Local sparsity coefficient-based mining of outliers.

Malik. Agyemang

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/2632

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
LOCAL SPARSITY COEFFICIENT-BASED MINING OF OUTLIERS

by

Malik Agyemang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfilment of the Requirements for the
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2002
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-75813-3
ABSTRACT

The process of discovering interesting, useful and previously unknown knowledge from very large databases is known as data mining. Outlier or exception mining focuses on the problem of finding patterns that apply to a small percentage of data objects. Outliers are observations that show different characteristics from all other data objects to arouse suspicion that they were generated by a different mechanism. Density-based algorithms for mining outliers are the most effective in finding all forms of outliers. Density-based algorithms determine outliers based on the concentration of data objects at a location and declare objects with few neighbours as outliers. However, existing density-based algorithms have the following drawbacks: (1) computing the local reachability distance and density for every object before the few outliers are found; (2) computing local outlier factor (LOF) for every object in the dataset before declaring those with very high LOF as outliers. These are very expensive computations since outliers form only a small fraction of the entire data.

This thesis proposes Local Sparsity Coefficient (LSC) and Enhanced Local Sparsity Coefficient (ELSC) algorithms based on the distance of an object and those of its k-nearest neighbours without computing reachability distance and density of every object. This reduces the number of computations and comparisons in LOF technique. In ELSC, data objects that can not possibly contain outliers are pruned (removed) based on their neighbourhood distances. The remaining set constitutes the candidate set on which outliers are determined resulting in an improved performance over LSC and LOF.

[Keywords: Data mining, sparsity coefficient, k-distance of p, k-distance neighbourhood, local outlier factor, discordancy test, exact exception, dissimilarity function, smoothing factor, outliers, partition-based, reachability distance]
To my late mother, (Mariam Pinamang)
ACKNOWLEDGEMENTS

I am very grateful to my supervisor, Dr Christie Ezeife for her guidance and direction throughout this thesis. I thank her for the research assistantships I received during the program, without which it would have been very difficult getting to this stage and above all the opportunity she gave me to study for this Masters degree. I wish to thank Drs Asish Mukhopadhyay, Myron Hlynka and Arunita Jaekel for serving on my thesis committee. I also wish to thank my brothers and their lovely wives for their emotional and financial contributions to my study.

My special smile goes to my dad, Mr. Eassah Agyemang for being such a wonderful father and my step-mother, Madina for taking very good care of my dad, my siblings Mohammed, Adam, Ayesha, Ahmed, Mohammed, Ibrahim, Fati, Suleman, Mariam, Yusif, Hawa, Halima, Sakina and Omar for their invaluable support. I am really proud of you and our entire family. I would like to thank Mrs Agyemang (Nuzrat-Emelia) for the immense role she played prior to the commencement of my program; your contribution is highly appreciated and deeply remembered. I appreciate the company of my friends (especially Ajumobi Udechukwu, Cosmas Anomah-Yeboah, Philip Awere-Gambrah, Timothy Ohaneku, Joseph Tenkorang) and all those who made my stay in Windsor lively.

Finally, all praise belongs to Allah, who has given me life, strength, knowledge and made everything possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Data Mining</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Association Rules</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Classification Rules</td>
<td>5</td>
</tr>
<tr>
<td>1.1.3 Clustering</td>
<td>7</td>
</tr>
<tr>
<td>1.2.4 Mining Outliers/Exceptions</td>
<td>9</td>
</tr>
<tr>
<td>1.2.4.1 Application of Outlier Mining</td>
<td>10</td>
</tr>
<tr>
<td>1.2 The Motivation of Thesis</td>
<td>11</td>
</tr>
<tr>
<td>1.3 The Thesis Contributions</td>
<td>13</td>
</tr>
<tr>
<td>1.4 The Thesis Outline</td>
<td>13</td>
</tr>
<tr>
<td>2 PREVIOUS/RELATED WORKS</td>
<td>14</td>
</tr>
<tr>
<td>2.1 Distribution-Based Outlier Mining</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Depth-Based Outlier Mining</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Distance-Based Outlier Mining</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1 The Exact Exception</td>
<td>21</td>
</tr>
<tr>
<td>2.3.2 The Unified Notion of Outliers</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3 The Partition-based Approach</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Density-based Outlier Methods</td>
<td>27</td>
</tr>
<tr>
<td>2.4.1 The Local Outlier Factor (LOF)</td>
<td>28</td>
</tr>
<tr>
<td>3 LOCAL SPARSITY COEFFICIENT ALGORITHM</td>
<td>38</td>
</tr>
<tr>
<td>3.1 Analysis of problems in LOF</td>
<td>38</td>
</tr>
</tbody>
</table>
3.2 Local Sparsity Coefficient 43
3.3 Enhanced Local Sparsity Coefficient Algorithm 47
3.4 Defining Bounds on $LSC_k$ and $Lsr_k$ 51
3.5 Comparing LSC and ELSC with LOF 52

4 EXPERIMENTAL EVALUATION 54
4.1 Experiment for Accuracy 54
4.2 Experiments for Response Time 57

5 CONCLUSIONS AND FUTURE WORK 62

REFERENCES 64
VITA AUCTORIS 68
LIST OF TABLES

1. Customer Purchases Data .......................... 3
2. The Exact Exception Using I={4, 10, 11, 14} .......... 23
3a. Generating Partitions .......................... 26
3b. Computing Bounds .......................... 27
3c. Computing Outliers .......................... 27
4. Results from LOF Computation .................. 36
5. Sample Data From NHL 1995 Player Statistics ....... 38
6. Computed Distances .......................... 39
7. LOF Results From NHL Data ................... 42
8. Results of Applying LSC to NHL Data .......... 47
9. Experimental Results on Test Data with MinPts = 50 55
10. Experimental Results on Test Data with MinPts = 100 56
11. Experimental Results on Test Data with MinPts = 200 56
12. Effect of Neighborhood Distance on response Time 58
13. Effect of Data Size on Response Time with MinPts= 50 59
14. Effect of Data Size on Response Time with MinPts = 500 60
# LIST OF FIGURES

1. Decision Tree for RiskInfo ........................................ 7
2. Salary versus Years of Experience ................................. 8
3. Box Plot of Marks Scored ........................................... 16
4. The Exact Exception Algorithm .................................... 22
5. 2-D Dataset ................................................................ 29
6. $K$-distance($P$) and Neighbours of $P$ ......................... 31
7. Local Sparsity Coefficient Algorithm ............................ 45
8. Enhanced Local Sparsity Algorithm ............................... 48
9. Effect of Neighborhood Distance on response Time ......... 58
10. Effect of Data Size on Response Time with MinPts = 50 ....... 60
11. Effect of Data Size on Response Time with MinPts = 500 ...... 61
1 INTRODUCTION

The amount of data collected and stored in databases increases tremendously everyday as a result of affordable database technology, competition among organisations, affordable hardware components, and the proliferation of the internet. The collected data need to be efficiently and effectively analysed to expose the useful patterns hidden in them for decision making. The process of finding useful information from data is a very complex task known as data mining. Data mining is described as the non-trivial process of identifying valid, potentially useful and ultimately understandable patterns in data [AIS93a, FPS96]. In the past, mining tasks have concentrated on finding frequent patterns (patterns that apply to majority of data objects) while discarding the less frequent ones. The less frequent patterns which are usually eliminated also contain another group of objects often described as nuisance, noise or outliers. Outliers are observations that deviate so much from other observations within the same group to arouse suspicion that they were generated by a different mechanism. In many applications, identifying exceptions or rare events can often lead to the discovery of interesting and unexpected knowledge as far as data mining is concerned.

There are algorithms in statistics, machine learning and data mining that have considered outliers, but only to the extent of tolerating outliers in whatever problem those algorithms are solving [FPS96, Jus01, NH94, Van01, ZRL96]. There are existing algorithms that specifically identify outliers and remove them before any further processing is done on the data [ARS01, Lop97]. ROR (rejection of outliers by rotation) is an algorithm developed by Adam et al. [ARS01], which rejects outliers in image processing (matching of points between two perspective views) with a potential of rejecting up to 85 percent of the entire data if they are really outliers even though in reality outliers might not be up to 10 percent of the entire data. This shows the aggression with which outliers are tackled among different data mining tasks. In this thesis, we concentrate on finding a subset of the less frequent patterns called outliers which have not gained much attention like the other areas of data mining. Mining outliers can lead to the discovery of unexpected
knowledge in areas such as electronic commerce, credit card fraud, phone cloning and monitoring of suspicious activities.

1.1 Data Mining
The major problem facing management in this ‘data rich’ world is how to analyze the abundant data collected and stored in databases and “legacy systems,” such that interesting hidden patterns can be revealed for decision making. Making a timely and accurate decision is very important and crucial for the survival of organizations against their major competitors. The process of finding useful and interesting information from data is a very complex task known as data mining. Fayyad et al. [FPS96] describe data mining as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns from data. Data mining is the entire process of extracting ‘gold’ from large amounts of data, and subsequent analysis of the obtained information for decision-making. ‘Gold’ is a useful pattern or piece of information that is hidden in databases and it needs to be mined using specific algorithms and techniques.

Data mining usually acts as the front-end to most databases and data warehouses for decision support systems. For example, association rule mining is employed by merchandise stores to arrange products that are bought together in the same area to attract the attention of customers with the view of making more sales and eventually maximising profit. There are different varieties of mining tasks which are handled with different algorithms under different conditions. Data mining tasks are classified into four different categories; they are association-rule mining, classification techniques, clustering, and exception/outlier detection [AIS93a, FPS96, KN98, SAM96, ZE01]. A brief description of each of the mining tasks is given in the subsequent sections even though the primary concern of this thesis is exception/outlier detection.

1.1.1 Association-Rule Mining
A set of items is called an itemset. A set of items with k-elements is called a k-itemset. For instance, the itemset \{pen, ink\} is a 2-itemset set whereas the itemset \{pen, ink, milk, juice\} is a 4-itemset. A market basket refers to transactions that take place in one
shopping trip. For instance, all items purchased by a shopper in a day could represent a market basket. Table 1 represents customer purchase data used to explain some of the terms connected with association-rule mining in market basket analysis. A transaction represents all items bought during a shopping trip by a customer. Thus, transaction ID 1 indicates that items pen, ink and milk are purchased whereas transaction ID 2 indicates the items pen and milk are purchased in the same shopping trip.

An association rule is an implication of the form \( \{A_1, A_2...A_n\} \Rightarrow \{B_1, B_2...B_m\} \), where \( A_i \) and \( B_j \) are items in a market basket. The rule is interpreted as “when items \( A_1, A_2...A_n \) are purchased then the items \( B_1, B_2...B_m \) are also purchased in the same market basket” [AIS93b]. The itemset to the left of the implication sign is called antecedent whereas the one to the right is called consequent (i.e., \( \{A_1, A_2...A_n\} \Rightarrow \{B_1, B_2...B_m\} \)). Thus, an association rule could be represented as antecedent \( \Rightarrow \) consequent. Generally, if \( I = \{I_1, I_2...I_n\} \) is a set of items in a market basket and \( D \) is a set of transactions where each transaction \( T \) is a set of items such that \( T \subseteq I \), then an association rule is an implication of the form \( X \Rightarrow Y \), where \( X \subseteq I \), \( Y \subseteq I \), \( X \cap Y = \emptyset \). Table 1 represents customer purchases data used to explain the terms connected with association-rule mining.

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pen, ink, milk</td>
</tr>
<tr>
<td>2</td>
<td>Pen, milk</td>
</tr>
<tr>
<td>3</td>
<td>Pen, ink, juice</td>
</tr>
<tr>
<td>4</td>
<td>Pen, ink, milk, juice</td>
</tr>
</tbody>
</table>

Table 1: Customer Purchases Data

Using Table 1, the set of items in the market basket \( I = \{pen, ink, milk, juice\} \). The itemset \( X = \{pen, milk\} \) is a subset of items in \( I \) (i.e., \( \{pen, milk\} \subseteq \{pen, ink, milk, juice\} \)) and the itemset \( Y = \{ink\} \) is also a subset of \( I \) (i.e., \( \{ink\} \subseteq \{pen, ink, milk, juice\} \)). The
itemsets \{pen, milk\} and \{ink\} are contained in \{pen, ink, milk, juice\}. Hence, the rule \{pen, milk\} \Rightarrow \{ink\} is a valid association rule in the above domain. Association-rule mining is the process of finding interesting association or correlation relationships among large sets of data. It is used to formulate rules about items that frequently occur together. The evaluation of an association rule depends on the support and confidence of the rule. Support is used to determine how often the rule applies whereas confidence is used to determine how often the rule is correct [AIS93b, ZE01].

