CHAPTER 01

INTRODUCTION

Data mining is the process of evaluating patterns within a large data set. Data mining techniques include creating regression models of the data, recognising anomalies, associations and clusters within the data, and summarising the entire data. It transforms raw data into a comprehensible form.

Data mining algorithms have a variety of applications. Data mining algorithms are used for detecting similarity in DNA sequences. Patient demographic data is data mined to yield insight into ailments and genetic mutations with respect to geography and inheritance.

Pattern assessment in interpreting market research consumer, can aid businesses identify current and predict future market demand and trends and accordingly develop marketing strategies.

Data science forms the basis of machine learning, which is crucial in automation and artificial intelligence. Classification and clustering algorithms are important to give machine decision making and learning capability.

Optimisation of data mining techniques would effectively reduce the computational time and enable us to make larger computations which require hardware of higher specification more feasible, thus making it more accessible to real time solutions.

In this project we focus on analysis of clustering and classification analysis namely, k-means, agglomerative hierarchical clustering, naive Baye's, k-nearest neighbours and decision tree classification.

This project has a different approach in the analysis, and it tries to compare both efficiency between algorithms and also the efficiency between classifiers of same algorithm but which operate on a segment of the dataset. Many algorithms used here, depend on the dimensionality of the data. If we can reduce the dimensionality of the data, we would reduce the effective functioning time of the data by a large factor. Hence we have conducted experiments on the dataset to show that this is a possible and a feasible optimisation solution which can be applied to a dataset to improve performance.

CHAPTER 02

BACKGROUND

To understand the functioning of each algorithm and its strengths and weaknesses, a literary survey was performed on the current level of research already made on the analysis of datasets. To do this, research papers were referenced which introduced and analysed of each algorithm. The conventional algorithms along with their optimised versions were studied. The optimised versions were studied to understand the scope of improvement in the conventional algorithms.
The following papers were used for algorithmic support and understanding the use and application of data mining algorithms.

'A Survey Of Decision Tree Methodology' by S. Rasoul Safavian and David Landgrebe : was studied to understand the concept of decision trees.

'Naive Bayes classification algorithm based on small sample set' by Yuguang Huang and Lei Li : was studied to understand the calculations of naive Baye's methods.

'An efficient k-means clustering algorithm: analysis and implementation' by T. Kanungo , D.M. Mount , N.S. Netanyahu C.D. Piatko , R. Silverman and A.Y. Wu : was studied to understand efficient optimisations which can be made to k-means algorithm, and also to understand the applications of a kd-tree data structure to implement this algorithm.

'Evaluation of hierarchical clustering algorithms for document datasets' by
Ying Zhao and George Karypis: used to understand types of hierarchical algorithm, and its extensions, and the discrepancy in performance. It was used to study how conventional algorithm can be optimised.
'A k-nearest neighbor based algorithm for multi-label classification' by Min-Ling Zhang and Zhi-Hua Zhou: studied the convention k-nn classification model and studied its optimisations using lazy learning approach.

CHAPTER 03

PROBLEM DEFINITION AND SCOPE

The primary objective of this project is the analysis of various classification and clustering algorithms used to mine data. Parameters of performance comparison is the time computed while testing the algorithm, and the accuracy of a particular algorithm. Based on the observation results, further optimisation possibilities have been discussed in the project.

The report analyses classification and clustering algorithms. The classification algorithms analysed are naive Baye's, k-nearest neighbours and decision tree classification. Clustering algorithms analysed are k-means clustering and agglomerative hierarchical clustering. These algorithms can be classified into either supervised or unsupervised learning.

Supervised learning is a branch of data science where the classifier or the algorithms is trained initially, by data similar to that on which it is tested. Classification is a branch of supervised learning. Performance of classification algorithms depends on the amount of training data fed.

Classification algorithms make predictions about data based on its attributes values.

Unsupervised learning is when algorithms are directly run on the data and do not require training. These algorithms directly operate on the data, and produce expected results. Clustering algorithms fall under the unsupervised category, and are used to detect clouts and gropings within the data.

All classifiers have been tested and trained on the Iris flower dataset, where 50% of the dataset has been used for training, whereas the other 50% is used for testing the classifier. The results from the tested data is then converted into a confusion matrix, and the accuracy of classification is calculated from the confusion matrix. A ratio between accuracy and time is computed then to rate the best classifier. The most optimal classifier has then been chosen amongst all classifiers.

In clustering, the algorithms have directly run on the entire Iris flower dataset. The silhouette value and the purity of cluster model has been calculated. Then a performance metric with respect to silhouette value, purity and time has been computed and the most optimal cluster model has been chosen to predict the most optimal model.

CHAPTER 04

PROJECT PLAN

The Iris flower dataset had 150 tuples, with 4 attributes each. Tuples have individual value for these 4 attributes, namely ‘Sepal Length’, ‘Sepal Width’, ‘Petal Length’ and ‘Petal Width’. Each of these tuples are divided into 3 respective classes, ‘Setosa’, ‘Versicolor’ and ‘Virginica’, depending on the value of their attributes.

For classification models, the data is randomly divided into 2 equal halves, of which one is used for training the classifier whereas the other is used for testing. In the testing set used for classification, the number of tuples for class ‘Setosa’ is 23, for ‘Versicolor’ is 25 and for ‘Virginica’ is 27.

For clustering models, the entire dataset has been used to test the algorithm, the set contains 50 tuples of each class which are then clustered by the model without prior knowledge about the data.

For each classification and clustering algorithm 11 different classifiers have been designed. The difference between each classifier, is the training methodology used and the testing data which is fed to them. For each classifier the number of attributes and the combination of attributed used to test and train the classifier is different. The classifier then predicts the test dataset on the combination of attributes for which training has been provided.

Clustering models similar to the classification models, cluster the dataset only on specific attributes. They produce results based on the of specific combination of attributes.

Models consider each combination of attribute as an entire dataset and then accordingly cluster and classify on the test dataset.

The combination of attributes used on classification and clustering models are ‘Sepal Length, Sepal Width, Petal Length And Petal Width’, ‘Sepal Length, Sepal Width And Petal Length’, ‘Sepal Width, Petal Length And Petal Width’, ‘Sepal Length, Sepal Width And Petal Width’, ‘Sepal Length, Petal Length And Petal Width’, ‘Sepal Width And Sepal Length’, ‘Petal Length And Sepal Length’, ‘Petal Width And Sepal Length’, ‘Petal Width And Sepal Length’, ‘Petal Width And Sepal Width’ and ‘Petal Width And Petal Length’.

The testing time for each classifier to classify all data points was recorded. The test time has been averaged over 10 iterations to get consistent results. From these classification results a confusion matrix has been calculated, which shows how many points have been classified in which class. It also shows the number of points which have been miscalculated and shows both the actual and the predicted class of the data point. This matrix has been used to calculate the average accuracy of each classifier. This has been tabulated in the results. A performance metric has been designed to normalise and then calculate the ratio between accuracy and time consumed to find out which is the most optimal classifier for an algorithm. An average performance metric and an average accuracy and testing time has also be computed so that we compare an algorithmic performance.

