2.465	67.298±0.002
87.780±0.299	15000	83.827±0.559	67.298±0.002	67.298±0.002
87.780±0.299	17500	84.112±0.622	67.298±0.002	67.298±0.002
87.780±0.299	20000	84.683±0.628	67.298±0.002	67.298±0.002

Table B.5: Agrawal-Logistic

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
67.298±0.002	2500	66.814±0.505	63.933±1.738	62.044±1.077
67.298±0.002	5000	**67.298±0.002**	65.259±0.559	65.384±0.528
67.298±0.002	7500	**67.298±0.002**	66.517±0.462	66.822±0.465
67.298±0.002	10000	**67.298±0.002**	67.190±0.074	64.837±1.940
67.298±0.002	12500	**67.298±0.002**	67.279±0.027	**67.298±0.002**
67.298±0.002	15000	**67.298±0.002**	**67.298±0.002**	**67.298±0.002**
67.298±0.002	17500	**67.298±0.002**	**67.298±0.002**	**67.298±0.002**
67.298±0.002	20000	**67.298±0.002**	**67.298±0.002**	**67.298±0.002**

Table B.6: Waveform21-NaiveBayes

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
80.430±0.228	2500	**82.276±0.220***	**81.316±0.255***	**80.269±0.441**
80.430±0.228	5000	**82.396±0.238***	**82.164±0.236***	**81.397±0.250***
80.430±0.228	7500	**82.042±0.244***	**82.129±0.265***	**81.478±0.267***
80.430±0.228	10000	**81.640±0.200***	**81.924±0.249***	**81.344±0.240***
80.430±0.228	12500	**81.246±0.173***	**81.684±0.243***	**80.807±0.282***
80.430±0.228	15000	**80.901±0.219***	**81.378±0.189***	79.312±0.828
80.430±0.228	17500	**80.644±0.223***	**80.986±0.533***	44.561±8.195
80.430±0.228	20000	**80.438±0.232**	33.745±0.001	33.745±0.001

Table B.7: Waveform21-Logistic

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
86.607±0.161	2500	85.755±0.288	83.943±0.941	79.600±2.864
86.607±0.161	5000	86.141±0.151	83.927±0.826	78.838±3.782
86.607±0.161	7500	86.331±0.181	84.007±0.899	76.970±6.324
86.607±0.161	10000	86.467±0.156	83.890±1.309	75.698±5.114
86.607±0.161	12500	**86.530±0.160**	83.281±1.245	66.358±6.988
86.607±0.161	15000	**86.535±0.165**	81.332±1.605	56.475±4.260
86.607±0.161	17500	**86.574±0.169**	76.669±4.646	54.370±10.617
86.607±0.161	20000	**86.602±0.162**	NA	NA

Table B.8: Waveform40-NaiveBayes

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
80.337±0.130	2500	**81.249±0.267***	79.462±0.517	77.727±0.907
80.337±0.130	5000	**82.388±0.149***	**81.437±0.201***	79.905±0.269
80.337±0.130	7500	**82.214±0.115***	**81.949±0.165***	**80.485±0.300**
80.337±0.130	10000	**81.714±0.118***	**81.986±0.193***	**80.418±0.314**
80.337±0.130	12500	**81.282±0.111***	**81.840±0.214***	79.711±0.347
80.337±0.130	15000	**80.947±0.107***	**81.557±0.163***	75.218±3.007
80.337±0.130	17500	**80.629±0.124***	**81.068±0.372***	40.460±8.833
80.337±0.130	20000	**80.341±0.128**	33.698±0.006	33.695±0.001

