Chapter 1

Introduction
CHAPTER 1

INTRODUCTION

1.1 Introduction to Data Mining

The past two decades have seen a dramatic increase in the amount of information or data being stored in electronic format. This accumulation of data has taken place at an explosive rate. It has been estimated that the amount of information in the world doubles every 20 months and the size and number of databases are increasing even faster. The increase in use of electronic data gathering devices such as point-of-sale or remote sensing devices has contributed to this explosion of available data. Figure 1.1 illustrates the data explosion.

![Figure 1.1 The Growing Base of Data](image)

Data storage became easier as the availability of large amounts of computing power at low cost. The falling cost of processing power and storage has made the data cheap [1]. There was also the introduction of new machine learning methods for knowledge representation based on logic programming, in addition to traditional statistical analysis of data. The new methods [2] tend to be computationally intensive and hence demands for more processing power.
Data, Information, and Knowledge

Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data in different formats and different databases. This includes:

- Operational or transactional data such as, sales, cost, inventory, payroll, and accounting
- Non operational data, such as industry sales, forecast data, and macro economic data
- Meta data - data about the data itself, such as logical database design

The patterns, associations, or relationships among all these data can provide information. For example, analysis of retail point of sales transaction data can yield information on which products are selling and when.

Information can be converted into knowledge about historical patterns and future trends. For example, summary information on retail supermarket sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus, a manufacturer or retailer could determine which items are most susceptible to promotional efforts.

Over the past few years, research and development in data mining has made great progress. It combines research areas such as databases, machine learning, artificial intelligence, statistics, automated scientific discovery, data visualization, decision science, and high performance computing [3]. While each of these areas contributes in its specific ways, data mining focuses on the value that is added by creative combination of the contributing areas in order to produce innovative solutions to the data analysis task. A large number of research and application papers have appeared in the literature. It is considered an important area for major cost savings and potential revenue with immediate applications in business, decision systems, information management, communication, scientific research and technology development. Many successful applications have been reported in various sectors such as marketing [4], finance, banking and telecommunications [5] and manufacturing [6].
Some examples of business applications include: Using data mining techniques to analyze customer databases so that potential customers can be selected more precisely.

The Business Week magazine [7] estimated that more than 50% of all U.S. retailers use or plan to use such approach of database marketing. Those uses the approach have obtained good results. Using data mining techniques for fraud detection from detecting cellular cloning fraud to identifying financial transactions that may indicate money-laundering activities. In short, data mining systems typically help businesses to expose previously unrecognized patterns in their databases. These information "nuggets" are used to improve profits, enhance customer service, and ultimately achieve a competitive advantage.

It has now been recognized that mining for information and knowledge from large databases and documents will be the next revolution in database systems. We can expect the next generation information systems to be more intelligent in that they are not only data intensive but also knowledge rich.

1.2 Steps in Knowledge Discovery in Databases (KDD) Process

KDD refers to the overall process of discovering useful knowledge from data. It involves the evaluation and possibly interpretation of the patterns to make the decision of what qualifies as knowledge. It also includes the choice of encoding schemes, preprocessing, sampling and projections of the data prior to the data mining step. Figure 1.2 summarizes some of the stages/processes identified in data mining and knowledge discovery. The phases depicted start with the raw data and finish with the extracted knowledge which was acquired as a result of the following stages [8].

1.2.1 Selection

Selecting or Segmenting the data according to some criteria, for example all those people who own a car among the people in the country, in this way subsets of the data can be determined. Many of the attributes in the data may be irrelevant to the classification or prediction task. For example, recording the day of the week on which a bank loan application was filed is unlikely to be relevant to the success of the
application. Furthermore, other attributes may be redundant. Hence, relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process.

Figure 1.2 An Outline of the Steps of the KDD Process

1.2.2 Preprocessing

For years, learning from low-quality information sources has been a major concern for the data mining community. When the underlying data bear a certain amount of errors, a common practice is to adopt data preprocessing techniques, such as noise cleaning [9], error detection [10, 11], and data imputation [12] to enhance the data quality for the benefit of the succeeding mining procedures. For many real-world applications, data preprocessing is a must to enhance the data quality before the actual mining process takes place, although the data cleaning or error correction process could possibly incur various negative impacts such as information loss or new errors. Extensive research studies have shown that if properly and carefully adopted, data pre-processing may enhance the data quality and help to build a more robust classification model [10, 13, 14].
In the data cleaning stage where certain information is removed which is deemed unnecessary and may slow down queries. This refers to the preprocessing of data in order to remove or reduce noise and the treatment of missing values (e.g. by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics). Most classification algorithms such as Iterative Dichotomizer 3, Classification and Regression Trees have some mechanisms for handling noise or missing data. This step can help to reduce confusions during learning. For example, unnecessary to note the sex of a patient when studying pregnancy. Also the data is reconfigured to ensure a consistent format as there is a possibility of inconsistent formats because the data is drawn from several sources e.g. sex may be recorded as for female (f) or (1) and for male (m) or (0).