The clustering model was fed the entire dataset and it separated the data into clusters. In both k-means and agglomerative hierarchical clustering the number of clusters were preset to 3. In k-means the algorithm repeatedly iterates 4 times to pick the minimum starting cluster points, as this affects the performance of the cluster. The testing time for each classifier to classify all data points was recorded. The test time has been averaged over 10 iterations to get consistent data. To evaluate the accuracy of the clusters the purity within each cluster has been calculated. The silhouette value has been additionally calculated to show the density of each cluster. A performance metric has been calculated considering all 3 performance parameters. Based on this the cluster models have been compared and the optimal cluster model for this dataset has been shown.

In calculation of time, the disk access cost, processing time and hardware costs have not been factored out and is included within the results. Although, due to the averaged iteration results the timing results obtained are stable and predictions can be made on them. The stability of each classifier was verified as with a given dataset the classifier produced the same set of results each time.

CHAPTER 05

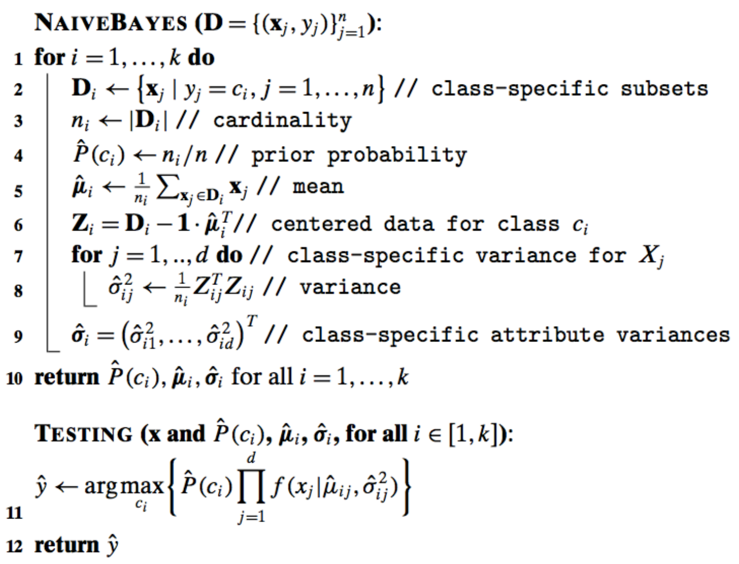
DETAILED DESIGN

Naive Bayes Algorithm:

Computational Complexity: $o(nd)$

Where ‘d’ is the number of dimensions in the data and ’n’ is the number of tuples in the data.

Figure 5.1: Naive baye’s Algorithm



```
NAIVEBAYES ($\mathbf{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n}$):
1  for $i = 1, \ldots, k$ do
2      $\mathbf{D}_i \leftarrow \{\mathbf{x}_j \mid y_j = c_i, j = 1, \ldots, n\}$ // class-specific subsets
3      $n_i \leftarrow |\mathbf{D}_i|$ // cardinality
4      $\hat{P}(c_i) \leftarrow n_i/n$ // prior probability
5      $\hat{\boldsymbol{\mu}}_i \leftarrow \frac{1}{n_i}\sum_{\mathbf{x}_j \in \mathbf{D}_i} \mathbf{x}_j$ // mean
6      $\mathbf{Z}_i = \mathbf{D}_i - \mathbf{1} \cdot \hat{\boldsymbol{\mu}}_i^T$ // centered data for class $c_i$
7      for $j = 1, .., d$ do // class-specific variance for $X_j$
8          $\hat{\sigma}_{ij}^2 \leftarrow \frac{1}{n_i} Z_{ij}^T Z_{ij}$ // variance
9      $\hat{\boldsymbol{\sigma}}_i = (\hat{\sigma}_{i1}^2, \ldots, \hat{\sigma}_{id}^2)^T$ // class-specific attribute variances
10 return $\hat{P}(c_i), \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\sigma}}_i$ for all $i = 1, \ldots, k$

TESTING (x and $\hat{P}(c_i)$, $\hat{\boldsymbol{\mu}}_i$, $\hat{\boldsymbol{\sigma}}_i$, for all $i \in [1, k]$):
11 $\hat{y} \leftarrow \arg\max_{c_i} \left\{ \hat{P}(c_i) \prod_{j=1}^{d} f(x_j | \hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) \right\}$
12 return $\hat{y}$
```

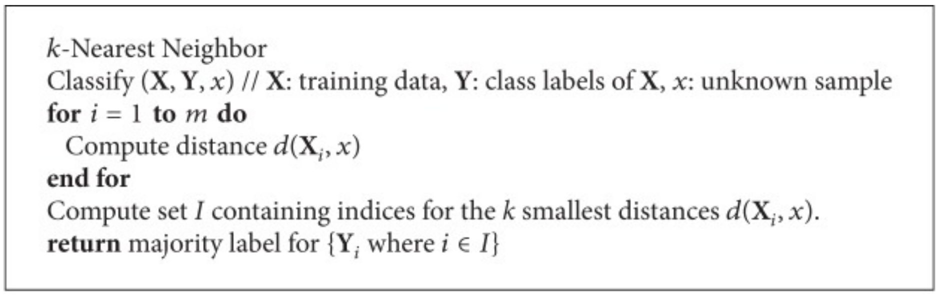
*Note: This algorithm has been sourced from‘DATA MINING AND ANALYSIS.’[1]

K-Nearest Neighbours Algorithm:

Computational Complexity: $o(ndk)$

Where ‘d’ is the number of dimensions in the data, ‘k’ is the number of neighbours and ’n’ is the number of tuples in the data.