The support for a set of items is the percentage of transactions that contain all of these items. The rule \(X \Rightarrow Y\) has support \(s\) in transaction set \(D\) if \(s\%\) of transactions in \(D\) contains \(X \cup Y\), i.e., \(s\% = (\text{transactions with } |X \cup Y|/|D|) \times 100\%\), where \(|D|\) is the number of transactions in \(D\). For instance, using the transaction set given in Table 1, if the set of items \(X = \{\text{pen, milk}\}\), \(Y = \{\text{ink}\}\) then the support of the rule \(X \Rightarrow Y\), i.e., \(\{\text{pen, milk}\} \Rightarrow \{\text{ink}\}\) = \((2/4) \times 100\% = 50\%\), since the item set \(\{\text{pen, ink, milk}\}\) occurs in 2 (transactions 2 and 4) out of the 4 transactions.

The confidence of a rule \(X \Rightarrow Y\) is the percentage of transactions containing \(X\) that contain both \(X\) and \(Y\). The rule \(X \Rightarrow Y\) holds in the transaction set \(D\) with confidence \(c\) if \(c\%\) of transactions in \(D\) that contain \(X\) also contain \(Y\). Thus, the confidence \(c\% = (|X \cup Y|/|X|) \times 100\%\). For example, using the transaction data in Table 1, \(X = \{\text{pen, milk}\}\) occurs in transactions 1, 2 and 4. Hence, the confidence of the rule \(\{\text{pen, milk}\} \Rightarrow \{\text{ink}\}\) = \((2/3) \times 100 = 75\%\).

Minimum support (minsupport) and minimum confidence (minconfidence) are thresholds assigned by the user before doing the association-rule mining. These thresholds are chosen by an expert based on the characteristics of the data observed over a long period of time. The minsupport and minconfidence are used to discard less-frequent itemsets using the Apriori algorithm [AIS93a, ZE01]. A large itemset is an itemset with support greater than or equal to the minimum support (minsupport). Large itemsets are also called frequent itemsets. For example, given that the minsupport for the transactions in Table 1

4
is 3, the itemsets \( \{pen\}, \{ink\} \) and \( \{milk\} \) are large itemsets since each of them occurs in either three or more of the transactions.

Association-rule mining is the process of finding interesting association or correlation relationships among a large set of data. The problem of mining association rules is that of discovering all associations and correlations among data item(s) where the presence of one set of item(s) in a transaction implies (with certain degree of confidence) the presence of the other item(s). Since such transaction-based databases contain extremely large amounts of data, rules are pruned according to the support and confidence for items under consideration [AIS93b]. The study of association-rule mining has attracted the attention of large number of researchers to propose different algorithms for mining association rules [BMS97, FU94, MTV94, SA95, ZE01].

1.1.2 Classification-Rule Mining

Classification-rule mining is a knowledge acquisition and extraction tool used in data mining. The basic aim of classification is the building of models on the basis of the characteristic of the existing data. These models are used to classify the behaviour of incoming data items into classes [AG1+92, SAM96]. For instance, an insurance company may classify its customers into high-risk or low-risk according to a schema such as: RiskInfo (age: integer, cart-type: string, risk: boolean). The database has information on current customers where each record has a customer’s age, type of car and a flag indicating whether a customer is considered high-risk or low-risk. This information is used to identify rules that predict the risk level of new customers whose ages and car types are known. The rule “if age is between 16 and 30 and car-type is Sport or Truck, then high-risk” is one of many rules that can be deduced from previous data captured in the database.

Classification rules usually have specific structure consisting of an attribute with a value to be determined, known as the dependent attribute and a set of attributes that determine the value of the dependent attribute called the predictor attributes. For instance, in the RiskInfo schema, age and car-type are the predictor attributes since they determine the
value of the dependent attribute risk. Given a dependent attribute $Y$ and a set of predictor attributes $X_i$, classification rule is of the form $P_1(X_1) \land P_2(X_2) \land \ldots \land P_k(X_k) \Rightarrow Y$, where $P_1, P_2, P_3 \ldots P_k$ are constants. For instance, the rule "if the age of a person is between 16 and 30 and car-type is Sport or Truck, then high-risk is true" is represented as:

$$(16 \leq \text{age} \leq 30) \land (\text{car-type} \in \{\text{Sport, Truck}\}) \Rightarrow \{\text{high-risk = true}\}.$$ Classification rules are referred to as regression rules when the dependent attribute is numeric. This is possible only when all the predictor attributes are numeric. For instance, given that the amount of timber exported from Canada in a year is a function of the amount of rainfall in the previous year and the world market price, we can establish a regression rule using the given information since all attributes are numeric.

Classification and regression-rule mining involves the discovery of hidden common patterns among data items and their classification according to these patterns. It allows for the development of profiles for items belonging to a particular group according to their common attributes. The profile can then be used to classify new data items that are added to the database in the future [SAM96]. Decision-tree classifiers are very popular among classification-rule mining since they are very simple to build, easy to understand and their interpretation is very easy as well.

The mining process usually consists of building the tree and later deducing rules out of the built tree. The tree building phase involves partitioning the data using the attributes of the database schema. For instance, given our database schema, the data are partitioned with age $>30$ and age $\leq 30$. The value of the dependent attribute is determined for each partition from previous data captured in the database. Each branch of the tree is further partitioned using the predictor attribute, car-type. The corresponding value of the dependent attribute for each branch is determined from the previous data. The partitioning continues until a dependent attribute value is obtained for each terminal node. The resulting classification tree is shown in Figure 1. A number of algorithms have been proposed for constructing and pruning decision trees, notable among them is SPRINT [SAM96]. SPRINT is one of the first most-scalable algorithms for classification.
It maintains a separate attribute list for both categorical and numeric data. It also maintains a sorted list of numeric attributes. The other classification algorithms are PUBLIC [RS98] and BOAT [GGR+99].

A sample classification tree resulting from an insurance database with the schema: RiskInfo (age: integer, car-type: string, risk: boolean) is shown in Figure 1.

![Decision Tree for RiskInfo](image)

Figure 1: Decision Tree for RiskInfo

The final phase of the mining process involves deducing the set of possible rules from the constructed tree. The three possible rules deduced from the tree of Figure 1 are: “if a person is aged 30 or younger and drives sport or truck, then he is likely to have high risk” represented as \((16 \leq \text{age} \leq 30) \land (\text{car-type} \in \{\text{Sport, Truck}\}) \Rightarrow \{\text{high-risk} = \text{true}\}\). The next rule is “if a person is aged above 30, then he is likely to have low risk irrespective of the type of car he drives” represented as: \((\text{age} \geq 30) \Rightarrow \{\text{low-risk} = \text{true}\}\). Finally, “If a person is aged 30 or younger and drives a van, then he is likely to have low insurance risk” represented as; \((16 \leq \text{age} \leq 30) \land (\text{car-type} \in \{\text{Van}\}) \Rightarrow \{\text{low-risk} = \text{true}\}\)

1.1.3 Cluster Mining
Clustering is the process of partitioning records into groups such that records within groups are similar to each other whereas those that belong to different groups are dissimilar. Similarity between records is measured by a distance function which takes two inputs and returns a value that is a measure of their similarity [NH94]. For instance,
Figure 2 shows the salary of employees and their level of experience. The first cluster shows employees with low experience and receiving low salary illustrated as cluster A. The second cluster represents employees with low experience but receiving high salary illustrated as cluster B. Finally, cluster C represents employees with high experience and receiving high salary. It should be noted that within a particular cluster there is not much difference between salaries of members within that cluster. The members within the first cluster labelled A receive between $20,000 and $25,000. In the last cluster labelled C, members receive above $70,000.

![Salary Versus Years of Experience](image)

Figure 2: Salary Versus Years of Experience

The goal of clustering is to partition a set of records into groups such that records within groups are homogeneous. Homogeneity within clusters is an important feature of all clustering algorithms. The output of a clustering process is a summarized representation of each cluster, which depends largely on the shape of the clusters as well. Other clustering algorithms such as DB-SCAN [EKS+95], BIRCH [ZRL96] are also used for partitioning data. Clustering algorithms are generally classified into two groups, hierarchical and partitional. Hierarchical clustering involves a sequence of partitions in which each partition is nested into the next partition whereas partitional clustering
algorithms attempt to determine the k partitions that minimize the distance function [JD98].

In CLARANS [NH94], a clustering algorithm for large databases based on randomized search was proposed. The most-centrally-located point in each cluster called the medoid was used to represent each cluster with the view of finding the k best medoids that maximize the criterion function. The problem was reduced into a graph by representing each of the k medoids as a node in the graph. A number of iterations were performed and in each case, a current node is chosen if it results in better clustering. Zhang et al. [ZRL96] proposed the algorithm BIRCH, which pre-clusters the entire data into two clusters, maximum possible and finest possible clusters both of which can fit into main memory. BIRCH employs a CF-tree (a balanced tree structure) to store the summaries. The CF-tree is traversed to find the closest cluster for each point and each of the points within a certain distance k of the medoid is absorbed into the cluster. Otherwise, they start a new cluster.

1.1.4 Mining Outliers / Exceptions

The forth category of mining deals with outliers/exceptions. An outlier is a very complex term since currently there is no clear definition for it. The description given by Hawkins [Haw80] is almost becoming accepted as a general definition for an outlier: “An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism”. In other words, an outlier is an observation that has different characteristics compared to the other members of the same group. Different researchers have given different descriptions of what an outlier is as well as proposing different ways for finding all such outliers [BKN+00, KN97, TN98].

Knorr et al. [KN97, KN98] described an outlier in a dataset O as a fraction p of the observations in O that lies at a distance greater than D (D is a specified distance) from O. Ramaswamy et al. [RRS00] however, described an outlier as an observation that has the greatest distance from all other observations when the calculated distances are ranked. In Breunig et al. [BKN00], an outlier is viewed as an observation with the highest local
outlier factor within a given neighbourhood. The local outlier factor is a measure of the
distribution or density of objects around a particular point. A high local outlier factor
means that the neighbours are far away, whereas a low local outlier factor indicates
nearer neighbours.

Outlier-mining is the process of designing models and algorithms that can detect rare
records that constitute only a small percentage of the entire data which are sometimes
ignored as noise and tolerated in other mining activities. Outliers by their definition are
very difficult to locate and as such have prompted different researchers to use different
mechanisms for mining them. Outlier-mining techniques are grouped into four different
categories, namely distribution-based, depth-based, distance-based and density-based
outlier-mining approaches.

1.1.4.1 Applications of Outlier-Mining
Outlier-mining algorithms can be applied in almost every sphere of life where data are
collected including science, medicine, business, banking, finance and education.

1. Outlier-mining algorithms were applied to National Hockey League (NHL) data
and it revealed interesting characteristics which were buried in those data [KN98].
Outlier-mining techniques would be most appropriate when one is interested in
finding the most outstanding player that takes into account all attributes and not
just scoring abilities or shots on goals.

2. In banking and finance, outlier-mining can be used to monitor suspicious
drawings from bank accounts. It is also used in credit card fraud and cellular
phone fraud detection popularly known as cloning. Outlier-mining algorithms can
also be used to analyze the financial data of an establishment to detect
outstanding performance.

3. In education, it can be used to establish the performance level of both educators
as well as students. It can also be used to determine why certain outliers occur
[KN99].

4. The basic aim of entrepreneurs is to maximize profit; outlier-mining techniques
can be used to achieve this goal by analyzing the business data to know which
merchandise has high sales and profit. It can also be used to determine the worst merchandise with respect to sales and profit regeneration.

5. In medicine, outlier-mining can be used to identify patients who respond to treatments exceptionally. It can also be used to explain the behaviour and patterns of many medical conditions such as cancer etc.

1.2 Motivation for Thesis

The majority of work done on outliers is in statistics where more than a hundred discordancy tests have been developed for finding different types of outliers [BL94, Haw80]. Discordancy tests are statistical methods designed for finding outliers under different conditions. The problem with the statistical approach is that most of the statistical tests developed for finding outliers (discordancy tests) are for single variable distributions (for example, finding outliers in rainfall pattern of Windsor in 1998). This factor makes the statistical approach less practically useful in this ‘data rich’ world, where data has many dimensions (attributes). Next, the underlying statistical distribution of the user’s data should also be known before any discordancy test can be applied. Depth contours were introduced to solve the problem of distribution-based method [Tuk75]. ISODEPTH [JKN98] and Fast Depth Contours (FDC) are algorithms based on depth contours designed for finding outliers irrespective of the underlying statistical distribution of the data of interest but they were found to be computationally infeasible when the number of variables to be examined exceeds three [RR96].

The distance-based outlier-mining concept proposed by Knorr et al. [KN97, KN98] addresses the problems found in depth-based and distribution-based algorithms. The Nested-Loop (NL) and the Index-Based algorithms proposed for mining all distance-based outliers have a time complexity of \( O(kN^2) \), which is linear in the number of dimensions \( k \) but quadratic in \( N \) (dataset size) making them highly unfavourable when the data size is large. The distance-based outlier-mining concept has two major drawbacks. First, it requires the user to specify the minimum acceptable distance for an outlier \( D \) based on which outliers are declared. The problem is that \( D \) should be chosen very carefully because if \( D \) is too small, most of the data objects become outliers whereas
none of the data objects become outliers if D is too large. Secondly, the method does not provide ranking for the outliers making it very difficult to distinguish between strong and weak outliers. The definition of an outlier proposed by Ramaswamy et al. [RRS00] together with the proposed Partition-Based outlier detection algorithm solves the problems of the earlier ones proposed by Knorr et al. [KN97, KN98].