1.2.3 Transformation

The data is not merely transferred across but transformed in that overlays may be added such as the demographic overlays commonly used in market research. The data is made useable and navigable. The data can be generalized to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous-valued attributes. For example, numeric values for the attribute income may be generalized to discrete ranges such as low, medium, and high. Similarly, nominal-valued attributes, like street, can be generalized to higher-level concepts, like city. Since generalization compresses the original training data, fewer input/output operations may be involved during learning.

The data may also be normalized, particularly when neural networks or methods involving distance measurements are used in the learning step [15]. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurement, for example, this would prevent attributes which have initially large ranges.
1.2.4 Data mining

Data mining is the process of analyzing data to identify patterns or relationships from different perspectives and summarizing it into useful information; information that can be used to increase revenue, cut costs, or both. It is the process of automatically searching large volumes of data for patterns using tools such as classification, association rule mining, clustering, etc.

1.2.5 Interpretation and evaluation

The patterns identified by the system are interpreted into knowledge which can then be used to support human decision-making e.g. prediction and classification tasks, summarizing the contents of a database or explaining observed phenomena. To mine information from an Application Domain, the following Data mining processes have been usually used.

1. Learning the application domain
2. Creating a target data set
3. Data Cleaning and pre-processing
4. Data reduction, Transformation, Normalization, Projection etc.
5. Choosing the Data mining Model
6. Choosing the data mining algorithm
7. Data mining
8. Interpretation
9. Using the discovered Knowledge

Based on the data collected, data mining algorithms are used to either produce a description of the data stored, or predict an outcome. Different kinds of algorithms are used to achieve either one of these tasks. However, in the overall KDD process, any mixture of these tasks may be called upon to achieve the desired results. There are two kinds of data mining tasks. They are

a) Description tasks: These tasks describe the data being mined and they are

i) Summarization: To extract compact patterns that describe subsets of data. The method used to achieve this task are Association Rule algorithms.
ii) Segmentation or Clustering: To separate data items into subsets that are similar to each other. Partition-based clustering algorithms are used to achieve this task.

iii) Change and Deviation Detection: To detect changes in sequential data (such as protein sequencing, behavioral sequences, etc.).

iv) Dependency Modeling: To construct models of causality within the data.

b) Prediction tasks: To predict some field(s) in a database based on information in other fields.

i) Classification: To predict the most likely state of a categorical variable (its class).

ii) Regression: To predict results that are numeric continuous variables.

1.3 Classification

Classification of large data set is an important problem in data mining. The goal of classification is to assign a new object to a class from a given set of classes based on the attribute values of this object. Different methods have been proposed for the task of classification and are explained elsewhere. For a database with a number of records and for a set of classes such that each record belongs to one of the given classes, the problem of classification is to decide the class to which a given record belongs. But there is much more to this than just simply classifying.

Data classification is a two-step process. In the first step, a model is built by describing a predetermined set of data classes or concepts. The model is constructed by analyzing databases tuples described by attributes. Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the class label attribute. The data tuples are analyzed to build the model collectively from the training data set. The individual tuples making up the training set are referred to as training samples and are randomly selected from the sample population. Since the class label
of each training sample is provided, this step is also known as supervised learning [16].

In the second step, the model is used for classification. First, the predictive accuracy of the model or classifier is estimated. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. For each test sample, the known class label is compared with the learned model's class prediction for that sample. Note that if the accuracy of the model were estimated based on the training data set, this estimate could be optimistic since the learned model is most relevant to the training data set only [16].

The classification problem is also concerned with generating a description or a model for each class from the given data set. Here, the type of classification concerned is called supervised classification. In supervised classification, there is a training data set of records and for each record of this set, the respective class to which it belongs is also known. Using the training set, the classification process attempts to generate the descriptions of the classes, and these descriptions help to classify unknown records. In addition to the training set, there is also a test data set, which is used to determine the effectiveness of a classification. The accuracy of the classifier is determined by the percentage of the test examples that are correctly classified [17].

There are two different types of attributes available. Attributes whose domains are numerical are called the numerical attribute, and the attribute whose domain is not numerical are called the categorical attributes. There is one distinguished attribute called the class label. The goal of the classification is to build a concise model that can be used to predict the class of the records whose class label is not known [17].

The Classifier's Output may be represented in different levels. They are
a) Abstract level: A classifier only outputs a unique class.
b) Rank level: The classifier ranks all the classes in a queue with the label at the top being the first choice.
c) Measurement level: The classifier attributes each class a measurement value for the degree that the input has the class.

1.4 Comparison of classification methods

Various classifiers can be compared and evaluated according to the criteria such as predictive accuracy, speed, robustness, scalability and interpretability. Accuracy and scalability are two important factors in assessing the success of Classifier.

Predictive Accuracy

Accuracy is an important factor in assessing the success of data mining. Classifier accuracy evaluates how accurately a given classifier will label the future data. This refers to the ability of the model to correctly predict the class label of new or previously unseen data. Among several classifiers, no one is always superior to the others in terms of accuracy. The number of training data affects the accuracy of a classifier. If classification accuracy is acceptable, the classification structure is then used to classify newly presented instances whose classification is unknown. In the field of pattern recognition, a number of studies are going on for improving and comparing the accuracy of different classifiers [18-22].