Figure 5.2: K-Nearest Neighbours Algorithm



```
$k$-Nearest Neighbor
Classify $(\mathbf{X}, \mathbf{Y}, x)$ // $\mathbf{X}$: training data, $\mathbf{Y}$: class labels of $\mathbf{X}$, $x$: unknown sample
for $i = 1$ to $m$ do
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set $I$ containing indices for the $k$ smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$
```

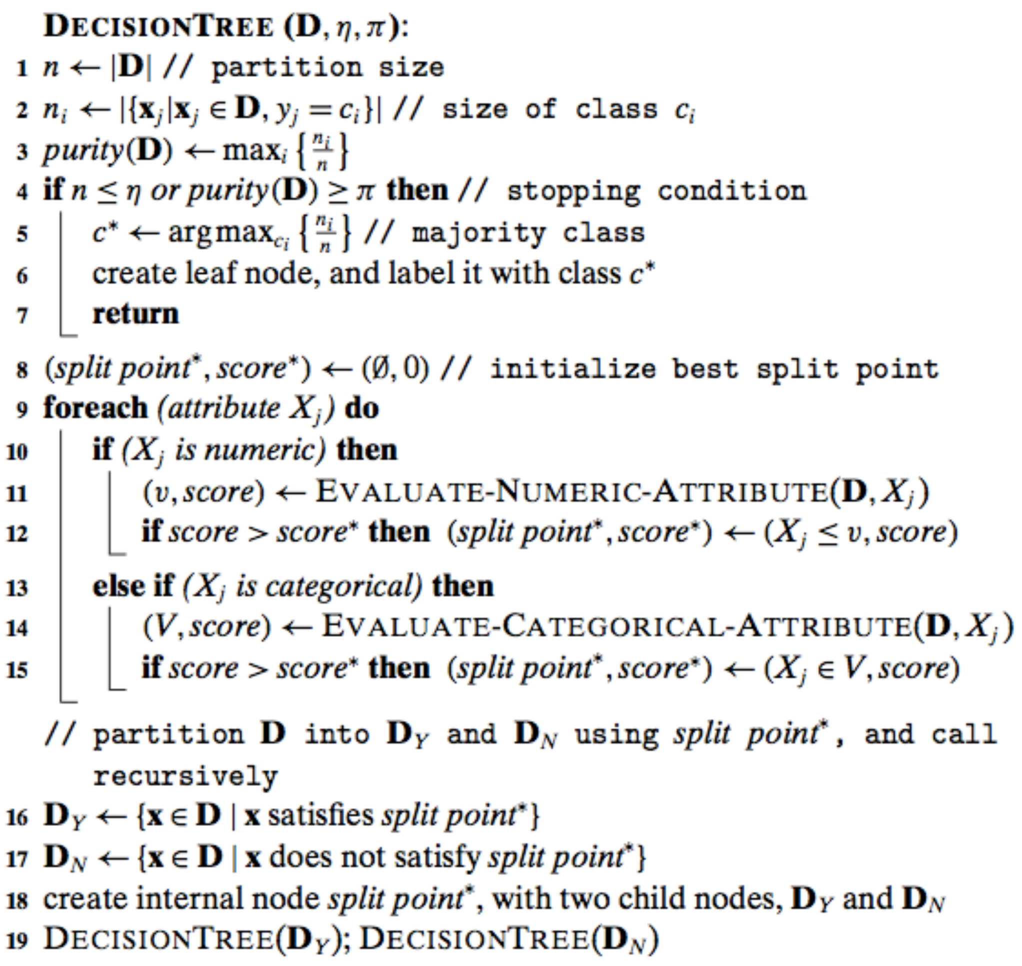
*Note: This algorithm has been sourced from ‘A Machine Learning Approach for Specification of Spinal Cord Injuries Using Fractional Anisotropy Values Obtained from Diffusion Tensor Images.’[2]

Decision Tree Algorithm:

Computational complexity: $o(dn^2 logn)$

Where d = number of dimensions, and n is the number of tuples.

Figure 5.3: Decision Tree Algorithm



```
DECISIONTREE ($\mathbf{D}, \eta, \pi$):
$n \leftarrow |\mathbf{D}|$ // partition size
$n_i \leftarrow |\{\mathbf{x}_j | \mathbf{x}_j \in \mathbf{D}, y_j = c_i\}|$ // size of class $c_i$
$purity(\mathbf{D}) \leftarrow \max_i \{\frac{n_i}{n}\}$
if $n \leq \eta$ or $purity(\mathbf{D}) \geq \pi$ then // stopping condition
    $c^* \leftarrow \arg\max_{c_i} \{\frac{n_i}{n}\}$ // majority class
    create leaf node, and label it with class $c^*$
    return
$(split\ point^*, score^*) \leftarrow (\emptyset, 0)$ // initialize best split point
foreach (attribute $X_j$) do
    if ($X_j$ is numeric) then
        $(v, score) \leftarrow$ EVALUATE-NUMERIC-ATTRIBUTE$(\mathbf{D}, X_j)$
        if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \leq v, score)$
    else if ($X_j$ is categorical) then
        $(V, score) \leftarrow$ EVALUATE-CATEGORICAL-ATTRIBUTE$(\mathbf{D}, X_j)$
        if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \in V, score)$
// partition $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$ using $split\ point^*$, and call
   recursively
$\mathbf{D}_Y \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ satisfies $split\ point^*\}$
$\mathbf{D}_N \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ does not satisfy $split\ point^*\}$
create internal node $split\ point^*$, with two child nodes, $\mathbf{D}_Y$ and $\mathbf{D}_N$
DECISIONTREE$(\mathbf{D}_Y)$; DECISIONTREE$(\mathbf{D}_N)$
```

*Note: This algorithm has been sourced from'DATA MINING AND ANALYSIS.'[1]

Figure 5.4: Gain of Numeric Attributes

```
EVALUATE-NUMERIC-ATTRIBUTE ($\mathbf{D}, X$):
sort $\mathbf{D}$ on attribute $X$, so that $x_j \leq x_{j+1}, \forall j = 1, \dots, n-1$
$\mathcal{M} \leftarrow \emptyset$ // set of midpoints
for $i = 1, \dots, k$ do $n_i \leftarrow 0$
for $j = 1, \dots, n-1$ do
    if $y_j = c_i$ then $n_i \leftarrow n_i + 1$ // running count for class $c_i$
    if $x_{j+1} \neq x_j$ then
        $v \leftarrow \frac{x_{j+1} + x_j}{2}$; $\mathcal{M} \leftarrow \mathcal{M} \cup \{v\}$ // midpoints
        for $i = 1, \dots, k$ do
            $N_{vi} \leftarrow n_i$ // Number of points such that $x_j \leq v$ and $y_j = c_i$
if $y_n = c_i$ then $n_i \leftarrow n_i + 1$
// evaluate split points of the form $X \leq v$
$v^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
forall $v \in \mathcal{M}$ do
    for $i = 1, \dots, k$ do
        $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}$
        $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{n_i - N_{vi}}{\sum_{j=1}^{k} n_j - N_{vj}}$
    $score(X \leq v) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
    if $score(X \leq v) > score^*$ then
        $v^* \leftarrow v$; $score^* \leftarrow score(X \leq v)$
return $(v^*, score^*)$
```

*Note: This algorithm has been sourced from 'DATA MINING AND ANALYSIS.'[1]