Breunig et al. [BKN00] argue that the problem with the earlier algorithms for mining outliers is that they take a global view of the entire dataset, thereby resulting in “global” outliers. However, in a multidimensional dataset with a more complex structure, there are some kinds of data objects that might be outliers with respect to their neighbourhood, but will never be outliers in the entire dataset. These are called “local” outliers. An algorithm for finding all local outliers called LOF was proposed. LOF computes the local outlier factor value (LOF value) for each object in the database and declare those with high LOF values as outliers.

One of the major problems of data mining algorithms is the issue of scalability. The issue becomes more crucial since outliers constitute a minute percentage of the data and yet the entire data set has to be processed before outliers can be trapped. The focus of this thesis is density-based outlier-mining techniques, which appears to be more promising in finding outliers. Although all the various algorithms proposed for different mining approaches are good in their own rights, density-based methods are more efficient and effective in detecting all forms of outliers. However, existing density-based algorithms have the following drawbacks: (1) they compute local reachability distance and density for every object before the few outliers are detected; (2) they compute local outlier factor (LOF) for every object in the dataset before declaring those with very high LOF as outliers. These are very expensive computations that must be avoided since outliers form only a small fraction of the entire data. This thesis proposes Local Sparsity Coefficient (LSC) and Enhanced Local Sparsity Coefficient (ELSC) algorithms based on the distance of an object and those of its k-nearest neighbours without computing the reachability distance and density of every object. This reduces the number of computations and comparisons in LOF technique. In ELSC, data objects that can not possibly contain
outliers are pruned (removed) on the basis of their neighbourhood distances. The remaining data constitute the candidate set on which outliers are determined resulting in an improved performance over LSC and LOF.

1.3 The Contributions of Thesis

The contributions of this thesis are that we propose two algorithms for mining outliers based on the distances of objects from their nearest neighbours without actually computing their local reachability distances and densities thereby making our algorithm less expensive and more efficient compared to local outlier factor algorithm proposed in [BKN00]. Since outliers form only a small percentage of the entire data, our second algorithm prunes the data objects based on their neighbourhood distances. Data objects that cannot possibly be outliers are pruned as soon as they are determined. This process further reduces the number of possible computations, resulting in an increased efficiency and scalability. The proposed algorithms are faster, less expensive and more scalable compared to related outlier-mining algorithms discussed in the literature.

1.4 The Outline of Thesis

The rest of the thesis is organized as follows: In Chapter 2, we present a review of the existing work on outlier-mining. A detailed description of Local Sparsity Coefficient (LSC) and Enhanced Local Sparsity Coefficient (ELSC) algorithms is presented in Chapter 3. Chapter 4 takes care of implementation and testing while Chapter 5 presents conclusions and discussions on future work.
2. PREVIOUS/RELATED WORK

This chapter reviews some of the previous and current work done on outlier-mining. The review is conducted along the four major types of outlier-mining techniques, namely distribution-based, depth-based, distance-based, and density-based outlier-mining methods.

2.1 Distribution-Based Outlier-Mining Approach

Distribution-based outlier-mining techniques are mostly found in statistics where the concept of outliers originated. The distribution-based methods make use of known standard statistical distributions (e.g. Normal, Poisson, Student t etc) to fit data points. Outliers are observations that show different characteristics from all other observations to arouse suspicion that they were taken from a different statistical distribution. The literature on outliers revealed more than hundred discordancy tests developed for distribution-based outliers. Discordancy tests are statistical tests designed for finding outliers under different conditions. The wide range of tests results from the fact that there are many statistical distributions with different underlying assumptions and applications.

The type of discordancy test to be applied to a particular data set depends on a number of scenarios and assumptions. First, assume the statistical distribution of the underlying data is known. In such a case the data to be tested should follow one of the standard statistical distributions (e.g., Normal, Poisson etc.). The next scenario is whether or not the distribution parameters are known (e.g., the mean and variance of the distribution). The expected number of outliers is also very important. Different discordancy tests are applied depending on the expected number of outliers. For instance, tests for finding single outliers are different from those for finding multiple outliers or a pair of outliers. Last but not the least is the expected type of outliers. For instance, given the mean and variance of a normal distribution, there are different tests for finding single upper outlier, upper outlier pair, k-upper outliers, single lower outlier, lower outlier pair, k-lower outliers etc. [BL94]. In most cases, visual plots such as box plot or scatter diagrams are used to determine which of the data points are potential outliers before an appropriate test
is conducted. Statistical conclusions are not drawn with respect to the outcome of the visual plots. They simply give an idea of which of the input points can possibly be outliers.

2.1.1 The Box Plot
This is a very simple visual plot that gives an idea of the data points likely to be outliers. The box is constructed using lower quartile ($Q_1$), median ($Q_2$), the upper quartile ($Q_3$), and the inter-quartile range $IQR = Q_3 - Q_1$. The main principle of the box plot is that, data points that are outliers will lie outside the region ($Q_1 - (1.5*IQR)$ and ($Q_3 - (1.5*IQR)$) whereas the non-outliers lie inside the region [BL94]. The terms associated with the box plot are explained below:

1. The median is the 50th percentile (data object that lies exactly in the middle) when data points are arranged in either increasing or decreasing order.
2. The lower quartile is the 25th percentile (data object that lies at the 25% position when data are arranged in increasing order)
3. The upper quartile is the 75th percentile (data object that lies at the 75% position when data are arranged in an increasing order).
4. The inter-quartile range is the difference between the upper and lower quartiles ($IQR = Q_3 - Q_1$).

The construction of a box plot is demonstrated using the marks scored by 25 students in an examination given as: {29, 36, 45, 55, 59, 60, 61, 64, 65, 70, 70, 71, 72, 72, 72, 72, 73, 73, 75, 76, 76, 78, 79, 98, 99}. The box plot will also be used to determine which of the marks are likely to be outliers. The marks have to be sorted in order to determine the lower, middle (median) and the upper quartiles. Using the definition of the quartiles, the lower, median and the upper quartiles are obtained as 61, 72 and 75 respectively. The inter-quartile range $(IQR) = Q_3 - Q_1 = 75 - 61 = 14$. The lower outlier cut-off point (Lower) is the point below which all objects are suspected outliers. It is given by: $Lower = Q_1 - (1.5*IQR)$. $Lower = 61 - (1.5 * 14) = 40$. The upper outlier cut-off point (Upper) is the point above which all objects in the given dataset are suspected outliers.
Upper = Q_3 + (1.5 \times IQR) = 75 + (1.5 \times 14) = 96. The corresponding box plot for the marks scored by the 25 students in an examination is shown in Figure 3.

![Box Plot Diagram]

Figure 3: Box Plot of Marks Scored

The lower and upper outlier cut-off points from Figure 3 are 40 and 96 respectively. This means that all marks that are below 40 are suspected outliers. Similarly, marks above 96 are suspected outliers. They are suspected outliers because they have to be tested using an appropriate discordancy test to verify if they are really outliers. The marks \{29, 36, 98, 99\} fall outside the outlier cut-off points as shown by the box plot in Figure 3. The type of discordancy test employed depends on a number of factors such as the type of the underlying distribution and whether the distribution parameters are known. The marks \{29, 36, 98, 99\} are tested using the z-score test. The z-score test assumes the underlying statistical distribution is normal. The test will determine if \{29, 36, 98, 99\} are really different from the rest of the marks scored. If they are different then they match our definition of outliers otherwise they are not outliers.
2.1.2 The Z-Score Test

The suspected outliers are labelled and tested using the z-score method. The z-score test assumes that the non-outliers come from a normal distribution. The suspected outliers would be confirmed outliers if they are determined to be from a different distribution. For each data point \(x_i\) suspected to be an outlier from \(n\) data points, the \(z_i\) value is computed with test statistic \(z_i = \frac{(x_i - \bar{x})}{s}\), where \(s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}\), \(\bar{x}\) and \(n\) are the mean and data size respectively. The test heuristics states that an observation with \(z_i > z\) (\(z\) specified by the analyst) is an outlier [BL94]. The z-score is obtained using the mean and standard deviation computed from the data containing the outliers. This method is not reliable in labelling outliers since both the mean and the standard deviation are affected by the outliers. The effect of the outliers on the sample variance is reduced by using the modified z-score.

2.1.2.1 The Modified Z-Score (\(Z_{\text{mi}}\))

The problem with the original z-score method is that the sample mean and the standard deviation are both affected by the values of the outliers, making it unreliable for labelling outliers. A reliable method is obtained by modifying the z-score using a more resistant estimator called the median of the absolute deviation about the median (MAD). The median of the absolute deviation about the median (MAD) is computed and substituted for the standard deviation. \(MAD = \text{median}\{\left| x_i - \text{median} \right| \}\). The computed MAD from the sample data with a median of 72 is 6 computed using the formula given above. The modified test statistic \(Z_{\text{mi}} = \frac{|x_i - \bar{x}|}{MAD}\), where \(\bar{x}\) is the mean of the sample data. It should be noted that MAD only replaces the standard deviation of the normal z-score test. The test states that any observation \((x_i)\) with a \(Z_{\text{mi}} > 3.5\) is an outlier [BL94]. We now test the set of suspected outliers \((29, 36, 98, 99)\) with the modified z-score. The mean of the marks scored by the 25 students is 68 and the median of the absolute deviation about the median (MAD) for the data is 6. For \(x_i = 29\), \(Z_{\text{mi}} = \left| 29 - 68 \right| / 6 = 6.5 > 3.5\) (the
standard value). Similarly, the $Z_{ni}$ of scores 36, 98, and 99 are obtained as 5.33, 5.0, and 5.10 respectively. It can therefore be concluded that the scored marks 29, 36, 98 and 99 are outliers since their computed statistic are each greater than the standard test value of 3.5

2.1.3 The Grand Tour Method

The Grand Tour algorithm was proposed by Asimov [Asi85] for detection of outliers in multidimensional data. The Grand Tour has gained a lot of popularity in the Statistics community. Several extensions have been proposed to the algorithm for analyzing different types of data [Bar97]. The principle of the Grand Tour algorithm is rotating the space containing the data points geometrically. Data objects that are not outliers will fall within an ellipse always, while those objects likely to be outliers will fall outside the ellipse at outstanding positions.

The steps in the Grand Tour algorithm should be carried out continuously several hundred times. The points found outside the concentration ellipse are suspected to be atypical (outliers). The number of times a particular point is noticed beyond the border of the concentration ellipse is recorded and a count plot is made. The objects that tend to have more points outside the concentration ellipse are noted as suspected outliers and tested using regression analysis. In [Bar98], Bartkowiak tests the outliers produced by the Grand Tour algorithm following these three steps for the analysis of the residual to confirm the outliers.

1. Regression analysis is performed using the entire data set. The regression model is: $y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n + e$, where $y$ is the dependent variable, $b_0, b_1, b_2, ..., b_n$ are parameters to be estimated, $x_1, x_2, x_3, ..., x_n$ are the independent variables and $e$ is the residual.

2. The same regression model is used but in estimating the parameters $b_0, b_1, b_2, ..., b_n$, the outliers are dropped from the actual data.

3. In the third step, artificial variables are introduced to account for the special effect of the outliers. That is: $y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n + b_\gamma + e$, where $\gamma$ is an
artificial variable, $\gamma_i = 1$, for $i \in$ outliers, $\gamma_i = 0$, for $i \notin$ outliers. Artificial variables are introduced for each outlier present in the data. In all the three different cases, regression parameters (squared multiple correlation coefficient ($R^2$), residual standard deviation ($\delta$)) are noted and compared with standard values read from the F-distribution. A Conclusion is drawn depending on whether the computed values agree or disagree with the standard values to confirm or disprove the outliers.

2.1.4 **Drawbacks of Distribution-Based Outlier-Mining Approach**

The major drawbacks of the distribution-based methods are that almost all the discordancy tests available are for single variable data, which makes them less practically useful for multidimensional data. The few other tests available for multivariate data are very complex to understand and use. Secondly, it is required that the underlying statistical distribution of the data of interest should be known before any known discordancy test could be applied. This is particularly difficult because in a typical data mining environment it is practically impossible to assign a specific statistical distribution to the underlying data. Data usually have several attributes and do not follow any statistical distribution.