Scalability

This refers to the ability to construct the model efficiently given large amounts of data. With the recent growth in the amount of data collected by information systems there is a need for algorithms that can handle large datasets.

Catlett [23] has examined two methods for efficiently growing decision trees from a large database by reducing the computation complexity required for induction. However, the Catlett method requires that all data will be loaded into the main memory before induction. Namely, the largest dataset that can be induced is bounded by the memory size.
Chan and Stolfo [24] suggest partitioning the datasets into several disjoint datasets, such that each dataset is loaded separately into the memory to construct the classifier. The individual classifiers are then combined to create a single classifier. However, the experimental results indicate that partition may reduce the classification performance, meaning that the classification accuracy of the combined classifier is not as good as the accuracy of a single classifier induced from the entire dataset.

Mehta et al. [25] have proposed an algorithm known as SLIQ (Supervised Learning in Quest where Quest is the Data mining project at the IBM Almaden research centre) that does not require loading the entire dataset into the main memory, instead it uses secondary memory (disk) namely a certain instance is not necessarily resident in main memory all the time. SLIQ creates a single classifier from the entire dataset. However, this method also has upper limit for the largest dataset that can be processed because it uses a data structure that scales with the dataset size and this data structure is required to be resident in main memory all the time.

Shafer et al. [26] have presented a similar solution known as SPRINT (Scalable PaRallelizable INndution of decision Trees). This algorithm induces decision trees relatively quickly and removes all of the memory restrictions from decision tree induction. SPRINT scales any impurity based split criteria for large datasets.

Most of the algorithms require rebuilding the classifier from scratch for reflecting new data that has become available. Several researches have addressed the issue of updating classifiers incrementally. Utgoff [27] presents several methods for updating classifiers incrementally. An extension to the CART algorithm that is capable of inducing incrementally is described by Crawford [28].

1.5 Data Classification Methods

The following list shows some of the categories of classification algorithms generally used in data mining applications.
Statistical Algorithms

Statistical packages such as Statistical Analysis Systems (SAS) and Statistical Packages for the Social Sciences (SPSS) have been used by analysts to detect unusual patterns and explain patterns using statistical models. Such systems have their place and will continue to be used [29].

Neural Networks

Artificial neural networks mimic the pattern-finding capacity of the human brain and hence some researchers have suggested applying Neural Network algorithms to pattern-mapping. Neural networks have been applied successfully in a few applications that involve classification [15].

Genetic algorithms

Optimization techniques that use process such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution[6].

Nearest neighbor method

A technique that classifies each record in a dataset based on a combination of the classes of the K record(s) most similar to it in a historical dataset. Sometimes this method is called as K-nearest neighbor technique [30].

Rule induction

The extraction of useful if-then rules from data based on statistical significance[31, 32].

Data visualization

The visual interpretation of complex relationships in multidimensional data.

As mentioned above, there are so many methods for data classification. Generally the selection of a particular method may depend on the application. The
selection of a particular methodology for data classification may depend on the volume of data and the number of classes present in that data. Further, the classification algorithms are designed in a custom manner for a specific purpose to solve a particular classification scenario.

Because of the large size of the database, it may not be possible to use the entire training data set to construct the classifier. Random sampling has been often used to handle large datasets when building a classifier. The size and other characteristics of the selected sample determine whether the sample used in modeling is a good representative of the entire database. Each observation or case in the database has an equal chance of being included in the sample. Representative samples require a relatively shorter time to clean, explore, and develop and validation of models. They are therefore more cost effective than using entire databases.

1.6 Measuring the Performance of Classifiers

The performance of classification algorithms is usually examined by evaluating the accuracy of the classification. However, since classification is often a fuzzy problem, the correct answer may depend on the user. Traditional algorithm evaluation approaches such as determining the space and time overhead can be used, but these approaches are usually secondary.

Classification accuracy is usually calculated by determining the percentage of tuples placed in the correct class. This ignores the fact that there also, may be a cost associated with an incorrect assignment to the wrong class. These perhaps should also be determined. The performance of classifications can be examined by the information retrieval systems.

Given a specific class \( C_j \), and a database tuple \( t_i \), that tuple may or may not be assigned to that class while its actual membership may or may not be in that class. With only two classes there are four possible outcomes with the classifications, as given below. The first and third case represents the correct actions. The remaining two
cases represent the incorrect actions. Suppose if we are having only two classes A and B. The class prediction is shown in Table 1.1.

1. True positive (TP): \( t_i \) predicted to be in \( C_j \) and is actually in it.
2. False positive (FP): \( t_i \) predicted to be in \( C_j \) but is not actually in it.
3. True negatives (TN): \( t_i \) not predicted to be in \( C_j \) and is not actually in it.
4. False negatives (FN): \( t_i \) not predicted to be in \( C_j \) but is actually in it.