Figure 5.5: Gain of Categorical Attributes

```
EVALUATE-CATEGORICAL-ATTRIBUTE ($\mathbf{D}, X, l$):
for $i = 1, \dots, k$ do
    $n_i \leftarrow 0$
    forall $v \in dom(X)$ do $n_{vi} \leftarrow 0$
for $j = 1, \dots, n$ do
    if $x_j = v$ and $y_j = c_i$ then $n_{vi} \leftarrow n_{vi} + 1$ // frequency statistics
// evaluate split points of the form $X \in V$
$V^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
forall $V \subset dom(X)$, such that $1 \leq |V| \leq l$ do
    for $i = 1, \dots, k$ do
        $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}}$
        $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$
    $score(X \in V) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
    if $score(X \in V) > score^*$ then
        $V^* \leftarrow V$; $score^* \leftarrow score(X \in V)$
return $(V^*, score^*)$
```

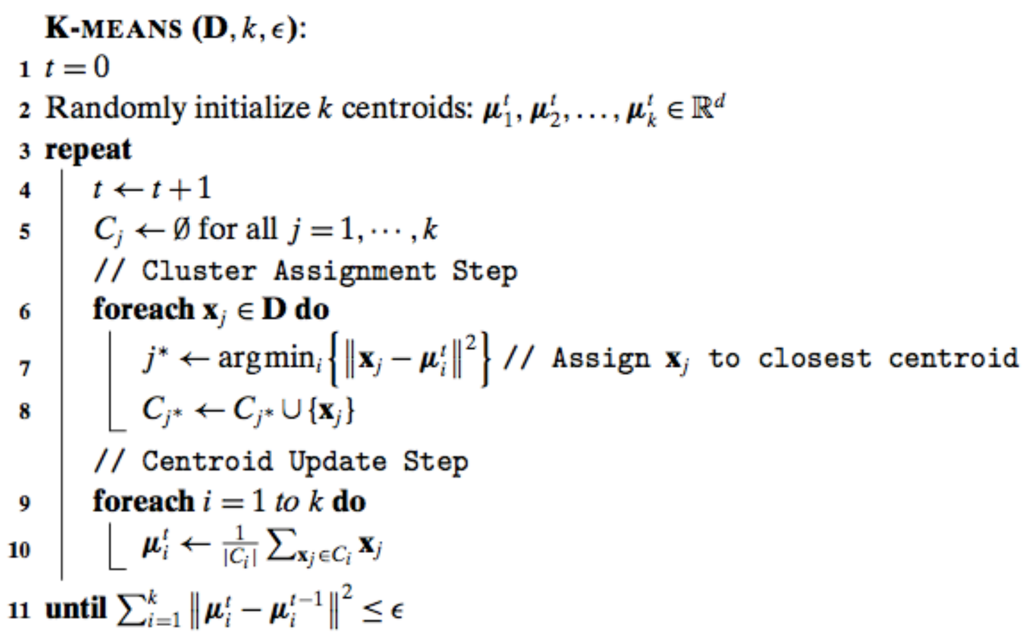
*Note: This algorithm has been sourced from 'DATA MINING AND ANALYSIS.'[1]

K-Means Clustering Algorithm:

Computational complexity: $o(tnkd)$

Where 't' is the number of iterations to convergence, and 'n' is the number of tuples.

Figure 5.6: K-means Clustering Algorithm



```
K-MEANS ($\mathbf{D}, k, \epsilon$):
t = 0
Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$
repeat
    $t \leftarrow t + 1$
    $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$
    // Cluster Assignment Step
    foreach $\mathbf{x}_j \in \mathbf{D}$ do
        $j^* \leftarrow \arg\min_i \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_i^t\|^2 \right\}$ // Assign $\mathbf{x}_j$ to closest centroid
        $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$
    // Centroid Update Step
    foreach $i = 1$ to $k$ do
        $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$
until $\sum_{i=1}^k \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \le \epsilon$
```

*Note:

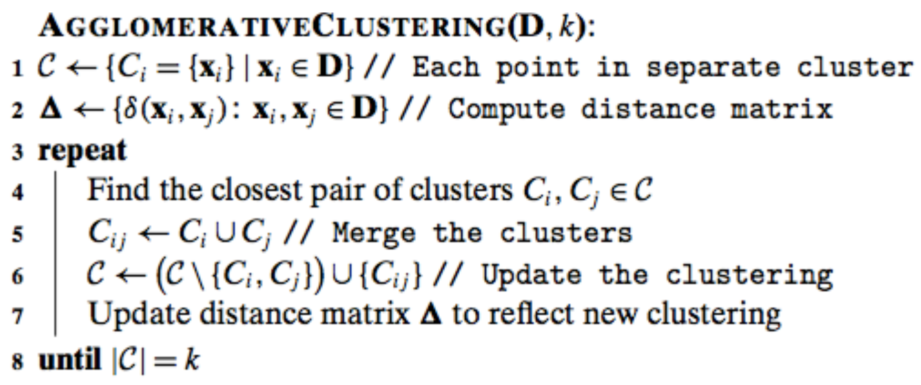
This algorithm has been sourced from 'DATA MINING AND ANALYSIS.'[1]

Agglomerative Hierarchical Clustering Algorithm:

Computational complexity: $o(n^2 logn)$

Where 'd' is the number of dimensions and 'n' is the number of tuples.

Figure 5.7: Agglomerative Hierarchical Clustering Algorithm



```
AGGLOMERATIVECLUSTERING($\mathbf{D}, k$):
$\mathcal{C} \leftarrow \{C_i = \{\mathbf{x}_i\} \mid \mathbf{x}_i \in \mathbf{D}\}$ // Each point in separate cluster
$\boldsymbol{\Delta} \leftarrow \{\delta(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i, \mathbf{x}_j \in \mathbf{D}\}$ // Compute distance matrix
repeat
    Find the closest pair of clusters $C_i, C_j \in \mathcal{C}$
    $C_{ij} \leftarrow C_i \cup C_j$ // Merge the clusters
    $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_i, C_j\}) \cup \{C_{ij}\}$ // Update the clustering
    Update distance matrix $\boldsymbol{\Delta}$ to reflect new clustering
until $|\mathcal{C}| = k$
```

*Note: This algorithm has been sourced from 'DATA MINING AND ANALYSIS.' [1]

CHAPTER 06

IMPLEMENTATION AND RESULT

Classification Algorithms:

The following tables show the result produced by each classifier. Each classifier is grouped according to its algorithm. The ‘PARAMETERS’ column states the attributes on which the classifier has been trained and tested upon. The ‘AVERAGE’ tuple shows the average performance of each classifier for a specific algorithm. The ‘CONFUSION MATRIX’ shows the number of misclassified points, and their erroneous classes.

The testing and training data for all classifiers across all algorithms are consistent so that comparable results can be formulated.

All the testings were performed on 1.3 GHz Intel Core i5 processor.

The measuring distance used for k-nearest neighbour is Euclidean distance.

Kd-tree data structure has been used to optimise the performance of the k-nearest neighbours algorithm.

Generally, points have been misclassified between class ‘Versicolor’ and the class ‘Virginica’. This is because of a high degree of overlap between these classes.