2.2 **Depth-Based Outlier-Mining Approach**

The depth-based outlier-mining approach for finding outliers was developed to solve the problems of distribution fitting (the situation where data should necessarily belong to a particular standard statistical distribution) and restriction to single (univariate) variable data in the distribution-based approach. This approach is based on some definition of depth, where data objects are organized in layers in a data space with the expectation that shallow layers are more likely to contain outlying data objects than are deep layers. In [Tuk75], a robust notion of depth called “depth contours” was introduced with the assumption that a point $p$ in a space is of depth $k$ if $k$ is the minimum number of data points that have to be removed to expose $p$. The $k$-th depth contour makes boundary between all points with depth $k$ and all those with depth $k+1$. An algorithm for computing 2-D depth contours was developed by Ruts et al. [RR96] called ISODEPTH. ISODEPTH
was coded using Fortran, which accepts 2-dimensional input data, a list of \( x \) and their corresponding \( y \) values and returns their depth. The depth returned is used to determine whether the point is an outlier or not. ISODEPTH has two major drawbacks. First, it relies on the computation of dividers (dividers explained next) for all \( n \) data points in the data set having at least quadratic complexity in \( n \). Secondly, it also relies on non-existence of collinear points meaning ISODEPTH has to remove all collinear points before computing depth. This step weakens the performance of ISODEPTH if there are many collinear points in the dataset.

In [JKN98], a fast algorithm for computing 2-dimensional depth contours called Fast Depth Contour (FDC) is proposed. FDC is robust against collinear points meaning FDC does not spend time removing collinear points. FDC restricts the computation of dividers to only a selected subset of points thereby making it more efficient compared to ISODEPTH, which has to compute dividers for all \( n \) points in the dataset. Hence, FDC scales better than ISODEPH.

2.2.1 Dividers

Given a point cloud \( D \) consisting of \( n \) points, a line \( L \) is an e-divider of \( D \) if there are e points in \( D \) to the left of \( L \), and \((n-e)\) points in \( D \) to the right of \( L \). For the sake of outlier detection, whenever \( e \leq (n-e) \), then e points are to the ‘outside’ of \( L \) and the remaining to the ‘inside’ of \( L \). Also, e-divider \( L \) divides the convex hull of \( D \) into two sub-regions called “inside region” and “outside region” denoted by IR (L) and OR (L) respectively. The intersection of all inside region of a collection of e-dividers is called \( e \)-intersected inside region [JKN98].

2.2.2 Problems of Depth-Based Outlier-Mining Approach

Theoretically, depth-based algorithms are supposed to work for high dimensions, thereby finding all outliers in multidimensional data but in practice they don’t work for more than two dimensions. Hence, they are unable to solve the problems of the distribution-based methods [JKN98, RP96].
2.3 Distance-Based Outlier-Mining Approach

The distance-based outlier-mining algorithms compute the distances of data objects from each other based on some distance algorithms. Outliers are data objects that tend to have higher distances from all other data objects. These algorithms are intended to solve the problem of scalability encountered in the earlier algorithms. In [AAR96], a technique for determining deviation/exception called the exact exception, which depends on the dissimilarity within a dataset, was proposed. The proposed algorithm does not require measuring any metric distance as in the case of most clustering algorithms, but rather requires a function that gives the degree to which elements in a data set causes the dissimilarity (heterogeneity) of the data set to increase.

The detection of deviation using implicit redundancy has been placed under distance-based outlier-mining approach because it measures the dissimilarity, which can be viewed as a form of distance. It should however, be noted that exact exception is actually different from the distance-based approach proposed by Knorr et al. [KN97]. The exact exception algorithm declares a set of object(s) to be outlier(s) if the removal of such object(s) causes the remaining objects of the set to be more homogenous whereas the distance-based algorithms define outliers as data objects that tend to have higher distances from all other data points.

2.3.1 The Exact Exception Algorithm

Let I be a set of items. Consider all the possible subsets of I called the power set P(I), then according to Arming et al. [AAR96], the item(s) which if removed from the power set contributes most to the dissimilarity of the itemset I with the least number of items constitutes an exception set (outlier set). In other words, an object which if eliminated contributes most to the homogeneity of the remaining set is an outlier (that is the presence of that object makes the set more heterogeneous). A smoothing factor is a measure of how much dissimilarity can be reduced by removing an element from the power set. The smoothing factor is a function of the cardinality (the number of items in a set) and the dissimilarity functions. The dissimilarity within a set is computed using a variance function for numeric data. The smoothing factor (SF) is an indication of how much dissimilarity can be reduced by removing a subset J of elements from the set I. It is
also very important to note that the smoothing factor (SF) can be negative for some subset $J$ if the dissimilarity of $(I-J)$ is higher than that of the original set $I$. The function $C(I-J)$ returns the number of items in the set $(I-J)$. It is also worth noting that the outlier set need not be unique and it can also be empty if all the elements in the set are the same. The Exact Exception algorithm is shown in Figure 4.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Identify the set of items.</td>
</tr>
<tr>
<td>2.</td>
<td>Compute dissimilarity within items using a dissimilarity function for each item removed from the power set.</td>
</tr>
<tr>
<td>3.</td>
<td>Compute the cardinality $(C: P(I) \rightarrow \mathbb{R}_0$, with $I_1 \subset I_2 \Rightarrow C(I_1) &lt; C(I_2)$ for all $I_1, I_2 \subseteq I$) for each item(s) removed from the power set.</td>
</tr>
<tr>
<td>4.</td>
<td>Compute the smoothing factor for each item(s) removed from the power set using the formula: $SF(J) := C(I-J) \cdot (D(I) - D(I-J))$, where $J \subset I$, $J \neq I$ is an exception set of $I$ with respect to $D$ and $C$ if $SF(J) \geq SF(I)$ for all $J \subseteq I$.</td>
</tr>
<tr>
<td>5.</td>
<td>Finally, the outlying set will be the set whose removal contributes most to the smoothing factor. In other words the set whose removal has the highest value for the smoothing factor.</td>
</tr>
</tbody>
</table>

Figure 4: The Exact Exception Algorithm

The item set $I = \{4, 10, 11, 14\}$ is used to explain the exact exception algorithm. The motive is to find the outlier(s) in the given set. During the first iteration, 4 is removed from the set $I$ leaving the set $(I-J) = \{10, 11, 14\}$ with cardinality $(C(I-J)) = 3$. The dissimilarity of the set $(I-J)$ is computed as 4.33 using the variance function $\sum_{i=1}^{n} (x_i - \bar{x})^2 / (n-1)$, where $x_i$ is the set of items, $n$ is the number of items and $\bar{x}$ is the mean of the set of items. The corresponding smoothing factor is 39.75 obtained using the function provided $((SF(J) = C(I-J) \cdot (D(I) - D(I-J)))$. The second iteration removes 10 from the remaining set leaving the set $(I-J) = \{11, 14\}$ with cardinality of 2. The computed variance of $(I-J)$ is 4.5 with a smoothing factor of 26.16. The process
continues until each of the elements in the initial set I has been removed and their effect noted as shown in Table 2. The element which if removed has the highest smoothing factor is the outlier. This means the removal of that particular element(s) causes the remaining data element to be more homogenous.

<table>
<thead>
<tr>
<th>Item(s) Removed (J)</th>
<th>Remaining Items (I-J)</th>
<th>Cardinality C(I-J)</th>
<th>Dissimilarity D(I-J)</th>
<th>Smoothing Factor SF(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{4}</td>
<td>{10, 11, 14}</td>
<td>3</td>
<td>4.33</td>
<td>39.75</td>
</tr>
<tr>
<td>{4,10}</td>
<td>{11, 14}</td>
<td>2</td>
<td>4.5</td>
<td>26.16</td>
</tr>
<tr>
<td>{4, 10, 11}</td>
<td>{14}</td>
<td>1</td>
<td>0</td>
<td>17.58</td>
</tr>
<tr>
<td>{11}</td>
<td>{4, 10, 14}</td>
<td>3</td>
<td>25.33</td>
<td>-23.25</td>
</tr>
<tr>
<td>{4, 11}</td>
<td>{10, 14}</td>
<td>2</td>
<td>8</td>
<td>19.16</td>
</tr>
<tr>
<td>{4, 11, 14}</td>
<td>{10}</td>
<td>1</td>
<td>0</td>
<td>17.58</td>
</tr>
<tr>
<td>{10}</td>
<td>{4, 11, 14}</td>
<td>3</td>
<td>26.33</td>
<td>-26.25</td>
</tr>
<tr>
<td>{10, 11}</td>
<td>{4, 14}</td>
<td>2</td>
<td>50</td>
<td>-64.84</td>
</tr>
<tr>
<td>{10, 11, 14}</td>
<td>{4}</td>
<td>1</td>
<td>0</td>
<td>17.58</td>
</tr>
<tr>
<td>{14}</td>
<td>{4, 10, 11}</td>
<td>3</td>
<td>14.33</td>
<td>9.75</td>
</tr>
<tr>
<td>{4, 14}</td>
<td>{10, 11}</td>
<td>2</td>
<td>0.5</td>
<td>34.16</td>
</tr>
<tr>
<td>{4, 10, 14}</td>
<td>{11}</td>
<td>1</td>
<td>0</td>
<td>17.58</td>
</tr>
</tbody>
</table>

Table 2: Exact Exception Using I = {4, 10, 11, 14}

The candidate set J = {4} has the highest smoothing factor with respect to a specified C and D, it is an outlier according to Arning et al. [AAR96]. The set with the highest smoothing factor is the set that is highly dissimilar to all the other data objects. It means that when the set {4} is removed from the list, the remaining set of items {10, 11, 14} are more homogeneous than when {4} is included in the original set.

2.3.2 The Unified Notion of Outliers

The fact that a huge number of outliers go undetected either because there are no specific discordancy tests developed for them or they do not fit into any of the known statistical
distributions motivated Knorr et al. [KN97] to propose a ‘unified notion of outliers’. A unified notion of outlier ($UO$) was defined as follows: “An object $O$ in a dataset $R$ is a $UO(p, D)$ - outlier if at least fraction $p$ of the objects in $R$ is at a distance greater than $D$ from $O$.” The distance between any pair of points is obtained using any distance function (e.g. Euclidean) but the choice of the minimum acceptable distance for an outlier ($D$) and the fraction ($p$) of data to be considered as outliers are left to the user to decide since $p$ and $D$ vary for different applications. The notion of outliers defined in [KN97] is intuitive enough because it captures the general idea of an outlier brilliantly described by Hawkins [Haw80]: “An outlier is an observation that deviates so much from other observations to arouse the suspicion that it was generated by a different mechanism”. This notion of outliers captures situations where observed data do not fit any standard statistical distribution or where no discordancy test has been developed. It has been demonstrated that if an object $k$ is an outlier according to a specific discordancy test, then $k$ is also a $UO(p, D)$-outlier for some specified $p$ and $D$.

In [KN98], the term $DB(p, D)$-outlier is used as a shorthand notation for a distance-based outlier detected using parameters $p$ and $D$. The distance-based outlier concept is not different from the unified notion of outlier. Unlike depth-based methods, which are restricted to low dimensionality ($k<3$), the DB-outlier methods are not restricted computationally. The Index-Based (IB), Nested-Loop (NL) and the Cell-Based (CB) algorithms were proposed for mining all $DB$-outliers. The Index-Based and the Nested-Loop algorithms have a time complexity of $O(kN^2)$, where $k$ is the number of dimensions and $N$ is the dataset size. The Index-Based algorithm does not enjoy much popularity because of the initial high cost of building indexes for finding all $DB(p, D)$-outliers. The Cell-Based algorithm has a time complexity of $O(N)$ which is linear in $N$ for $k<=3$ but exponential when $k>3$ making it unfeasible for most multidimensional applications.

### 2.3.2.1 Drawbacks of Unified Notion

The approach proposed in [KN97, KN98] suffers from three major setbacks. The Nested-Loop and the Index-Based algorithms have a time complexity $O(kN^2)$, which is linear in the number of dimensions($k$) but quadratic in dataset size($N$) making them highly
unfavourable when the data size is large. Both algorithms do not scale well for data mining applications where the size of data is emphatically large. Secondly, the algorithms require the user to specify a minimum acceptable distance for an outlier ($D$), which could be very difficult to guess correctly. Specifying the right $D$ is very important for efficient performance of the algorithms because if $D$ is too large none of the data objects are outliers whereas most of the data objects are labelled outliers when $D$ is too small. Finally, the algorithms do not distinguish between strong and weak outliers.

2.3.3 The Partition-Based Algorithm

In their quest to address the shortfalls of the distance-based algorithms described earlier, Ramaswamy et al. [RRS00] proposed a new definition for outliers based on the distance of a point from its $k^{th}$ nearest neighbour that allows outliers to be ranked. An outlier is defined as follows: "Given $k$ and $n$, a point $p$ is an outlier if no more than $n-1$ other points in the data set have higher value of $D^k$ than $p$. In other words, the top $n$ points with a maximum $D^k$ values are considered outliers". $D^k$ is the distance of a point $p$ from its $k^{th}$ nearest neighbour. In addition to providing rankings for outliers, the proposed definition of outliers does not require the user to specify the distance parameter $D$ (the minimum acceptable distance for an outlier). However, $D$ is computed once the number of outliers is known.