<table>
<thead>
<tr>
<th>Classification into class A</th>
<th>Class prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned Class A in Class A</td>
<td>Assigned class B in class A</td>
</tr>
<tr>
<td>Assigned Class A in Class B</td>
<td>Assigned Class B in class B</td>
</tr>
</tbody>
</table>

A confusion matrix (Kohavi and Provost, 1998) [33] contains information about actual and predicted classifications done by a classification system. It illustrates the accuracy of the solution to a classification problem. Given \( m \) classes, a confusion matrix is an \( m \times m \) matrix where entry \( C_{ij} \) indicates the number of tuples from the Database that were assigned to class \( C_j \) but where the correct class is \( C_i \). Performance of such systems is commonly evaluated using the data in the matrix. Table 1.2 shows the confusion matrix for a two class classifier. The entries in the confusion matrix have the following meaning in the context of finding the accuracy:

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>Negative</td>
</tr>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td>Actual</td>
<td>True negative</td>
</tr>
<tr>
<td></td>
<td>c</td>
</tr>
<tr>
<td>Positive</td>
<td>False positive</td>
</tr>
</tbody>
</table>
• a is the number of correct predictions that an instance is negative
• b is the number of incorrect predictions that an instance is positive
• c is the number of incorrect predictions that an instance negative
• d is the number of correct predictions that an instance is positive.

Several standard terms have been defined for the 2 class matrix [33]:

• The accuracy (AC) is the proportion of the total number of predictions that were correct. It is determined using the equation
  \[ AC = \frac{a+d}{a+b+c+d} \]  
  (1.1)

• The recall or true positive rate (TP) is the proportion of positive cases that were correctly identified as calculated using the equation
  \[ TP = \frac{d}{c+d} \]  
  (1.2)

• The false positive rate (FP) is the proportion of negatives cases that were incorrectly classified as positive as calculated using the equation
  \[ FP = \frac{b}{a+b} \]  
  (1.3)

• The true negative rate (TN) is defined as the proportion of negatives cases that were classified correctly as calculated using the equation
  \[ TN = \frac{a}{a+b} \]  
  (1.4)

• The false negative rate (FN) is the proportion of positives cases that were incorrectly classified as negative as calculated using the equation
  \[ FN = \frac{c}{c+d} \]  
  (1.5)

• Finally, precision (P) is the proportion of the predicted positive cases that were correct as calculated using the equation
  \[ P = \frac{d}{b+d} \]  
  (1.6)

In machine learning, current research has shifted away from simply presenting accuracy results when performing an empirical validation of new algorithms. This is especially true when evaluating algorithms that output
probabilities of class values. Provost et al. (1998) [33] have argued that simply using accuracy results can be misleading. They recommended when evaluating binary decision problems to use Receiver Operating Characteristic (ROC) curves, which show how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. ROC graphs are another way besides confusion matrix to examine the performance of classifiers (Swets, 1988) [34]. However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew.

A ROC curve provides a graphical representation of the relationship between the true-positive and false-positive prediction rate of a model. The curve was originally used in information retrieval to examine fallout (percentage of retrieved that are not relevant) versus recall (percentage of retrieved that are relevant). A ROC graph is a plot with the false positive rate on the x axis and the true positive rate on the y axis [35].

The y-axis corresponds to the sensitivity of the model, i.e. how well the model is able to predict true positives (real cleavages) from sites that are not cleaved, and the y-coordinates are calculated as

$$ Y = \frac{TP}{TP + FN} $$

The x-axis corresponds to the specificity (expressed on the curve as 1-specificity), i.e. the ability of the model to identify true negatives. An increase in specificity (i.e. a decrease along the x-axis) results in an increase in sensitivity. The x-coordinates are calculated as

$$ X = 1 - \left[ \frac{TN}{TN + FP} \right] $$
Features of ROC Graphs

- An ROC curve or point is independent of class distribution or error costs (Provost et al., 1998) [27].
- An ROC graph encapsulates all information contained in the confusion matrix since $FN$ is the complement of $TP$ and $TN$ is the complement of $FP$ (Swets, 1988) [28].
- ROC curves provide a visual tool for examining the tradeoff between the ability of a classifier to correctly identify positive cases and the number of negative cases that are incorrectly classified.

Figure 1.3 shows the ROC Curve. The point (0,1) is the perfect classifier: it classifies all positive cases and negative cases correctly. It is (0,1) because the false positive rate is 0 (none), and the true positive rate is 1 (all). The point (0,0) represents a classifier that predicts all cases to be negative, while the point (1,1) corresponds to a classifier that predicts every case to be positive. Point (1,0) is the classifier that is incorrect for all classifications. In many cases, a classifier has a parameter that can be adjusted to increase TP at the cost of an increased FP or decrease FP at the cost of a decrease in TP. Each parameter setting provides a (FP, TP) pair and a series of such pairs can be used to plot an ROC curve. A non-parametric classifier is represented by a single ROC point, corresponding to its (FP,TP) pair [36].