Naive Baye’s Classifier has an average accuracy of 91.64% and a mean testing time of 0.03065 seconds. K-nearest neighbours classifier has an average accuracy of 90.82% and a mean testing of 0.03601 seconds. Decision Tree Classifier has an average accuracy of 89.93% and a mean testing time of 0.023582 seconds. We can see that 96% is the highest accuracy we can achieve with a computational time of 0.021303 seconds by using naive Baye’s algorithm trained on parameters ‘Sepal Width, Petal Length And Petal Width’ .

Naive Baye’s classifier has the highest average accuracy of considering all attributional combinations and the Decision Tree classifier has the lowest computational testing speed of all algorithms. K-nearest neighbours has the highest computational testing time whereas decision tree classifier has the lowest accuracy.

Table 6.1-Naive Baye's Classifier

PARAMETERS	ACCURACY	TESTING TIME	CONFUSION MATRIX	PERFORMANCE
Sepal Length, Sepal Width, Petal Length And Petal Width	94.6667%	0.1691s	23 0 0 0 22 3 0 1 26	3.8424
Sepal Length, Sepal Width And Petal Length	88%	0.020681s	23 0 0 0 21 4 0 5 22	4.3857
Sepal Width, Petal Length And Petal Width	96%	0.021303s	23 0 0 0 23 2 0 1 26	4.4681
Sepal Length, Sepal Width And Petal Width	94.6667%	0.023927s	23 0 0 0 22 3 0 1 26	4.4428
Sepal Length, Petal Length And Petal Width	94.6667%	0.018614s	23 0 0 0 22 3 0 1 26	4.4664
Sepal Width And Sepal Length	80%	0.010596s	22 1 0 0 19 6 0 8 19	4.3358
Petal Length And Sepal Length	89.3333%	0.016948s	23 0 0 0 20 5 0 3 24	4.4169
Petal Width And Sepal Length	90.6667%	0.013831s	23 0 0 0 22 3 0 4 23	4.4459
Petal Width And Sepal Length	90.6667%	0.016223s	23 0 0 0 22 3 0 4 23	4.4357
Petal Width And Sepal Width	94.6667%	0.013999s	23 0 0 0 23 2 0 2 25	4.4871
Petal Width And Petal Length	94.6667%	0.01191s	23 0 0 0 22 3 0 1 26	4.4964
AVERAGE VALUES	91.64%	0.03065s	23 0 0 0 22 3 0 3 24	4.3815

*Note: Self Generated data. This table shows the results attained by performing tests on classifiers running on Naive Baye's algorithm.

Table 6.2-K-Nearest Neighbours Classifier

ATTRIBUTES	ACCURACY	TESTING TIME	CONFUSION MATRIX	PERFORMANCE
Sepal Length, Sepal Width, Petal Length And Petal Width	96%	0.19601s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 22 & 3 \\ 0 & 0 & 27 \end{matrix}$	3.7519
Sepal Length, Sepal Width And Petal Length	94.6667%	0.026067s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 22 & 3 \\ 0 & 1 & 26 \end{matrix}$	4.4333
Sepal Width, Petal Length And Petal Width	93.3333%	0.027566s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 21 & 4 \\ 0 & 1 & 26 \end{matrix}$	4.4128
Sepal Length, Sepal Width And Petal Width	93.3333%	0.019711s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 20 & 5 \\ 0 & 0 & 27 \end{matrix}$	4.4476
Sepal Length, Petal Length And Petal Width	96%	0.024565s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 22 & 3 \\ 0 & 0 & 27 \end{matrix}$	4.4536
Sepal Width And Sepal Length	69.6667%	0.017959s	$\begin{matrix} 22 & 0 & 1 \\ 0 & 18 & 7 \\ 0 & 15 & 12 \end{matrix}$	4.1682
Petal Length And Sepal Length	89.3333%	0.015347s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 23 & 2 \\ 0 & 2 & 25 \end{matrix}$	4.4234
Petal Width And Sepal Length	93.3333%	0.018308s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 22 & 3 \\ 0 & 2 & 25 \end{matrix}$	4.4539
Petal Width And Sepal Length	88%	0.018493s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 18 & 7 \\ 0 & 2 & 25 \end{matrix}$	4.3953
Petal Width And Sepal Width	92%	0.014737s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 21 & 4 \\ 0 & 2 & 25 \end{matrix}$	4.4556
Petal Width And Petal Length	93.3333%	0.01731s	$\begin{matrix} 23 & 0 & 0 \\ 0 & 20 & 5 \\ 0 & 0 & 27 \end{matrix}$	4.4583
AVERAGE VALUES	90.82%	0.03601	$\begin{matrix} 23 & 0 & 0 \\ 0 & 21 & 4 \\ 0 & 2 & 25 \end{matrix}$	4.3494

*Note: Self Generated data. This table shows the results attained by performing tests on classifiers running on K-Nearest Neighbours algorithm.

Table 6.3-Decision Tree Classifier

ATTRIBUTES	ACCURACY	TESTING TIME	CONFUSION MATRIX	PERFORMANCE
Sepal Length, Sepal Width, Petal Length And Petal Width	92%	0.129402s	23 0 0 0 19 6 0 0 27	3.9724
Sepal Length, Sepal Width And Petal Length	92%	0.007909s	23 0 0 0 19 6 0 0 27	4.4864
Sepal Width, Petal Length And Petal Width	93.3333%	0.008138s	23 0 0 0 20 5 0 0 27	4.4994
Sepal Length, Sepal Width And Petal Width	88%	0.007875s	23 0 0 0 17 8 0 1 26	4.4422
Sepal Length, Petal Length And Petal Width	92%	0.009617s	23 0 0 0 19 6 0 0 27	4.4785
Sepal Width And Sepal Length	69.3333%	0.0064s	22 1 0 2 16 7 0 13 14	4.4212
Petal Length And Sepal Length	92%	0.006715s	23 0 0 0 19 6 0 0 27	4.4915
Petal Width And Sepal Length	94.6667%	0.004689s	23 0 0 0 23 2 0 2 25	4.5291
Petal Width And Sepal Length	93.3333%	0.007387s	23 0 0 0 20 5 0 0 27	4.5028
Petal Width And Sepal Width	89.3333%	0.006183s	23 0 0 0 18 7 0 1 26	4.4647
Petal Width And Petal Length	93.3333%	0.005095s	23 0 0 0 20 5 0 0 27	4.5131
AVERAGE	89.93%	0.023582s	23 0 0 0 19 7 0 1 25	4.3942

*Note: Self Generated data. This table shows the results attained by performing tests on classifiers running on Decision Tree Classification**.**

Clustering Algorithms:

The following tables show the result produced by each cluster model. Each model is grouped according to its algorithm. The 'PARAMETERS' column states the attributes on which the cluster model has operated on. The 'AVERAGE' tuple shows the average performance of each classifier for a specific algorithm. The 'PURITY' and 'SILHOUETTE VALUE' show the proportion of data points clustered together which belong to the same class in the test data and density of points within a cluster.