The Partition-Based outlier detection algorithm with a linear time complexity was proposed to address the problem of scalability. It partitions (divides) the entire set of data into disjoint subsets using clustering algorithms (it should be noted that clustering algorithms are used for the purpose of generating clusters and not for detecting outliers). Data objects within each cluster are tested and pruned as soon as it is determined that the entire cluster cannot contain outliers. The remaining partitions that contain outliers form the candidate partition. The candidate partition is used as an input to either the Nested-Loop or Index-Based algorithm to find the outliers. The marks scored by 25 students in an examination given below are used as an example to explain the basic steps of the Partition-Based outlier detection algorithm. $Score = \{60, 70, 80, 90, 96, 84, 71, 65, 66, 72, 83, 97, 98, 86, 73, 81, 64, 75, 88, 92, 89, 87, 76, 78, 79\}$. 

25
1. Generating partitions

The first step of the algorithm involves partitioning the data into disjoint subsets using clustering algorithms. It should be noted that clustering algorithms are employed for the purpose of partitioning and not for finding outliers. For the sake of simplicity, we assume the data are partitioned as follows 60-69, 70-79 etc. as shown in Table 3a.

<table>
<thead>
<tr>
<th>Partitions</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>90+</td>
<td>90, 92, 96, 97, 98</td>
</tr>
<tr>
<td>80+</td>
<td>80, 81, 83, 84, 86, 87, 88, 89</td>
</tr>
<tr>
<td>70+</td>
<td>70, 71, 72, 73, 75, 76, 78, 79</td>
</tr>
<tr>
<td>60+</td>
<td>60, 64, 65, 66</td>
</tr>
</tbody>
</table>

Table 3a: Generating Partitions

2. Computing bounds on $D^k(p)$ to obtain MinDkDist

$D^k(p)$ denotes the distance of a point $p$ from its $k^{th}$ nearest neighbour. The minimum distance that can be included on a possible outlier list is denoted by MinDkDist. MinDkDist is used to determine partitions that are candidate for outliers called candidate set (candset). MinDkDist is obtained by first computing the lower ($P.lower$) and upper ($P.upper$) bounds for all partitions. The lower bound for a partition is obtained by subtracting the smallest number in the entire data set (min_data) from the minimum number in that partition (min_part) (i.e., $P.lower = min_part - min_data$) whereas the upper bound is obtained by subtracting the minimum number in the entire data from the maximum number in that partition (max_part) (i.e., $P.upper = max_part - min_data$). For instance, P.lower and P.upper for partition 90+ are (90-60) = 30 and (98-60) = 38 respectively since the minimum for the entire data is 60. The computation of bounds for the remaining partitions are obtained in a similar manner and shown in Table 3b. MinDkDist is the minimum of all distances in the partition with the maximum lower bound. From the sample data, 90+ is the partition with the maximum lower bound with element {90, 92, 96, 97, 98}. Their corresponding distances from the minimum of the entire
data (\textit{min\_data}) are \{30, 32, 36, 37, 38\}. The minimum of \{30, 32, 36, 37, 38\} gives the \textit{MinDKDist}. Hence, \textit{MinDKDist} = 30.

<table>
<thead>
<tr>
<th>Partitions</th>
<th>90+</th>
<th>80+</th>
<th>70+</th>
<th>60+</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.lower (min_part - 60)</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>P.upper (max_part - 60)</td>
<td>38</td>
<td>29</td>
<td>19</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3b: Computing Bounds

3. \textbf{Identifying candidate partition(s)}

The candidate partition is the partition(s) with the upper bound not less than \textit{MinDKDist} (i.e., \textit{P.upper} >= \textit{MinDKDist}). The elements in the candidate partition(s) form the candidate set. From Table 3b, the only partition with upper bound satisfying this condition is 90+, with members \{90, 92, 96, 97, 98\}. Hence, candidate set = \{90, 92, 96, 97, 98\}.

4. \textbf{Computing outliers from candidate set}

Finally, outliers are determined based on the candidate set. Individual distances are computed from each other and ranked. The top-n points with the highest distances are declared outliers. The distance computation is shown in Table 4c with the top three outliers \{96, 97, 98\} in that order.

<table>
<thead>
<tr>
<th>Points</th>
<th>90</th>
<th>92</th>
<th>96</th>
<th>97</th>
<th>98</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Ranking</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table3c: Computing Outliers

2.4 \textbf{Density-Based Outlier-Mining Approach}

The density-based outlier-mining algorithms concentrate on the density (number of nearest neighbours) of an object with respect to its surrounding neighbourhood. Data points that have high-density neighbourhoods are non-outliers whereas those with low-
density neighbourhoods (very sparse) are likely to be outliers with respect to their neighbourhood. Density-based outlier-mining techniques lay more emphasis on the locality of objects in determining outliers. The existing algorithms compute local outlier factors (LOF) and declare those with very high LOF values as outliers. In other words, LOF shows the degree to which an object is outlying with respect to its neighbours.

2.4.1 Local Outlier Factor (LOF)

Breunig et al. [BKN00] argue that being an outlier is not just a binary property as claimed by the distance-based outlier-mining concept. In other words, data objects can not just be classified as an outlier or a non-outlier, rather every object has some degree of "outlierness" in it. They contend that for many scenarios, it is more appropriate to assign to each object a degree of being an outlier. The degree is called local outlier factor (LOF) because it depends on the remoteness of an object with respect to its surrounding neighbourhood. Outliers are objects that tend to have high LOF values. The algorithm proposed for mining local outliers is called Local Outlier Factor (LOF). LOF is able to detect all forms of outliers including those that could not be detected by the earlier (distance-based, depth-based and distribution-based) algorithms. This was demonstrated with the National Hockey League player statistics for 1996, the German National Soccer League 1998/99 and a synthetic data set containing outliers [BKN00]. In all three cases, LOF identified all the outliers that were found by the distance-based algorithms. In addition, it identified new outliers that could not be detected by the distance-based algorithms.

The problem with the earlier algorithms [KN97, KN98, RSS00] for mining outliers is that they take a "global" view of the entire dataset thereby resulting in "global" outliers. However, in a multidimensional dataset with a more complex structure, there are data objects that might be outliers with respect to their neighbourhood but would not be outliers in the entire dataset. These outliers would not be identified by any of the "global" outlier detection algorithms. The problem is better explained with a visual example using Figure 5.
Consider Figure 5 for instance, $C_1$ and $C_2$ are two separate clusters whereas $O_1$ and $O_2$ are two data objects that are found outside the clusters. The objects $O_1$ and $O_2$ are outliers according to Hawkins ("An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by different mechanism") whereas the objects in $C_1$ and $C_2$ are not. LOF would also label objects $O_1$ and $O_2$ as outliers, but within the framework of distance-based outliers only $O_1$ would be a reasonable outlier since the distance between $O_2$ and its nearest neighbour would be less than some of the distances within the cluster. Thus, LOF captures all such outliers as shown in Figure 5 that could not be identified by the distance-based outlier algorithms. The inability of the distance-based algorithms to identify outliers like the ones shown in Figure 5 is largely due to the definition of the distance-based outlier concept rather than the efficiency of the individual distance-based algorithms themselves.

2.4.1.1 Computation of Local Outlier Factor (LOF)

This section explains how local outlier factor (LOF) of an object is computed. Let $D$ be a database and $p, q, o$ be objects in $D$, then the distance between a pair of objects $p$ and $q$ is denoted as $d(p,q)$. Supposing the objects $p_i$ and $q_j$ are represented by $(x_i, y_i, z_i)$ and $(x_j, y_j, z_j)$ respectively, then the distance between the pair of points is given by
\[ d(p_i, q_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \]

where \( i, j \) are the labels of \( p \) and \( q \) respectively. MinPts denoted as \( k \), is the minimum number of objects considered within a neighbourhood. For instance, if we choose MinPts to be 6, then we expect every object in the dataset to have at least 6 neighbours. The value of MinPts is very crucial to all density-based algorithms since it determines the value of LOF and subsequently which data objects are outliers. However, to choose the appropriate value for MinPts, the user exercises some discretion. In most cases, the appropriate value for MinPts depends on the size of data being mined. The value of MinPts should be high enough to remove any statistical fluctuations in LOF. The reason is that LOF values changes as MinPts changes even with the same data. The changes are more pronounced with smaller values of MinPts but LOF stabilises when MinPts is large enough. The algorithm proceeds to determine the value of LOF for each object once the appropriate value of MinPts has been determined. The intermediate steps involved in computing LOF are computing k-distance of \( p \), neighbourhood distance of \( p \), reachability distance of \( p \) and local reachability density of \( p \).

1. **Computing k-distance of \( p \)**

The motive for computing k-distance of \( p \) is to be able to determine which data objects constitute the neighbours of \( p \). It also helps in determining whether the neighbourhood around \( p \) is dense or sparse in relation to its neighbours. In simple terms, k-distance of \( p \) is the maximum distance from object \( p \) when every object in the data set is considered to have at least k-neighbours. Any object with a distance from \( p \) that is greater than k-distance of \( p \) can not be considered a neighbour \( p \). A high value means the area around \( p \) is less dense whereas a low value indicates a dense area around \( p \). The diagram in Figure 6 explains k-distance of \( p \) and the objects that constitute the neighbours of \( p \).
Figure 6: K-distance of $P_1$ and Neighbours of $P_1$

Figure 6 shows objects $P_2$, $P_3$, $P_4$, $P_5$ and $P_6$ and their respective distances from $P_1$. There are $(k-1)$ objects inside the circle, $(n-k-1)$ objects outside the circle and 1 object on the boundary of the circle, where $k$ is the MinPts and $n$ is the size of data. The distance of the object on the boundary of the circle ($P_4$) from $P_1$ is the $k$-distance of $P_1$ with a value of 5 (i.e., 3-distance of $P_1 = 5$). It is the maximum distance from $P_1$ when considering only the 3 most nearest distinct distances from $P_1$. The object on the boundary of the circle and those within the circle constitute the neighbours of $P_1$ (i.e., $P_2$, $P_5$, $P_4$). The value of $k$-distance of $P_1$ changes if the number of distinct distances changes. For instance, 3-distance of $P_1 = 5$ and 4-distance of $P_1 = 6$.

Mathematically, $k$-distance of p, denoted as $k$-distance(p) is obtained as follows:

1. Compute the distances of all objects from p
2. Select the $k$-distinct minimum distances
3. Select the maximum of the $k$-distinct minimum distance as $k$-distance of p.

Suppose $P_1$, $P_2$, $P_3$ and $P_4$ are objects in a database $D$ and we want to compute their $k$-distances. Assume further that the distances between any pair of objects have been obtained using the Euclidean distance function. The distance between
$P_1$ and $P_2$ is represented as $P_1P_2$. Assume further that the pair of distances obtained are $P_1P_2 = 4$, $P_1P_3 = 3$, $P_1P_4 = 7$, $P_2P_3 = 5$, $P_2P_4 = 6$, and $P_3P_4 = 8$. It is important to note that $P_1P_2 = P_2P_1$, $P_1P_3 = P_3P_1$ etc. This example will be used to explain all the steps in the local outlier factor algorithm and how LOF is obtained using $k=2$, where $k$ is the MinPts.

The $k$-distance of $P_1$ is computed as follows:

1. First, compute the distances of all objects from $P_1$ using a distance function. The distances are obtained as $P_1P_2 = 4$, $P_1P_3 = 3$, $P_1P_4 = 7$ (assumed previously).

2. Next, select the first 2-distinct minimum distances from $P_1$. All distances from $P_1$ are ordered and the first 2-minimum distinct distances are chosen (i.e., min$(P_1P_2 = 4, P_1P_3 = 3, P_1P_4 = 7)$). The resulting output is $(3, 4)$.

3. Finally, the maximum of the first 2-minimum distinct distances is selected as $k$-distance of $P_1$. Thus, $k$-distance$(P_1) = \max(3, 4)$. Hence, $k$-distance$(P_1) = 4$.

For $k$-distance of $P_2$:

1. First, compute the distances of all objects from $P_2$ using a distance function. These distances are obtained as $P_1P_2 = 4$, $P_2P_3 = 5$, $P_2P_4 = 6$.

2. Next, select the first 2-distinct minimum distances from $P_2$. All distances from $P_2$ are ordered and the first 2-minimum distinct distances chosen (i.e., $\min(P_1P_2 = 4, P_2P_3 = 5, P_2P_4 = 6)$). The resulting output is $(4, 5)$.

3. Finally, the maximum of the first 2-minimum distinct distances is selected as $k$-distance of $P_2$. Thus, $k$-distance$(P_2) = \max(4, 5)$. Hence, $k$-distance$(P_2) = 5$.