![Figure 1.3 Receiver Operating Characteristic Curve](image)
The curve always goes through two points (0,0 and 1,1). 0,0 is where the classifier finds no positives (detects no alarms). In this case it always gets the negative cases right but it gets all positive cases wrong. The second point is (1,1) where everything is classified as positive. So the classifier gets all positive cases right but it gets all negative cases wrong [37].

The graph shown in figure 1.4 has three ROC curves representing excellent, good, and worthless tests plotted on the same graph. A classifier that randomly guesses has ROC which lies somewhere along the diagonal line connecting (0,0) and (1,1) [38]. The ROC curve sometimes could be below the diagonal. It shows that for all threshold values its performance is worse than random. Alternatively the ROC curve may cross the diagonal. In this case its overall performance can be improved by selectively reversing the classifier's answer, depending upon the range of threshold values which put it below the diagonal. Figure 1.5 shows the ROC curve of a poor classifier in performance. It shows that the classifier could not be used to predict the future values [39].

![Figure 1.4 Different kinds of Accepted ROC Curves](image)

An ROC curve demonstrates several things:

1. It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
2. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate is the test.

3. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate is the test.

![ROC Curve with Poor Quality](image)

Figure 1.5 ROC curve with poor quality

1.7 Literature Review

In the research of pattern recognition, combining multiple classifiers has been studied for more than a decade and has reported improved performance over single classifier approaches [40-42]. The aim of a Multiple Classifiers System (MCS) is to determine an effective combination method that makes use of the benefits of each classifier but avoids the weaknesses. An MCS contains several homogeneous or heterogeneous classifiers. It is now an established research area known under different names in the literature such as committees of learners, mixtures of experts, classifier ensembles, combining classifiers and consensus theory [43].

The concept of MCS was proposed by Ching Y. Suen [44] with the aim to obtain improved recognition result in the character recognition area. In spite of its initial purpose, MCS is a general problem in various application areas of pattern recognition. There are two main reasons for the generation of MCS [45]. One of them is that there are a number of classification algorithms available in almost any one of the current pattern
recognition application areas. Complementary information may exist between some of these algorithms. The classical approach for a pattern recognition problem is to search for the best individual classification algorithm. Thus it is not possible to exploit the complementary information that other classification algorithms may encapsulate. Furthermore, the best classification algorithm for the classification task at hand is difficult to identify unless enough prior knowledge is available. The other reason is that for a specific recognition problem, there usually exist numerous types of features that may be too diversified to lump into one single classifier for decision-making. In the past ten years, much energy has been put into MCS [46-48].

Most studies on combining multiple classifiers for the improvement of classification performance have been actively performed in the field of Pattern recognition. These studies have mainly proposed some effective combination methods on how to combine the recognition results of individual classifiers [49-52]. Several combination methods have reported significant improvements in the classification performance with respect to the individual classifiers. However, only a few studies have been investigated on how to construct multiple classifier systems from available classifiers pool [53, 54]. Thus, the construction of multiple classifier systems still remains an important problem.

Roughly, existing integration techniques can be distinguished into two categories:
1. Combine base classifiers for the final decision. When classifying a test instance, the results from all base classifiers are combined to work out the final decision. This is referred as Classifier Combination (CC) techniques. In CC, individual classifiers are applied in parallel and their outputs are combined in some manner to achieve a "group consensus.

2. Select a single "best" classifier from base classifiers for the final decision, where each base classifier is evaluated with an evaluation set to explore its domain of expertise. When classifying an instance, only the "best" classifier is used to
determine the classification of the test instance. This technique is called classifier Selection (CS) techniques.

Previous classifier fusion algorithms include the majority vote [55], the Borda count [56], unanimous consensus [52, 56], thresholded voting [52], polling methods which utilize heuristic decision rules [57, 58], the "averaged Bayes classifier" [52], logistic regression to assign weights to the ranks produced by each classifier [56], Dempster-Shafer theory to derive weights for each classifier's vote [52, 59], and methods of multistage classification [60].

The CC techniques were categorized into three types [61], depending on the level of information being exploited. Type 1 makes use of class labels. Type 2 uses class labels plus a priority ranking assigned to each class. Finally, Type 3 exploits the measurements of each classifier and provides each classifier with some measure of support for the classifier's decision. The CS takes the opposite direction. Instead of adopting the combining techniques, it selects the "best" classifier to classify a test instance. Two types of techniques are usually adopted:

1. **Static Classifier Selection (SCS)**. The selection of the best classifier is specified during a training phase, prior to classifying a test instance [62, 63].

2. **Dynamic Classifier Selection (DCS)**. The choice of a classifier is made during the classification phase. We call it "dynamic" because the classifier used critically depends on the test instance itself [64-66].

Many existing data stream mining efforts are based on the Classifier Combination techniques [67-70], and as they have demonstrated, a significant amount of improvement could be achieved through the ensemble classifiers. However, given a data stream, it usually results in a large number of base classifiers, where the classifiers from the historical data may not support (or even conflict with) the learner from the current data. This situation is compounded when the underlying concept of the data stream experiences dramatic changes or evolving, or when the data suffers
from a significant amount of noise, because the classifiers learned from the data may vary dramatically in accuracy or in their domain of expertise (i.e., they appear to be conflictive). In these situations, choosing the most reliable one becomes more reasonable than relying on a whole bunch of likely contradictive base classifiers.