The complete Iris dataset has been used for testing both these algorithms.

Table 6.4-Agglomerative Hierarchical Clustering

PARAMETERS	PURITY	TESTING TIME	SILHOUETTE VALUE	PERFORMANCE
Sepal Length, Sepal Width, Petal Length And Petal Width	68%	0.009507s	0.6184	0.4165
Sepal Length, Sepal Width And Petal Length	68%	0.010582s	0.6433	0.4328
Sepal Width, Petal Length And Petal Width	68%	0.007566s	0.5985	0.4039
Sepal Length, Sepal Width And Petal Width	66.67%	0.004728s	0.5999	0.3981
Sepal Length, Petal Length And Petal Width	69.33%	0.005387s	0.5934	0.4092
Sepal Width And Sepal Length	35.33%	0.005569s	0.1028	0.0361
Petal Length And Sepal Length	67.33%	0.005797s	0.4068	0.2723
Petal Width And Sepal Length	34.67%	0.00942s	-0.3799	-0.1305
Petal Width And Sepal Length	66.67%	0.008267s	0.8019	0.5302
Petal Width And Sepal Width	66.67%	0.02518s	0.6777	0.4406
Petal Width And Petal Length	67.33%	0.008337s	0.6221	0.4154
AVERAGE VALUE	61.63%	0.009122s	0.4804	0.3295

*Note: Self Generated data. This table shows the results attained by performing tests on clustering models running on Agglomerative hierarchical clustering.

All the testings were performed on 1.3 GHz Intel Core i5 processor.
The measuring distance used for k-means and agglomerative hierarchical clustering is Euclidean distance.The number of clusters were preset in both models to 3 so that a comparison can be made between the result and the known data.
For k-means clustering the process has been replicated 4 times, so the most accurate clustering result has been chosen of all the iterations. The testing time given in the table is an average of all the 4 iterations. These iterations have been performed to negate the effect of a starting point in k-means algorithm.

Table 6.5-K-Means Clustering

ATTRIBUTES	PURITY	TESTING TIME	SILHOUETTE VALUE	PERFORMANCE
Sepal Length, Sepal Width, Petal Length And Petal Width	89.33%	0.0097835	0.7357	0.6508
Sepal Length, Sepal Width And Petal Length	88%	0.0075335	0.7330	0.6402
Sepal Width, Petal Length And Petal Width	95.33%	0.00829	0.7659	0.7241
Sepal Length, Sepal Width And Petal Width	82.67%	0.0090135	0.6654	0.5452
Sepal Length, Petal Length And Petal Width	89.33%	0.00589225	0.7523	0.6681
Sepal Width And Sepal Length	82%	0.00891975	0.6201	0.504
Petal Length And Sepal Length	88%	0.00694425	0.7560	0.6607
Petal Width And Sepal Length	82.67%	0.0070455	0.6813	0.5593
Petal Width And Sepal Length	92.67%	0.0097835	0.7612	0.6985
Petal Width And Sepal Width	92.67%	0.0075335	0.7067	0.65
Petal Width And Petal Length	96%	0.00829	0.8055	0.7669
AVERAGE VALUE	88.97%	0.008093	0.7257	0.6425

*Note: Self Generated data. This table shows the results attained by performing tests on clustering models running on Agglomerative hierarchical clustering.

CHAPTER 07

CONCLUSION AND FUTURE ENHANCEMENT

Classification Algorithms:

The performance of the algorithms can be compared by normalising both the accuracy and time and then calculating a ratio.The performance ratio can be calculated by finding a ratio between normalised accuracy and normalised time. $Performance = ln(a)/e^t$, where a=Accuracy and t=testing time. Decision Tree Classifier has an average performance ratio of 4.394, followed by naive Baye's 4.381 and then the K-Nearest Neighbours algorithm which has a ratio of 4.349. Hence, for general application we see that decision tree classification is the best classification algorithm. Although, for mid-sized datasets where accuracy has higher priority than testing time, naive Baye's algorithm is a preferred option.

The figure 7.1 shows the time and accuracy bar graphs of different classifiers.

Figure 7.1-Accuracy Comparison

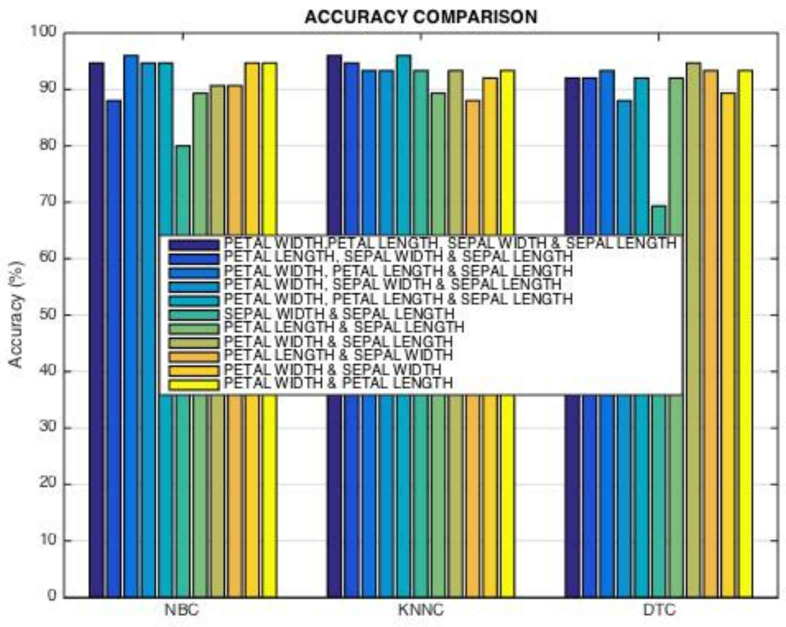


Figure 7.2(a)-Time Comparison

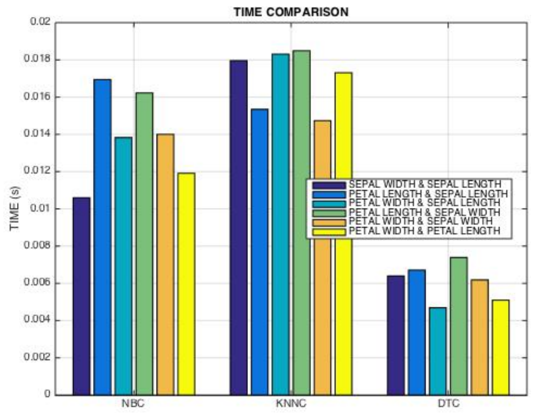
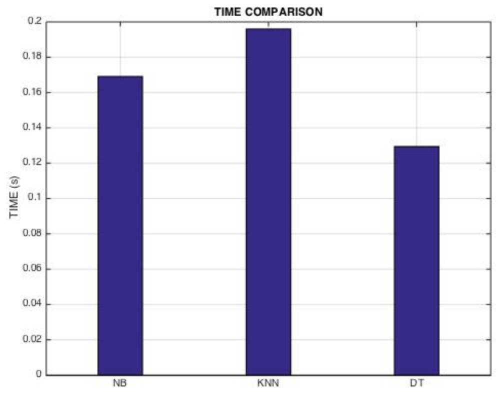