The $k$-distances of $P_3$ and $P_4$ are 5 and 7 respectively computed in a similar manner.
2. Finding k-distance neighbourhood of p

The k-distance neighbourhood of \( p \) (\( N_k(p) \)), contains every object that has a distance not greater than k-distance(p). The rationale for computing the k-distance neighbourhood is to find the nearest neighbours of each object. Data objects are later declared outliers based on their neighbourhood and not the entire data. Mathematically, \( N_k(p) = \{ q \in D \setminus \{ p \} \mid d(p, q) \leq k\text{distance}(p) \} \). Using the data given, \( N_k(P_1) \) contains every object with distance from \( P_1 \) not greater than \( k\text{distance}(P_1) = 4 \). Since \( P_1P_2 = 4 \), \( P_1P_3 = 3 \) and \( P_1P_4 = 7 \), \( P_2 \) and \( P_3 \) constitute the neighbourhood distance of \( P_1 \). The reason is that the distances of \( P_2 \) and \( P_3 \) from \( P_1 \) are each less than 4. Hence, \( N_k(P_1) = \{ P_2, P_3 \} \). Similarly, \( N_k(P_2) \) contains every object with distance from \( P_2 \) not greater than \( k\text{distance}(P_2) = 5 \). The distances of \( P_1 \) and \( P_3 \) from \( P_2 \) are 3 and 5 respectively, each of which is not greater than \( k\text{distance}(P_2) = 5 \). Hence, \( N_k(P_2) = \{ P_1, P_3 \} \). The k-distance neighbourhood of \( P_3 \) and \( P_4 \) are obtained in a similar manner as \( N_k(P_3) = \{ P_2, P_1 \} \) and \( N_k(P_4) = \{ P_2, P_1 \} \).

3. Computing reachability distance of p

The reachability distance of an object \( p \) with respect to \( o \) is the distance \( d(p, o) \) between the objects when \( p \) is far from \( o \) (i.e., \( d(p, o) > k\text{distance}(o) \)). However, if \( p \) is closer to \( o \) (i.e., \( d(p, o) \leq k\text{distance}(o) \)), then the distance is replaced with the k-distance of \( o \). The objective is to ensure that all the objects within a neighbourhood are very close to each (objects within a neighbourhood are more homogeneous) to remove any fluctuations that may exist in the reachability distances. In addition, LOF stabilizes when the objects within a neighbourhood are uniform (closer to each other) even if MinPts changes. The fluctuations in the reachability distances can also be controlled by choosing a large value for \( k \). The effect of fluctuations is reduced since reachability distances become very similar within the same neighbourhood when \( k \) is large. Hence, the reachability distance of \( p \) with respect to \( o \) denoted as reach-dist_k(p,o) is given by: \( \text{reach\_dist\_k}(p, o) = \text{Max}(k\text{distance}(o), d(p, o)) \).
The sample data are used to show the computation of reachability distances of $P_1$, $P_2$, $P_3$ and $P_4$.

For $P_1$:
1. First, identify $k$-distance neighbourhood of $P_1$: i.e., $N_k(P_1) = (P_2, P_3)$.
2. Since $k$-distance neighbourhood of $P_1$ contains $P_2$ and $P_3$, the reachability distance is computed based on $P_2$ and $P_3$.

For $P_2$ within the neighbourhood of $P_1$:
\[ \text{reach-dist}_k(P_1, P_2) = \max (k\text{-distance}(P_2), \text{dist}(P_1, P_2)) \]
\[ = \max (5, 4), \text{ since } k\text{-distance}(P_2) = 5 \text{ and } \text{dist}(P_1, P_2) = 4. \]
\[ = 5. \]

For $P_3$ within the neighbourhood of $P_1$:
\[ \text{reach-dist}_k(P_1, P_3) = \max \{k\text{-distance}(P_3), \text{dist}(P_1, P_3)\} \]
\[ = \max (5, 3), \text{ since } k\text{-distance}(P_3) = 5 \text{ and } P_1P_3 = 3. \]
\[ = 5. \]

Hence, $\text{reach-dist}_k(P_1, 0) = (5, 5)$. It is a combination of reachability distances of the neighbours of $P_1$. Similarly, the reachability distances of $P_2$, $P_3$ and $P_4$ are obtained as $\text{reach-dist}_k(P_2, 0) = (5, 4)$, $\text{reach-dist}_k(P_3, 0) = (5, 4)$ and $\text{reach-dist}_k(P_4, 0) = (6, 7)$ respectively.

4. **Computing the local reachability density of p**

The neighbourhood of an object can have more than $\text{MinPts}(k)$ number of objects within it, thus, making it very impracticable comparing reachability distances. In other words, different objects may have different number of neighbours for the same $\text{MinPts}$ value. The fact that reachability distances can not be effectively compared leads us to the computation of local reachability density. The local reachability density of an object $p$, denoted $lrd_k(p)$ is the inverse of the average reachability distances from $k$-nearest-neighbours of $p$.

\[
lrd_k(p) = \frac{\sum_{o \in N_k(p)} \text{reach-dist}_k(p, o)}{|N_k(p)|} \]

34
The local reachability densities for \( P_1, P_2, P_3 \) and \( P_4 \) are computed using the sample data as follows: 
\[
lrd_k(P_1) = 1/\{(5+5)/2\} = 2/10 , \quad \text{since } (5, 5) \text{ constitutes the local reachability distance of } P_1 \text{ and the number of k-distance neighbours is 2. Also,}
\]
\[
lrd_k(P_2) = 2/9, \quad lrd_k(P_3) = 2/9 \quad \text{and} \quad lrd_k(P_4) = 2/13.
\]

5. **Computing the local outlier factor of** \( p \)

The local outlier factor is a ratio that determines whether or not an object is an outlier with respect to its neighbourhood. The local outlier factor is affected by the value of MinPts. The same data may have different local outlier factor values for the same objects depending on the value of MinPts. However, local outlier factor values stabilize when MinPts is chosen large enough [BKN00]. The local outlier factor of an object \( p \), denoted \( LOF_k(p) \) is the average of the ratios of local reachability density of \( p \) and that of \( p \)'s k-nearest-neighbours. Generally, \( LOF_k(p) \) will be very high if the local reachability density of \( p \) is lower than those of \( p \)'s nearest neighbours.

\[
LOF_k(p) = \frac{\sum_{o \in N_k(p)} lrd_k(o)}{lrd_k(p)} \frac{lrd_k(p)}{|N_k(p)|}
\]

We compute \( LOF_k(P_1) \) using the sample data as follows:

\[
LOF_k(P_1) = \frac{\sum_{o \in N_k(p)} lrd_k(p_2) + lrd_k(p_2)}{lrd_k(p_1)} \frac{lrd_k(p_1)}{|N_k(p)|}, \quad \text{where } P_2 \text{ and } P_3 \text{ are the 2-nearest neighbours of } P_1 \text{ and } |N_k(p_1)| = 2.
\]

\( LOF_k(P_1) = (((0.22 + 0.22) / 0.20) / 2), \) where \( lrd_k(P_2) = 0.22, \ lrd_k(P_3) = 0.22 \) and \( lrd_k(P_1) = 0.20. \) Hence, \( LOF_k(P_1) = 1.1 \)

Similarly, \( LOF_k(P_2) = \frac{\sum_{o \in N_k(p)} lrd_k(p_1) + lrd_k(p_3)}{lrd_k(p_2)} \frac{lrd_k(p_2)}{|N_k(p)|}, \) where \( P_1 \) and \( P_3 \) are the 2-nearest neighbours of \( P_2 \) and \( |N_k(p_2)| = 2. \)
\[ \text{LOF}_k(P_2) = ((0.22 + 0.20) / 0.22) / 2, \] where \( \text{lr}_{dk}(P_2) = \text{lr}_{dk}(P_3) = 0.22 \) and \( \text{lr}_{dk}(P_1) = 0.20. \) Hence, \( \text{LOF}_k(P_2) = 0.95. \) In a similar manner, the local outlier factor values for \( P_3 \) and \( P_4 \) are obtained. Table 4 shows the results of applying LOF algorithm to the sample data. The order of the outliers based on the computed local outlier factor values from the lowest to the highest is \( P_2, P_3, P_1 \) and \( P_4. \)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )-distance(p)</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>( N_k(p) )</td>
<td>( P_3, P_2 )</td>
<td>( P_3, P_1 )</td>
<td>( P_2, P_1 )</td>
<td>( P_2, P_1 )</td>
</tr>
<tr>
<td>reach-dist_k(p,o)</td>
<td>5,5</td>
<td>5,4</td>
<td>5,4</td>
<td>6,7</td>
</tr>
<tr>
<td>( Lr_{dk}(p) )</td>
<td>0.20</td>
<td>0.22</td>
<td>0.22</td>
<td>0.154</td>
</tr>
<tr>
<td>( \text{LOF}_k(p) )</td>
<td>1.1</td>
<td>0.95</td>
<td>0.95</td>
<td>1.30</td>
</tr>
</tbody>
</table>

Table 4: Results from LOF Computation

2.4.2 Finding Top-\( n \) Local Outliers

In [JTH01], a Micro-Cluster-Based algorithm for finding top-\( n \) local outliers based on the concept of local outlier factor is proposed. The motivation for this algorithm stems from the fact that LOF [BKN00] has to compute local outlier factor (LOF) for every object in the database making the outlier determination computationally very expensive and less scalable. The proposed algorithm computes the lower and upper limits for each cluster based on their local reachability densities. The limits are used to determine which of the clusters could possibly contain outliers. Those clusters that cannot possibly contain outliers are eliminated from the possible outlier candidate set. Although pruning of the clusters based on the computed limits reduces the number of LOF value computations, it still relies on computation of local reachability distances and local reachability densities for all objects in the data set. A micro-cluster \( mc(n, c, r) \) is a summarized representation of a group of data \( P_1, \ldots, P_n \), which are so close together that they are likely to belong to the same cluster, where \( n, c \) and \( r \) represent the number of objects in the cluster, the center of the cluster and the radius of the cluster respectively.
2.4.3 Drawbacks of Existing Density-Based Algorithms

The major drawback of LOF is that it is computationally expensive since local reachability distance has to be computed for every object in the dataset before LOF values are obtained. There are at least \( kN \) computations and comparisons for computing reachability distances where \( k \) is the MinPts and \( N \) is the dataset size. Since \( N \) is usually very large, the effect of \( kN \) on the entire LOF algorithm is very huge, which makes the algorithm less scalable.
3. Local Sparsity Coefficient Outlier Algorithm

The problem domain for the thesis is National Hockey League (NHL) data. Although the algorithms proposed in the thesis can be applied to credit card data, soccer and many other data, we have chosen NHL so that we can compare the output and performance of our algorithms to similar algorithms discussed in the literature. We will specifically use NHL 1995 and 1996 player statistics obtained from NHL site [NHL95].

3.1 Analysis of Problems in LOF

The local outlier factor algorithm is applied to mine for outliers in the NHL player statistics data for 1995. The NHL player statistics data set has many attributes, but we will consider the three attributes total points score (PTS), plus/minus (+/-) and penalty minutes (PIM). Table 5 shows the sample data drawn from 1995 NHL player statistics. Each PTS, +/-, PIM pair constitute a point in a plane and are represented as Pᵢ, i = 1 ... n.

<table>
<thead>
<tr>
<th>Player</th>
<th>Total Points Score (PTS)</th>
<th>Plus/minus (+/-)</th>
<th>Penalty Minutes (PIM)</th>
<th>Represented (Pᵢ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAUL KARIYA</td>
<td>39</td>
<td>-17</td>
<td>4</td>
<td>P₁</td>
</tr>
<tr>
<td>SHAUN VAN ALLEN</td>
<td>29</td>
<td>-4</td>
<td>32</td>
<td>P₂</td>
</tr>
<tr>
<td>STEPHAN LEBEAU</td>
<td>24</td>
<td>6</td>
<td>12</td>
<td>P₃</td>
</tr>
<tr>
<td>TODD KRYGIER</td>
<td>22</td>
<td>1</td>
<td>10</td>
<td>P₄</td>
</tr>
<tr>
<td>PETER DOURIS</td>
<td>21</td>
<td>4</td>
<td>12</td>
<td>P₅</td>
</tr>
<tr>
<td>MILOS HOLAN</td>
<td>10</td>
<td>4</td>
<td>14</td>
<td>P₆</td>
</tr>
<tr>
<td>JASON YORK</td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>P₇</td>
</tr>
<tr>
<td>TOM KURVERS</td>
<td>7</td>
<td>-13</td>
<td>6</td>
<td>P₈</td>
</tr>
<tr>
<td>R. LADouceUR</td>
<td>6</td>
<td>2</td>
<td>36</td>
<td>P₉</td>
</tr>
<tr>
<td>DAVE KARPA</td>
<td>6</td>
<td>-1</td>
<td>91</td>
<td>P₁₀</td>
</tr>
</tbody>
</table>

Table 5: Sample Data From NHL 1995 Player Statistics

The distance between any pair of players (points) is computed using the distance function, \( d(pᵢ, qⱼ) = \sqrt{(xᵢ - xⱼ)^2 + (yᵢ - yⱼ)^2 + (zᵢ - zⱼ)^2} \), where i, j are the coordinates of p and q respectively. The distances of every point from each other for the sample data
are computed using the defined distance function. The outputs of these are shown in Table 6.