The two main reasons of employing multiple classifiers for data stream mining are efficiency and accuracy. Although the efficiency could be the most attractive reason for adopting multiple classifiers, because a data stream can always involve a huge volume of data which turns to be a nightmare for any single learner. The accuracy of MCS in handling stream data is also remarkable: especially when the concept in the data stream is subject to evolving, changing drifting [67, 70]. Like many partitioning-based or scale-up learning algorithms (e.g., Bagging, Boosting and Meta learning) have demonstrated, by partitioning the whole dataset into subsets, the system efficiency can be dramatically improved, with a limited sacrifice of the accuracy [71-73].

When adopting MCS in stream data, the most intuitive (and probably also the simplest) scheme is simple voting which is also called Select All Majority (SAM) where the prediction from each base classifier is equally weighted to vote for the final prediction. Although simple, SAM has been proved to be effective to integrate multiple classifiers, and many revised versions [67, 68, 70] have been successfully developed to handle data streams. In comparison with CC based schemes, the CS schemes select one classifier for the final decision, where two kinds of techniques, SCS and DCS, are usually adopted. Among all SCS schemes, the most intuitive one is Cross-Validation Majority (CVM). In CVM, cross-validation is adopted and the base classifier with the highest classification accuracy from the cross-validation is selected to classify all test instances.

For dynamic classifier selection, a method of partitioning the input samples is required. For example, partitions can be defined by the set of individual classifier decisions [74], according to which classifiers agree with each other [56], or even by
features of the input samples. Then, the "best" classifier for each partition is determined using training or validation data. For classification, an unknown sample is assigned to a partition, and the output of the best classifier for that partition is used to make the final decision.

In comparison with SCS where the "best" classifier has been selected before the testing phase, DCS dynamically selects the "best" classifier for each test instance. Among different DCS schemes, the most representative one is Dynamic Classifier Selection by Local Accuracy (DCS_{LA}) which explores a local community for each test instance to evaluate the base classifiers, where the local community is characterized as the K Nearest Neighbors (KNN) of the test instance in the evaluation set.

A referee based dynamic classifier selection scheme was proposed in [56] where referees, in the form of decision trees, partition the whole evaluation set into subsets, and each base classifier is evaluated with these subsets to explore its domain of expertise. The advantage of the Referee is that it partitions the evaluation set into subsets by joining the features of the base classifiers. The less important attributes won't be used in partitioning the evaluation set, and the partitioned subsets tend to be more reasonable in exploring the domain expertise. However, the drawbacks are threefold.

(1) To learn each referee, one has to explore the features of each base classifier.

(2) It uses the learned decision trees to partition the original training set into subsets, and the system performance will critically depend on the quality of learned decision trees.

(3) Because each classifier has its own referee, and the reliabilities from different referees are evaluated from different subsets. Without the same measurements, the selected classifier might not work well.
Two of the most popular weighting policies used in CC are Bagging [75] and Boosting [76]. The Bagging algorithm works with a training sample of fixed size. The examples are re-sampled according to a given probability distribution. The algorithm constructs many different bags of samples by performing bootstrapping iteratively. Each bag is a set of training samples and is collected by randomly and uniformly re-sampling of the original training set. The algorithm then applies a base classifier to classify each bag. Finally, it performs some type of an average of the classifications of each sample via a vote. Because the algorithm does not change the distribution of the samples, all classifiers in the Bagging algorithm have equal weights during the voting. It has been shown that if the base classifier is unstable, then Bagging can improve the classification accuracy significantly. However, if the base classifier is stable, then Bagging can reduce the classification accuracy because each classifier receives less of the training data. The Bagging algorithm reduces the variance of the classification but it has little effect on the bias of the classification. Its main advantage is that it can improve the classification accuracy significantly if the base classifier is properly selected. It is also not very sensitive to noise in the data.

The Boosting algorithm also works with a fixed training sample, but it assumes that the weak learning algorithm can receive weighted examples. In the beginning of a Boosting method, all training samples have the same weight. Then, the samples are re-weighted in such a way that the incorrectly classified samples have more weight than the correctly classified ones. Boosting algorithm has a tendency to reduce both the variance and the bias of the classification. Its main advantage is that in many cases it increases the overall accuracy of the classification. On the other hand, its major problem is that it usually does not perform well in terms of accuracies when there is noise in the data.

AdaBoost is an advanced version of Boosting which assign a weight to each training sample that determines its probability of being selected for a training set for a classifier. The weight of the sample is reduced if it is correctly classified while the weight is increased if it is misclassified by successive classifiers [77].
M. Skurichina, and Robert P. W. Duin [78] proposed a CC techniques known as Random Subspace Method (RSM). Instead of sampling the training data, RSM samples the feature space. In RSM, ‘m’ features are randomly selected from the ‘n’ dimensional feature vector and thus the m-dimensional random subspace of the original n-dimensional space is obtained (m<n). Then classifiers are generated on the training sets, which are constructed by changing each example in the original training set into an m-dimensional vector. At last, all classifiers are combined by proper combination rule. The subspace dimension is smaller than that in the original feature space, while the number of training examples remains the same. Therefore, the relative size of the training set increases.