Figure 7.2(b)-Time Comparison

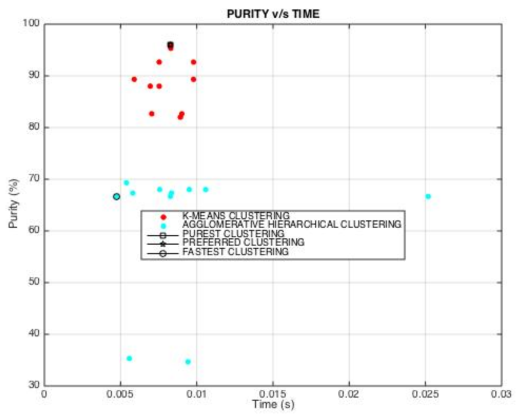


*NOTE: All figures are self generated.

Clustering Algorithms:

The performance of clustering algorithms is measured by all 3 parameters involved in its measure. The performance can be defined as: $Performance = s * (p/e^t)$, where s=silhouette value, p=purity and t is the testing time of the graph. This measure would enable us to see the quality of the result our model produces. Using these values we see that k-means clustering algorithm is more optimal than hierarchical clustering and has an average performance metric of ‘0.6425’ whereas hierarchical routing has a performance metric of ‘0.3925’. As shown by the graph 7.3 below, the k-means algorithm outclasses hierarchical clustering and has an average purity of 88.97% as compared to 61.63% of hierarchical routing. K-means clustering also has faster testing time of 0.008093 than 0.009122 of agglomerate hierarchical clustering.

Figure 7.3(a)(on the left)-Purity v/s Time and Figure 7.3(b)(on the right)-Purity v/s Silhouette Value



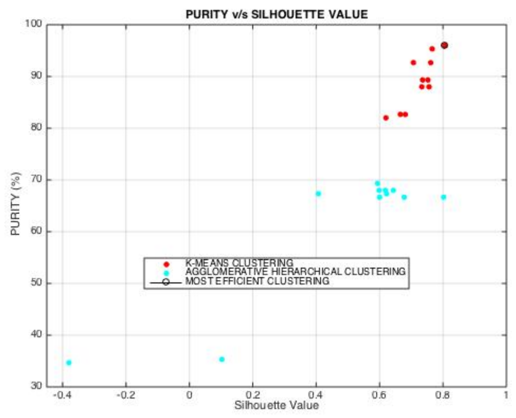
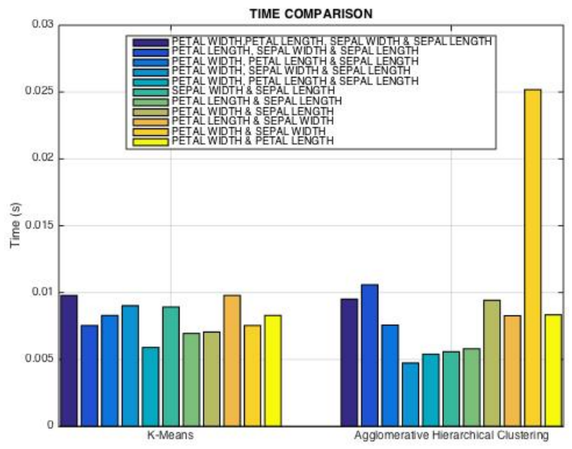
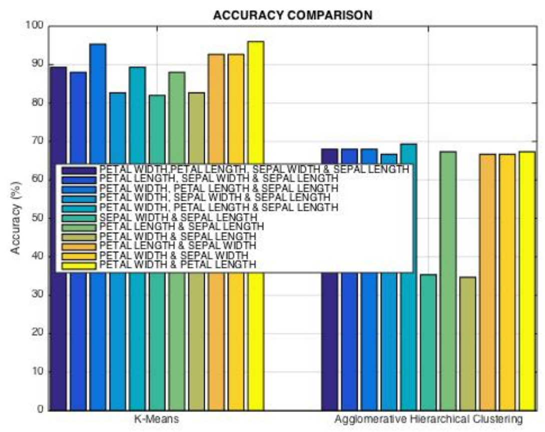


Figure 7.4(a)(on the left)-Time Comparison and Figure 7.4(b)(on the right) -Accuracy Comparison



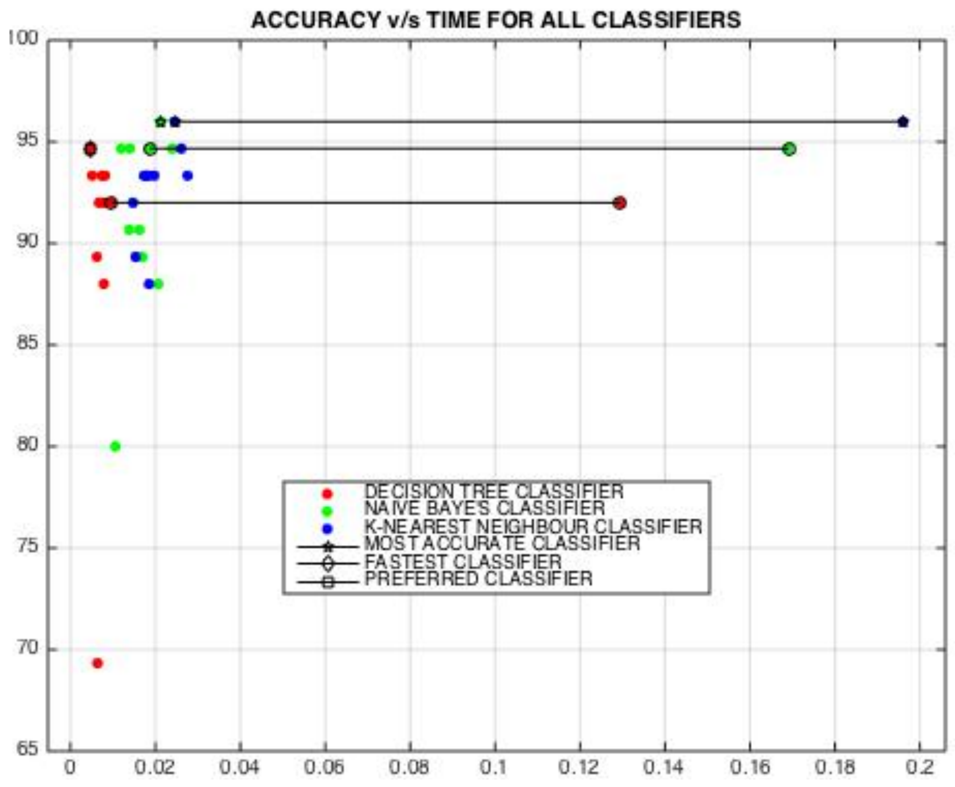


*Note: All figures are self generated.