<table>
<thead>
<tr>
<th>Players</th>
<th>P_1</th>
<th>P_2</th>
<th>P_3</th>
<th>P_4</th>
<th>P_5</th>
<th>P_6</th>
<th>P_7</th>
<th>P_8</th>
<th>P_9</th>
<th>P_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>0</td>
<td>32.4</td>
<td>28.6</td>
<td>25.5</td>
<td>28.8</td>
<td>37.2</td>
<td>35.2</td>
<td>32.3</td>
<td>49.7</td>
<td>94.4</td>
</tr>
<tr>
<td>P_2</td>
<td>32.4</td>
<td>0</td>
<td>22.9</td>
<td>23.6</td>
<td>23</td>
<td>27.4</td>
<td>29.7</td>
<td>35.2</td>
<td>24.1</td>
<td>63.4</td>
</tr>
<tr>
<td>P_3</td>
<td>28.6</td>
<td>22.9</td>
<td>0</td>
<td>5.7</td>
<td>3.6</td>
<td>14.3</td>
<td>14.7</td>
<td>26.2</td>
<td>30.3</td>
<td>81.3</td>
</tr>
<tr>
<td>P_4</td>
<td>25.5</td>
<td>23.6</td>
<td>5.7</td>
<td>0</td>
<td>3.7</td>
<td>13</td>
<td>12</td>
<td>20.9</td>
<td>30.5</td>
<td>82.6</td>
</tr>
<tr>
<td>P_5</td>
<td>28.8</td>
<td>23</td>
<td>3.6</td>
<td>3.7</td>
<td>0</td>
<td>11.2</td>
<td>11.4</td>
<td>22.8</td>
<td>28.4</td>
<td>80.6</td>
</tr>
<tr>
<td>P_6</td>
<td>37.2</td>
<td>27.4</td>
<td>14.3</td>
<td>13</td>
<td>11.2</td>
<td>0</td>
<td>4.5</td>
<td>19</td>
<td>22.4</td>
<td>77.3</td>
</tr>
<tr>
<td>P_7</td>
<td>35.2</td>
<td>29.7</td>
<td>14.7</td>
<td>12</td>
<td>11.4</td>
<td>4.5</td>
<td>0</td>
<td>15.8</td>
<td>26.3</td>
<td>81.2</td>
</tr>
<tr>
<td>P_8</td>
<td>32.3</td>
<td>35.2</td>
<td>26.2</td>
<td>20.9</td>
<td>22.8</td>
<td>19</td>
<td>15.8</td>
<td>0</td>
<td>33.6</td>
<td>85.8</td>
</tr>
<tr>
<td>P_9</td>
<td>49.7</td>
<td>24.1</td>
<td>30.3</td>
<td>30.5</td>
<td>28.4</td>
<td>22.4</td>
<td>26.3</td>
<td>33.6</td>
<td>0</td>
<td>55.1</td>
</tr>
<tr>
<td>P_{10}</td>
<td>94.4</td>
<td>63.4</td>
<td>81.3</td>
<td>82.6</td>
<td>80.6</td>
<td>77.3</td>
<td>81.2</td>
<td>85.8</td>
<td>55.1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6: Computed Distances

The five steps described below involved in the local outlier factor algorithm are then followed to obtain the LOF value for each object using k = 4, where k is the MinPts. The value of k should be chosen large enough to remove any effect of fluctuations that might exist in LOF values. It has been proved experimentally in [BKN+00] that LOF values stabilize when k (MinPts) is large. Thus large k leads to a better and a more stable set of outliers. We have chosen k = 4 since our sample data set is very small, however, k is large enough to stabilise LOF values.

1. **Computing k-distance(p)**

The k-distances of points P_1, P_2 … P_{10} are each computed using k= 4. Computing k-distance of P_1 requires finding all distances of objects from P_1 (shown in Table 6). The next step involves selecting the 4-distinct minimum distances from P_1 are selected as 28.6, 25.5, 28.8 and 32.3 from the set of all distances from P_1. Finally, k-distance of P_1 (k-distance(P_1)) is the maximum of the 4-distinct minimum distances from P_1. Thus, k-distance(P_1) = max(28.6, 25.5, 28.8, 32.3) = 32.3. Also, k-distance of P_2 (k-distance(P_2)) is obtained by first computing the
distances of all objects from $P_2$ (shown in Table 6). The second step requires selecting the 4-distinct minimum distances from $P_2$ (i.e., 22.9, 23.6, 23, 24.1). Finally, $k$-distance of $P_2$ is the maximum of the 4-minimum distances. Thus, $k$-distance($P_2$) = 24.1. The $k$-distances of the remaining objects are obtained in the same manner and shown in Table 7.

2. Finding $k$-distance neighbourhood

The $k$-distance neighbourhood of $p$, $N_k(p)$, contains every object with distance from $p$ not more than $k$-distance($p$). For instance, $k$-distance($P_1$) = 32.3 and all objects whose distance from $P_1$ are not more 32.3 are $P_3$, $P_4$, $P_5$ and $P_8$ with distances 28.6, 25.5, 28.8 and 32.3 respectively. Hence, $N_k(P_1)$ = ($P_3$, $P_4$, $P_5$, $P_8$). The distances of $P_3$, $P_4$, $P_5$ and $P_9$ from $P_2$ are 22.9, 23.6, 23, 24.1 respectively, each of which is not more than $k$-distance($P_2$) = 24.1. Thus, $N_k(P_2)$ = ($P_3$, $P_4$, $P_5$, $P_9$). The $k$-distance neighbourhood of the remaining objects are obtained following the same procedure and shown in Table 7.

3. Computing reachability distance of $p$ with respect to object $o$.

The reachability distance of an object $p$ with respect to an object $o$ denoted $reach-dist_k(p,o)$ is the maximum of $k$-distance($o$) and the distance between $p$ and $o$. It is given by $reach-dist_k(p,o) = \max\{k$-distance($o$), dist($p,o$)$\}$. Reachability distance of $P_1$ is computed with respect to $P_3$, $P_4$, $P_5$ and $P_8$ since they form the nearest neighbours of $P_1$. Thus, $p$ in the above equation is repeatedly replaced with $P_1$ while the objects within the neighbourhood of $P_1$ replaces $o$ in turn until the list is exhausted. Hence, $reach-dist_k(P_1) = \{\max(k$-distance($P_3$), dist($P_1$,$P_3$)), \max(k$-distance($P_4$), dist($P_1$,$P_4$)), \max(k$-distance($P_5$), dist($P_1$,$P_5$)), \max(k$-distance($P_8$), dist($P_1$,$P_8$))\}$. Replacing the points with their distances and $k$-distances, $reach-dist_k(P_1) = \{\max(14.7, 28.6), \max(13, 25.5), \max(11.4, 28.6), \max(22.8, 32.3)\}$ obtained from Table 6 and Table 7. Hence, $reach-dist_k(P_1) = \{28.6, 25.5, 28.6, 32.3\}$. 

40
Moreover, the k-distance neighbourhood of $P_2$ is \{P_3, P_4, P_5 and P_9\}. The reachability distance of $P_2$ is computed with respect to its neighbourhood as: 
\[ \text{reach-dist}_k(P_2) = \{\max(\text{k-distance}(P_3), \text{dist}(P_2,P_3)), \max(\text{k-distance}(P_4), \text{dist}(P_2,P_4)), \max(\text{k-distance}(P_5), \text{dist}(P_2,P_5)), \max(\text{k-distance}(P_9), \text{dist}(P_2,P_9))\} \].

Thus, \[ \text{reach-dist}_k(P_2) = \{(14.7, 22.9), (13, 23.6), (11.4, 23), (28.4, 24.1)\} \]. Hence, \[ \text{reach-dist}_k(P_2) = \{22.9, 23.6, 23, 28.4\} \]. The reachability distances of the remaining objects are computed in the same manner. The k-distances are obtained from Table 7 whereas the actual distances obtained from Table 6.

4. **Computing local reachability density of p (lr\_d\_k(p))**

The local reachability density of p is the inverse of the average reachability distances from the k-nearest neighbours of p. 
\[ \text{lr\_d}_k(P_1) = 4 / (28.6 + 25.5 + 28.6 + 32.3) \] where 4 is the number of nearest-neighbours of $P_1$. Hence, \[ \text{lr\_d}_k(P_1) = 0.03478 \]. Since the reachability distance of $P_2$ is \{22.9, 23.6, 23, 28.4\}, 
\[ \text{lr\_d}_k(P_2) = 4 / (22.9 + 23.6 + 23 + 28.4) = 0.0409 \]

5. **Computing the local outlier factor (LOF)**

The local outlier factor is the average of the ratios of the local reachability density of p and those of p’s nearest-neighbours. The number of nearest-neighbours of $P_1$ is 4 i.e. \[ |N_k(P_1)| = 4 \]. 
\[ \text{LOF}_k(P_1) = (\text{lr\_d}_k(P_3) + \text{lr\_d}_k(P_4) + \text{lr\_d}_k(P_5) + \text{lr\_d}_k(P_6)) / 4\text{lr\_d}_k(P_1) \], where \[ \text{lr\_d}_k(P_3), \text{lr\_d}_k(P_4), \text{lr\_d}_k(P_5), \text{lr\_d}_k(P_6) \] are the local reachability densities of $P_3, P_4, P_5$ and $P_6$ which constitute the 4-nearest neighbours of $P_1$. Table 7 shows the results obtained by applying LOF to the sample NHL data. The top 4 outliers revealed are $P_{10}, P_1, P_9$ and $P_2$ with $P_{10}$ being the strongest and $P_2$ the weakest.
<table>
<thead>
<tr>
<th>k-distance(p)</th>
<th>P_1</th>
<th>P_2</th>
<th>P_3</th>
<th>P_4</th>
<th>P_5</th>
<th>P_6</th>
<th>P_7</th>
<th>P_8</th>
<th>P_9</th>
<th>P_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_k(p)</td>
<td>P_3</td>
<td>P_3</td>
<td>P_4</td>
<td>P_5</td>
<td>P_3</td>
<td>P_4</td>
<td>P_3</td>
<td>P_4</td>
<td>P_3</td>
<td>P_2</td>
</tr>
<tr>
<td></td>
<td>P_4</td>
<td>P_4</td>
<td>P_5</td>
<td>P_5</td>
<td>P_4</td>
<td>P_4</td>
<td>P_4</td>
<td>P_5</td>
<td>P_5</td>
<td>P_5</td>
</tr>
<tr>
<td></td>
<td>P_5</td>
<td>P_5</td>
<td>P_5</td>
<td>P_6</td>
<td>P_6</td>
<td>P_5</td>
<td>P_5</td>
<td>P_6</td>
<td>P_6</td>
<td>P_6</td>
</tr>
<tr>
<td></td>
<td>P_6</td>
<td>P_5</td>
<td>P_9</td>
<td>P_7</td>
<td>P_7</td>
<td>P_6</td>
<td>P_7</td>
<td>P_7</td>
<td>P_7</td>
<td>P_9</td>
</tr>
<tr>
<td>Reach_dist(p)</td>
<td>28.6</td>
<td>22.9</td>
<td>13</td>
<td>14.7</td>
<td>14.7</td>
<td>14.7</td>
<td>14.7</td>
<td>20.9</td>
<td>24.1</td>
<td>63.4</td>
</tr>
<tr>
<td></td>
<td>25.5</td>
<td>23.6</td>
<td>11.4</td>
<td>11.5</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>22.8</td>
<td>28.4</td>
<td>80.6</td>
</tr>
<tr>
<td></td>
<td>28.8</td>
<td>23</td>
<td>14.3</td>
<td>14.3</td>
<td>14.3</td>
<td>11.4</td>
<td>11.4</td>
<td>19</td>
<td>22.4</td>
<td>77.3</td>
</tr>
<tr>
<td></td>
<td>32.3</td>
<td>28.4</td>
<td>14.7</td>
<td>14.7</td>
<td>14.7</td>
<td>14.7</td>
<td>14.3</td>
<td>15.8</td>
<td>26.3</td>
<td>55.1</td>
</tr>
<tr>
<td>Sum of (reach_dist)</td>
<td>115.2</td>
<td>97.9</td>
<td>53.4</td>
<td>55.2</td>
<td>56.7</td>
<td>53.8</td>
<td>53.4</td>
<td>78.5</td>
<td>101.2</td>
<td>276.4</td>
</tr>
<tr>
<td>Lrd_k(p)</td>
<td>0.035</td>
<td>0.041</td>
<td>0.075</td>
<td>0.072</td>
<td>0.071</td>
<td>0.074</td>
<td>0.075</td>
<td>0.051</td>
<td>0.04</td>
<td>0.014</td>
</tr>
<tr>
<td>LOF_k(p)</td>
<td>1.92</td>
<td>1.57</td>
<td>0.97</td>
<td>1.02</td>
<td>1.04</td>
<td>0.99</td>
<td>0.97</td>
<td>1.43</td>
<td>1.63</td>
<td>4.04</td>
</tr>
<tr>
<td>Rank LOF</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: LOF Results From NHL Data

3.1.1 Problems Identified

The problems identified with the Local Outlier Factor algorithm [BKN00] are:

1. The first problem identified is the computation of reachability distance for every object in p's neighbourhood. This problem involves computing k-distances of all objects in p's neighbourhood as well as distances between p and each of its neighbours and selecting the maximum in each case. The number of comparisons to be made before getting the local reachability distances of all objects is kN since every object has at least k-nearest neighbours, where N is the size of data and k is the minimum number of points within a neighbourhood (MinPts). The effect of the number of computations and comparisons becomes pronounced when N is large and thus reduces the running time of LOF.