Although there are a number of methods for constructing classifier ensembles, what is the best method for a particular application is still an open issue. For this problem, the "test and select" approach [79] is proposed as a useful trial, but far from a complete solution. In the beginning of "test and select", it is a phase of over-producing possible ensemble combinations. Then these ensembles are tested on a validation set and the best one is selected. This selected ensemble is tested on a final test set that has not been implicated in the selection. It can be seen that the "test and select" approach actually require two test sets, a validation set and a final test set, which are used to select ensemble and to test performance respectively. A separate validation set can avoid the over-estimation of ensemble performance that may occur if both selection and assessment are based on the same test set.

D. Wolpert [80] proposed a CC techniques known as Stacking. It was proposed as a general framework for combining classifiers. There are two levels of induction in stacking. The basic idea is to perform the second level induction over the outputs of classifiers, thus to correct the bias that single classifier introduces. Each classifier predicts examples in the training set and get its decision. A new training example is constructed by adopting the decision of all classifiers as attributes and pair them with the original correct answer. All new training examples compose a new training set, which is used to generate the second level stacking classifier. When an unknown
pattern is given, it passes through a two level inductions and the final decision is obtained.

Sabourin et al. [81] present an algorithm which has some similarities to DCS-LA approach. One variation is that, the algorithm selects the classifier that correctly classifies the most consecutive neighboring training samples (relative to the unknown test sample). The selected classifier is said to have the highest "rank." Although they do not associate their algorithm with the concept of local accuracy, their notion of classifier rank certainly has this flavor.

The Behavior-Knowledge Space (BKS) algorithm has been proposed in connection with an application for recognizing handwritten numerals [74]. Behavior-Knowledge Space is an N-dimensional space where each dimension corresponds to the decision of one classifier. Each classifier can assign a sample to one of M possible classes. Each unit of a BKS represents a particular intersection of individual classifier decisions. Thus, the BKS represents all possible combinations of the individual classifier decisions. Each BKS unit accumulates the number of training samples from each class. For an unknown test sample, the decisions of the individual classifiers index a unit of BKS, and the unknown sample is assigned to the class with the most training samples in that BKS unit.

Many methods for building a MCS have been proposed and applied in many application areas. Usually the number of classifiers in a MCS is designed by an ad-hoc selection. After the ad-hoc selection of the number of classifiers in the MCS, either Simple, Bagging (Bootstrap Aggregating), Boosting, AdaBoost or Cross-validation is then applied to build the MCS. The ad-hoc choices are usually either 3, 5 or 7 Classifiers [82, 83].

Xingquan Zhu, Xindong Wu, Ying Yang, (2004) proposed a selection method which dynamically selects a single "best" classifier to classify each test instance at run time. It uses statistical information from attribute values, and uses each attribute to partition the evaluation set into disjoint subsets, followed by a procedure that evaluates
the classification accuracy of each base classifier on these subsets. Given a test instance, its attribute values determine the subsets that the similar instances in the evaluation set have constructed, and the classifier with the highest classification accuracy on those subsets is selected to classify the test instance [84].

Ted E. Senator [85] constructed a Multi-Stage Classifier from a series of increasingly accurate and expensive individual classifiers, considering a variety of metrics such as accuracy, cost/benefit ratio, and lift. It suggests architectures appropriate for both independent instances and for highly linked data.

Tin Kam Ho Basu, M. [86] studied a number of measures that characterize the difficulty of a classification problem, focusing on the geometrical complexity of the class boundary. They compared a set of real-world problems to random labeling of points and found that real problems contain structures in this measurement space that are significantly different from the random sets. This can guide static and dynamic selection of classifiers for specific problems as well as sub problems formed by confinement, projection, and transformations of the feature vectors.

Kuncheva, L.I. [87] proposed a method by combining classifier selection and fusion by using statistical inference to switch between the two. Selection is applied in those regions of the feature space where one classifier strongly dominates the others from the pool and fusion is applied in the remaining regions. Decision templates method is adopted for the classifier fusion part.

Random sampling is often used to handle large datasets when building a classifier. Chan and Ytolfo [88, 89] considered partitioning the data into subsets that fit in memory and then developing classifier on each subset in parallel. The output of multiple classifiers is combined using various algorithms to reach the final classification. Their studies show that although this approach reduces running time significantly, the multiple classifiers did not achieve the accuracy of a single classifier built using all the data.
1.8 Scope of the Research Work

Generally, there are two directions in the area of pattern recognition for improving classification performance. One is to improve the classification performance of a classifier itself. The other one is to improve the performance of the MCS which consists of a set of classifiers and a decision combination method.