Table B.9: Waveform40-Logistic

Baseline	Clusters	Threshold-0	Threshold-2.5	Threshold-5
86.644±0.130	2500	85.317±0.474	81.188±1.317	77.204±1.719
86.644±0.130	5000	86.159±0.200	84.636±0.584	78.421±1.432
86.644±0.130	7500	86.340±0.195	84.826±0.583	76.770±2.556
86.644±0.130	10000	86.454±0.190	85.080±0.766	73.936±4.313
86.644±0.130	12500	**86.527±0.174**	84.725±0.957	66.459±6.475
86.644±0.130	15000	**86.603±0.168**	83.138±1.191	44.642±3.203
86.644±0.130	17500	**86.633±0.130**	74.832±2.201	48.560±7.704
86.644±0.130	20000	**86.631±0.131**	NA	NA

Table B.10: CorelFeatures-LayoutHistogram-LinearRegression

Baseline	Clusters	Threshold-0	Threshold-5
0.161±0.003	1000	0.071±0.023	0.018±0.035
0.161±0.003	5000	0.106±0.019	-0.010±0.026
0.161±0.003	10000	0.123±0.014	0.012±0.026
0.161±0.003	15000	0.132±0.010	0.000±0.031
0.161±0.003	20000	0.145±0.007	-0.012±0.042
0.161±0.003	25000	0.155±0.004	0.000±0.032
0.161±0.003	30000	**0.160±0.003**	-0.018±0.039

Table B.11: CorelFeatures-LayoutHistogram-M5

Baseline	Clusters	Threshold-0	Threshold-5
0.285±0.011	1000	0.005±0.032	0.006±0.030
0.285±0.011	5000	0.018±0.011	-0.012±0.037
0.285±0.011	10000	0.010±0.013	0.002±0.032
0.285±0.011	15000	0.029±0.012	-0.010±0.023
0.285±0.011	20000	0.072±0.013	-0.002±0.052
0.285±0.011	25000	0.166±0.042	0.011±0.045
0.285±0.011	30000	0.232±0.082	-0.006±0.013

Table B.12: CorelFeatures-CoocTexture-LinearRegression

Baseline	Clusters	Threshold-0	Threshold-5
0.220±0.009	1000	0.147±0.025	0.037±0.071
0.220±0.009	5000	0.173±0.013	0.076±0.060
0.220±0.009	10000	0.177±0.014	0.085±0.065
0.220±0.009	15000	0.188±0.012	0.080±0.047
0.220±0.009	20000	0.196±0.009	0.073±0.038
0.220±0.009	25000	**0.212±0.012**	0.024±0.082
0.220±0.009	30000	**0.219±0.010**	-0.006±0.037

Table B.13: CorelFeatures-CoocTexture-M5

Baseline	Clusters	Threshold-0	Threshold-5
0.301±0.003	1000	0.110±0.012	-0.004±0.055
0.301±0.003	5000	0.123±0.012	0.011±0.041
0.301±0.003	10000	0.138±0.008	0.056±0.060
0.301±0.003	15000	0.152±0.007	0.047±0.043
0.301±0.003	20000	0.173±0.008	0.074±0.040
0.301±0.003	25000	0.218±0.006	0.029±0.093
0.301±0.003	30000	0.275±0.007	-0.001±0.029

Table B.14: CorelFeatures-ColorMoments-LinearRegression

Baseline	Clusters	Threshold-0	Threshold-5
0.221±0.005	1000	0.167±0.019	0.029±0.069
0.221±0.005	5000	0.192±0.010	0.059±0.058
0.221±0.005	10000	0.197±0.009	0.037±0.061
0.221±0.005	15000	0.202±0.006	0.042±0.067
0.221±0.005	20000	0.209±0.005	0.063±0.049
0.221±0.005	25000	0.216±0.005	0.012±0.066
0.221±0.005	30000	**0.220±0.005**	-0.007±0.033