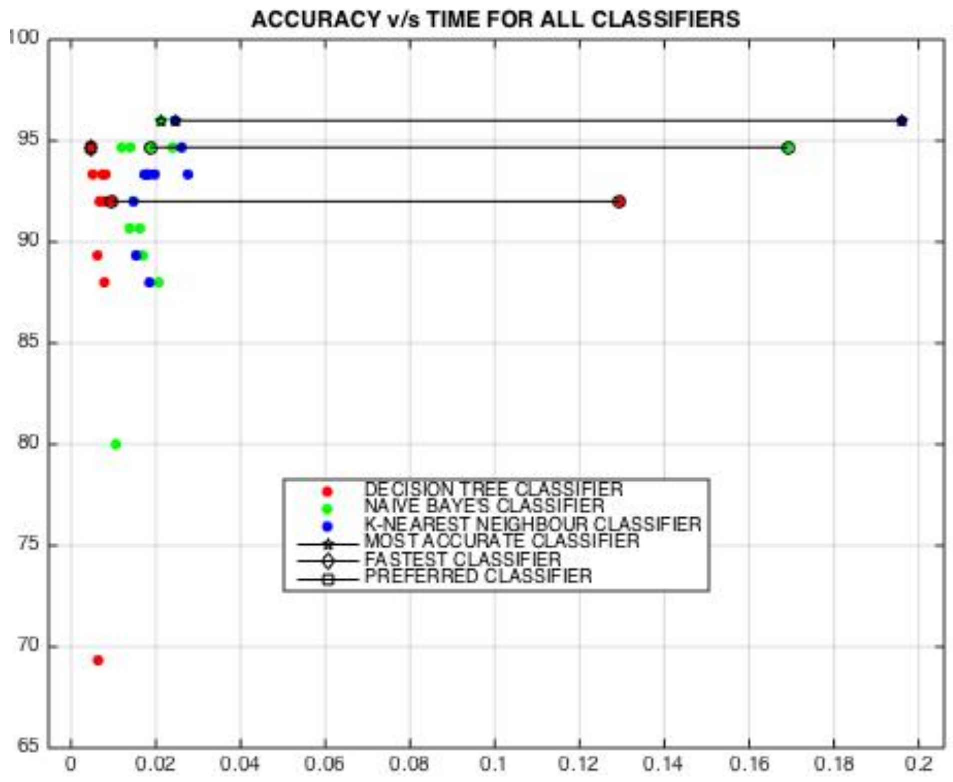
FUTURE ENHANCEMENT:

Figure 7.5 shows 2 classifiers, which are joined by a single line produce the same accuracy, but with different computational times. During experiment it has been noticed that the classifier classifying on the basis of all 4 attributes, and the classifier classifying on the basis of sepal length, petal length and petal width produce similar results for all 3 classifiers. Hence, we can observe that classification based on a combination of attributes in the entire dataset, can be used to model the classification done on an entire attribute set. Hence we can optimise performance of classification, by using this specific combination of attributes. We can thus reduce the dimensionality of a dataset to be classified, and greatly increase speed. In the Iris dataset we see that the classification based on sepal length, petal length and petal width for Naive Baye's algorithm runs 9.085 times faster, for K-Nearest Neighbours algorithm runs 8.32% faster and for Decision Tree runs 13.456 times faster. Hence, an enhancement to this project would be the optimisation of performance of classification algorithms by reducing the dimensionality of data.

Figure 7.5-Accuracy v/s Time



*NOTE: All figures are self generated.



Looking at the below table 7.1 showing that the performance score of both clustering and classification models based on 3 attributes outperforms that based on 4 attributes, we can hypothesise that a large dataset can be modelled by smaller representation of the same dataset. We can also see that despite lower accuracy in modelling with 2 attributes, they have a high performance score as they compute results significantly faster. Because we were functioning on a dataset with only 4 dimensions, the difference is not notable, but with detests regarding a higher number of dimensions, and especially categorical dimensions, this reduction in dimensionality will result in a significant difference in time between the reduced and the complete dataset.

In reference to the table and the calculated performance parameters the decision tree classifiers working on 2 attributes has the most optimum classifiers. It has an average accuracy of 87% and an average computational time of 0.006078 seconds. The best classifier is the decision tree classifier trained on ‘Petal Width And Sepal Length’.

The k-means clustering algorithm, significantly outperforms the agglomerative hierarchical algorithm. The best performer was the k-means cluster model on all 4 attributes with an accuracy of 89.33% and testing time of 0.0097835.

Table 7.1-Performance Comparison

ALGORITHM	4 ATTRIBUTES	3 ATTRIBUTES	2 ATTRIBUTES
NAIVE BAYE’S	3.8424	4.4407	4.4284
K-NEAREST NEIGHBOURS	3.7519	4.4368	4.4392
DECISION TREE	3.9724	4.4766	4.4871
K-MEANS	0.6508	0.6444	0.6399
AGGLOMERATIVE HIERARCHICAL	0.4165	0.4111	0.22974

*Note: Self Generated Data. This table shows a performance comparison analysis with classifiers trained on different number of attributes.

Bibliography

1. Mohammed J. Zaki and Wagner Meira, ‘DATA MINING AND ANALYSIS ’.
2. Bunheang Tay, Jung Keun Hyun and Sejong Oh, ‘A Machine Learning Approach for Specification of Spinal Cord Injuries Using Fractional Anisotropy Values Obtained from Diffusion Tensor Images’.
3. S. Rasoul Safavian and David Landgrebe, ’A Survey Of Decision Tree Methodology’.
4. Yuguang Huang and Lei Li, ‘Naive Bayes classification algorithm based on small sample set’.
5. T. Kanungo , D.M. Mount , N.S. Netanyahu C.D. Piatko , R. Silverman and A.Y. Wu , ‘An efficient k-means clustering algorithm: analysis and implementation’.
6. Ying Zhao and George Karypis, ‘Evaluation of hierarchical clustering algorithms for document datasets’.
7. Min-Ling Zhang and Zhi-Hua Zhou, ‘A k-nearest neighbor based algorithm for multi-label classification’.
8. Ahmad Ashari, Iman Paryudi and A Min Tjoa, ‘Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool’.
9. Michael Steinbach, George Karypis and Vipin Kumar, ‘A Comparison of Document Clustering Techniques’.
10. Rich Caruana and Alexandru Niculescu-Mizil, ‘An Empirical Comparison of Supervised Learning Algorithms’.