When improving the performance of a classifier, so many methods have been proposed in the literature. But no one has concentrated on the quality of the database especially when performing the random sampling. Learning from imperfect information sources requires significant data preprocessing efforts to enhance the underlying data quality before the model is built.

To improve the quality of the information sources, a new method known as Re-Sampling based on Threshold (RST) has been proposed and used in the research work. The efficient re-sampling technique RST checks the quality of the random sample and is used to construct most accurate classifier. The new method RST, which nicely takes care of the diversity and accuracy among base learners for effective classifier ensembling.

In the literature, many methods have been proposed for MCS for the task of classification. Most studies on combining multiple classifiers for the improvement of classification performance are made on static classifier fusion. Some of the studies are made on the selection of single classifier among several classifiers statically and also dynamically. But no one has no far combined the classifier selection and classifier fusion dynamically. A new ensemble creation method known as Dynamic Combination of Multiple Classifiers System (DCMCS) has been proposed and studied in the research work. This new method DCMCS produces more accuracy than any other combination of classifiers.

The classifiers Decision tree, K nearest neighbor and Neural Network are used to construct ensemble of classifiers. When constructing individual classifiers, the
simple random sampling has been used. The probability of occurrences of every class for the entire training data set has been estimated. Based on RST, thresholds have been fixed for the individual classes. When the data set have been selected randomly, the probabilities of the classes have been checked against the thresholds. If the random sample does not satisfy the threshold, Re-Sampling is performed. The random set of samples, which satisfies the threshold is allowed to build the classifier and this classifier participates in DCMCS.

When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. In result, poor accuracies have been obtained for unseen data for the classifier obtained using this data set. Tree pruning methods typically use statistical measures to remove the least reliable branches, generally resulting in faster classification and an improvement in the ability of the tree to correctly classify independent test data. When constructing the Decision tree classifier, the performance before pruning and after pruning have been estimated and analyzed for each set of random samples. The analysis shows that the accuracy of classifiers after pruning has produced better accuracy than the classifiers before pruning.

When considering K-Nearest Neighbor classifier, it produces different Accuracies for different value of K. Hence, the performance of classifiers for the different neighborhood size has been analyzed. Experiments have been carried out for the neighborhood size ranging from 1 to 25 in each iteration. In every iteration, the highest accuracy is identified and reported.

In applying neural network, it has been trained for different number of hidden units varying from 1 to 10 and the performance has been analyzed. Different accuracies have been produced by Neural Network using different number of hidden units for the same set of data samples. For each data set, the results relating to the number of hidden units, producing the highest accuracy have been identified.
The basic idea in DCMCS is to estimate each classifier's accuracy in local regions of feature space surrounding an unknown test sample, and then use the decision of the most locally accurate 2 classifiers out of 3 classifiers. In this implementation, "local regions" are defined in terms of the K Nearest neighbors in the training data. The accuracy of the classifier for the local region is estimated by simply calculating the percentage of training samples in the region that are correctly classified. Determining the appropriate size for a local region is a part of designing the DCMCS approach. The experiments have been carried out for various region sizes ranging from K = 1 to K = 25 using Euclidean distance metric. In general, choosing \[\lceil \frac{n}{2} \rceil\] best performance classifiers out of n classifiers dynamically, would result in better performance.

1.9 Organization of the thesis

Introduction to Data mining, preprocessing techniques, steps followed in KDD process, different Classification methods, measuring the performance of classifier in different methods and Literature Review has been dealt in chapter 1.

In chapter 2, an enhanced new approach to Re-sampling technique known as RST has been defined. The database which is considered for the analysis is the Adult database and the Earthquake databases. For both the databases, the statistical details, their domain and transformations made on these databases are discussed.

Chapter 3 discusses about the algorithm which is used to construct the decision tree classifier. It discusses the pruning techniques and also the potential and drawbacks of decision tree classifier. The performance analyses of Decision tree classifier before pruning and after pruning are made in this chapter.

Chapter 4 discusses the K Nearest Neighbor Classifier algorithm. The best choice of K depends upon the data. Hence the performance of KNN classifier for the neighborhood size varying from 1 to 25 has been analyzed in this chapter. In every iteration, the highest accuracy is identified and reported.
In Chapter 5, a study is made on the Back Propagation network. There is no clear rule for the "best" number of hidden layers. Usually the network design is a trial-and-error process and may affect the accuracy of the resulting training network. Hence in this chapter the network has been trained using different number of hidden units varying from 1 to 10 and the performance has been analyzed. For each data set, the results relating to the number of hidden units which produces the highest accuracy has been reported.

Chapter 6 discusses the multiple classifier system. It introduces a new method known as DCMCS and discusses the algorithm for DCMCS. Comparison of the performance of DCMCS with the best individual classifier and also with the static classifier combination are studied and discussed. The experimental results and comparative studies made prove that the performance of proposed DCMCS outperforms any other combinations of classifiers.

Chapter 7 concludes the review of work done and provides scope for the future enhancement of this thesis work. It also compares the time complexity of various classifiers. It concludes that the implementation of Re-Sampling technique RST for the Dynamic Combination of Multiple Classifiers System is efficient.