Table B.15: CorelFeatures-ColorMoments-M5

Baseline	Clusters	Threshold-0	Threshold-5
0.333±0.004	1000	-0.012±0.021	-0.005±0.066
0.333±0.004	5000	0.005±0.013	-0.046±0.059
0.333±0.004	10000	0.015±0.018	-0.051±0.035
0.333±0.004	15000	0.055±0.014	-0.009±0.072
0.333±0.004	20000	0.106±0.015	0.051±0.051
0.333±0.004	25000	0.190±0.007	0.018±0.078
0.333±0.004	30000	0.285±0.006	-0.020±0.083

Table B.16: CorelFeatures-ColorHistogram-LinearRegression

Baseline	Clusters	Threshold-0	Threshold-5
0.247±0.002	1000	0.153±0.025	0.041±0.055
0.247±0.002	5000	0.212±0.010	0.038±0.051
0.247±0.002	10000	0.220±0.009	0.021±0.049
0.247±0.002	15000	0.230±0.004	0.064±0.038
0.247±0.002	20000	0.237±0.003	0.030±0.046
0.247±0.002	25000	0.242±0.002	0.004±0.055
0.247±0.002	30000	**0.246±0.002**	-0.032±0.061

Bibliography

Ankerst, M., Breunig, M. M., Kriegel, H.-P. & Sander, J. (1999). Optics: ordering points to identify the clustering structure. *SIGMOD Rec.*, *28(2)*, 49–60.

Barandela, R., Valdovinos, R. & Sánchez, J. (2003). New applications of ensembles of classifiers. *Pattern Analysis and Applications*, *6(3)*, 245–256.

Blake, C., Keogh, E. & Merz, C. (1998). UCI machine learning repository.

Brighton, H. & Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, *6(2)*, 153–172.

Cano, J. R., Herrera, F. & Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *Evolutionary Computation, IEEE Transactions on*, *7(6)*, 561–575.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the *em* algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39(1)*, 1–38.

Domingo, C., Gavaldà, R. & Watanabe, O. (2002). Adaptive sampling methods for scaling up knowledge discovery algorithms. *Data Mining and Knowledge Discovery*, *6(2)*, 131–152.

DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C. & Pregibon, D. (1999). Squashing flat files flatter. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 6–15). New York, NY, USA: ACM Press.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York, NY, USA: John Wiley & Sons, Inc.

Hochbaum, Dorit S. & Shmoys, David B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, *10(2)*, 180–184.

Inaba, M., Katoh, N. & Imai, H. (1994). Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *SCG '94: Proceedings of the tenth annual symposium on Computational geometry* (pp. 332–339). New York, NY, USA: ACM Press.

Li, B., Chi, M., Fan, J. & Xue, X. (2007). Support cluster machine. In *Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR*.

LI, X.-B. (2002). Data reduction via adaptive sampling. *Comunications in Information and Systems*, *2(1)*, 53–68.

Liu, H. & Motoda, H. (2002). On issues of instance selection. *Data Mining and Knowledge Discovery*, *6(2)*, 115–130.

Madigan, D., Raghavan, N., Dumouchel, W., Nason, M., Posse, C. & Ridgeway, G. (2002). Likelihood-based data squashing: A modeling approach to instance construction. *Data Mining and Knowledge Discovery*, *6(2)*, 173–190.

Nadeau, C. & Bengio, Y. (1999). Inference for the generalization error. In Solla, S. A., Leen, T. K. & Müller, K.-R. (Eds.), *NIPS* (pp. 307–313). The MIT Press.

Reinartz, T. (2002). A unifying view on instance selection. *Data Mining and Knowledge Discovery*, *6(2)*, 191–210.

Steinbach, M., Karypis, G. & Kumar, V. (2000). A comparison of document clustering techniques.

Vassilvitskii, S. & Arthur, D. (2006). How slow is the k-means method?". In *Symposium on Computational Geometry (SOCG)*.

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (2 Ed.). San Francisco, CA: Morgan Kaufmann.

Zhang, T., Ramakrishnan, R. & Livny, M. (1996). Birch: an efficient data clustering method for very large databases. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data* (pp. 103–114). New York, NY, USA: ACM Press.