2. The next problem is the computation of local reachability density for every object in p's neighbourhood before local outlier factor can be computed.

3. The final problem identified is the computation of local outlier factor (LOF) for all objects in the dataset before finding the outliers. The approach of computing local outlier factor for every object is not practical since outliers constitute only a small percentage of the entire dataset. Thus, computing LOF for all data objects before getting the very few needed outliers is not a desirable exercise. A solution to this has been proposed in [JTH01], but the proposed algorithm still relies on the computation of local reachability distances for computing limits on the set of
possible clusters that can possibly contain outliers. Clusters that can not possibly contain outliers are eliminated as soon as they are identified, which results in an improved performance for LOF.

3.1.2 The Proposed Solution
The above problems are addressed by proposing two algorithms Local Sparsity Coefficient (LSC) and Enhanced Local Sparsity Coefficient (ELSC) based on the original idea of local outliers. The two algorithms avoid the computation of reachability distances and local reachability densities, which are considered very expensive in the existing density-based algorithm. Instead, a local sparsity ratio derived from the neighbourhood distances is computed. Since the computation of reachability distance and local reachability density for every object consists of a huge number of comparisons, it is hoped that by removing such steps our algorithms will perform better than the existing Local Outlier Factor algorithm. In ELSC, data objects that can not possibly contain outliers are pruned (removed) based on their neighbourhood distances. The remaining data form the candidate set on which outliers are determined resulting in an improved performance over LSC and LOF since more than half of the entire data are removed as a result of the pruning.

3.2 The Local Sparsity Coefficient Algorithm (LSC)
The Local Sparsity Coefficient algorithm computes outliers based on the k-distance of an object and k-distance neighbourhood of p. Data objects are ranked depending on their local sparsity coefficients and the top-n objects with the highest local sparsity coefficients are declared outliers. The local sparsity coefficient is a measure that shows the concentration of objects within a specified neighbourhood distance. The neighbourhood of an object is more crowded if the local sparsity coefficient is low and sparse, otherwise. Data objects with high sparsity coefficients are more likely to be outliers whereas those with low values are not. In other words, a low local sparsity coefficient is an indication of a dense neighbourhood.
Definition 1: Local Sparsity Ratio of \( p \)

The local sparsity ratio of an object \( p \) denoted \( lsr_k(p) \) is defined as the ratio of the cardinality of \( k \)-distance neighbourhood of \( p \) to the sum of all the distances of objects in that neighbourhood.

\[
lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} \text{dist}_k(N_k(p))}, \quad \text{where dist}_k(N_k(p)) \text{ consists of actual distances of the objects in } k\text{-distance neighbourhood of } p. \]

The local sparsity ratio is a measure that indicates the concentration of objects around a particular object. A low local sparsity ratio indicates that the object has a high potential of being an outlier whereas a high value indicates that the object is not likely to be an outlier. The final declaration of outliers is based on the local sparsity coefficient rather than the local sparsity ratio.

Definition 2: Local Sparsity Coefficient of \( p \)

The local sparsity coefficient of \( p \) denoted \( LSC_k(p) \) is the average ratio of the local sparsity ratio of \( p \) to that of its \( k \)-nearest neighbours. The local sparsity coefficient is a measure that shows the dispersion of objects around a particular object.

\[
LSC_k(p) = \frac{\sum_{o \in N_k(p)} lsr_k(o)}{|N_k(p)|} \quad \text{lsr}_k(p)
\]

A high local sparsity coefficient indicates the neighbourhood around an object is not crowded and hence that object is more likely to be an outlier whereas a low local sparsity coefficient means the neighbourhood around the object is more crowded and hence very unlikely to be an outlier. Figure 7 shows the Local Sparsity Coefficient algorithm.
1. Compute $k$-distance($p$)
   i. Compute all distances from $p$
   ii. Select the first $k$-minimum distance from $p$
   iii. Select the maximum of the $k$-minimum distance as $k$-distance($p$)

2. Find $k$-distance neighbourhood of $p$ ($N_k(p)$)
   // the set of all objects whose distance are less than $k$-distance of $p$

3. Compute local sparsity ratio ($lsr(p)$)
   
   $$lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} dist_{\_\_}N_k(p)}$$

4. Compute local sparsity coefficient of $p$

   $$LSC_k(p) = \frac{\sum_{o \in N_k(p)} lsr_k(o)}{|N_k(p)|}$$

5. Rank LSC$_k(p)$ and declare the top $n$ as outliers

Figure 7: Local Sparsity Coefficient Algorithm

The sample NHL player statistics of 1995 is used to test LSC algorithm. The sample data are used to walk through the steps in the Local Sparsity Coefficient algorithm.

1. **Computing $k$-distance($p$)**

   We compute $k$-distance of each object in the database satisfying the following three conditions:

   1. Compute the distances of all objects from $p$.
   2. Select the $k$-distinct minimum distances.
   3. Select the maximum of the $k$-distinct minimum distance as $k$-distance of $p$.

   The $k$-distances of the objects have already been computed and the steps are the same as that of LOF.
2. Finding k-distance neighbourhood of p

The set of objects with distance from p that are not greater than \( k\)-distance\( (p) \) are noted. With the sample data, each object has four neighbours. The \( k\)-distance neighbourhood for \( P_1 \) are \( P_3, P_4, P_5, P_8 \) whereas those for \( P_2 \) are \( P_3, P_4, P_5, P_9 \). The \( k\)-distance neighbourhood for the remaining data objects are shown in Table 8.

3. Computing local sparsity ratio of p \( (lsr_k(p)) \)

The local sparsity ratio is the ratio of the absolute neighbourhood distance of \( p \) to the sum of all objects in the neighbourhood distance of \( p \) and it is given by:

\[
\text{lsr}_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} \text{dist}_o N_k(p)}.
\]

The local sparsity ratios of all objects within the data set are computed. For instance, using the sample data, the local sparsity ratio of \( P_1 \) is obtained as \( \text{lsr}_k(P_1) = 4/115.2 = 0.035 \). The local sparsity ratio of \( P_2 \) is \( \text{lsr}_k(P_2) = 4/93.6 = 0.043 \). The local sparsity ratios of the remaining data objects are obtained in the same way and shown in Table 9. It is worth noting that \( k\)-distance neighbourhood of all objects in the data set may not necessarily be the same as it happened to be in our sample data.

4. Computing local sparsity coefficient of p \( (LSC_k(p)) \)

The local sparsity coefficient of \( p \) is the average ratio of the local sparsity ratios of \( p \) to that of \( p \)'s nearest neighbours. The local sparsity coefficients for the objects in the sample data are computed. For instance,

\[
LSC_k(p_1) = \frac{\sum_{o \in N_k(p)} \text{lsr}_k(p_3) + \text{lsr}_k(p_4) + \text{lsr}_k(p_5) + \text{lsr}_k(p_8)}{|N_k(p_1)|}, \text{where}
\]

\( \text{lsr}_k(P_3), \text{lsr}_k(P_4), \text{lsr}_k(P_5) \) and \( \text{lsr}_k(P_8) \) are the local sparsity ratios of \( P_3, P_4, P_5 \) and \( P_8 \) respectively, which constitute the 4-nearest neighbours of \( P_1 \). Thus, \( LSC_k(P_1) = \{(0.104+ 0.116 + 0.134 + 0.051) / 0.035 \}/ 4 \} = 2.893 \). Also, the local sparsity coefficient of \( P_2 \) is \( LSC_k(P_2) = \{(\text{lsr}_k(P_3) + \text{lsr}_k(P_4) + \text{lsr}_k(P_5) + \text{lsr}_k(P_9)) / \text{lsr}_k(P_2)\}/ 4 \) where \( \text{lsr}_k(P_3), \text{lsr}_k(P_4), \text{lsr}_k(P_5) \) and \( \text{lsr}_k(P_9) \) are the local sparsity
ratios of $P_3$, $P_4$, $P_5$ and $P_9$ respectively, which constitute the 4-nearest neighbours of $P_2$. Hence, $LSC_k(P_2) = ((0.104 + 0.116 + 0.134 + 0.051) / 0.043) / 4) = 2.291$

Finally, the local sparsity coefficients are ranked and the top-$n$ objects with the highest $LSC$ values declared outliers. The top-$4$ outliers produced by $LSC$ algorithm are P10, P1, P9 and P8. Although, the two algorithms differ in some of the trailing outliers, it can still be concluded that $LSC$ is very accurate since in real life data outliers are not supposed to be more than 10 percent of the entire data. The entire results of the $LSC$ algorithm with the sample data is shown in Table 8.

<table>
<thead>
<tr>
<th>Players</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
<th>$P_8$</th>
<th>$P_9$</th>
<th>$P_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-distance ($p$)</td>
<td>32.3</td>
<td>24.1</td>
<td>14.7</td>
<td>13</td>
<td>11.4</td>
<td>14.3</td>
<td>14.7</td>
<td>22.8</td>
<td>28.4</td>
<td>80.6</td>
</tr>
<tr>
<td>$N_k(p)$</td>
<td>P3</td>
<td>P3</td>
<td>P4</td>
<td>P3</td>
<td>P3</td>
<td>P3</td>
<td>P3</td>
<td>P4</td>
<td>P2</td>
<td>P2</td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>P4</td>
<td>P5</td>
<td>P5</td>
<td>P4</td>
<td>P4</td>
<td>P4</td>
<td>P5</td>
<td>P5</td>
<td>P5</td>
</tr>
<tr>
<td></td>
<td>P5</td>
<td>P5</td>
<td>P6</td>
<td>P6</td>
<td>P6</td>
<td>P5</td>
<td>P6</td>
<td>P6</td>
<td>P6</td>
<td>P6</td>
</tr>
<tr>
<td></td>
<td>P8</td>
<td>P9</td>
<td>P7</td>
<td>P7</td>
<td>P7</td>
<td>P7</td>
<td>P7</td>
<td>P7</td>
<td>P7</td>
<td>P9</td>
</tr>
<tr>
<td>Dist-$N_k(p)$</td>
<td>32.3</td>
<td>24.1</td>
<td>5.7</td>
<td>5.7</td>
<td>11.2</td>
<td>4.5</td>
<td>14.7</td>
<td>20.9</td>
<td>28.4</td>
<td>63.4</td>
</tr>
<tr>
<td></td>
<td>28.6</td>
<td>22.9</td>
<td>3.6</td>
<td>3.7</td>
<td>11.4</td>
<td>14.3</td>
<td>12</td>
<td>22.8</td>
<td>22.4</td>
<td>55.1</td>
</tr>
<tr>
<td></td>
<td>25.5</td>
<td>23.6</td>
<td>14.3</td>
<td>13</td>
<td>3.6</td>
<td>13</td>
<td>11.4</td>
<td>19</td>
<td>26.3</td>
<td>77.3</td>
</tr>
<tr>
<td></td>
<td>28.8</td>
<td>23</td>
<td>14.7</td>
<td>12</td>
<td>3.7</td>
<td>11.2</td>
<td>4.5</td>
<td>15.8</td>
<td>24.1</td>
<td>80.6</td>
</tr>
<tr>
<td>$\Sigma$dist-$N_k(p)$</td>
<td>115.2</td>
<td>93.6</td>
<td>38.3</td>
<td>34.4</td>
<td>29.9</td>
<td>43</td>
<td>42.6</td>
<td>78.5</td>
<td>101.2</td>
<td>276.4</td>
</tr>
<tr>
<td>$LSR_p(p)$</td>
<td>0.035</td>
<td>0.043</td>
<td>0.104</td>
<td>0.116</td>
<td>0.134</td>
<td>0.093</td>
<td>0.094</td>
<td>0.051</td>
<td>0.04</td>
<td>0.014</td>
</tr>
<tr>
<td>$LSC_k(p)$</td>
<td>2.893</td>
<td>2.291</td>
<td>1.05</td>
<td>0.916</td>
<td>0.759</td>
<td>1.204</td>
<td>1.189</td>
<td>2.142</td>
<td>2.375</td>
<td>5.536</td>
</tr>
<tr>
<td>Rank of $LSC$</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8: Results of Applying $LSC$ to NHL Player Statistics Data.

3.3 **Enhanced Local Sparsity Coefficient Algorithm (ELSC)**

Outliers generally form a small percentage of the entire data and as a result computing local sparsity coefficients ($LSC$ values) for all data objects before declaring the top $n$ objects with the highest $LSC$ values as outliers would not be very efficient. In view of this, a limit on the local sparsity ratio is computed based on the entire neighbourhood distances. Data objects which have local sparsity ratios greater than the computed limit cannot possibly be outliers and are removed as soon as they are identified. The remaining data objects that are possible outlier candidates constitute the candidate set (candset). The
local sparsity coefficient \((LSC)\) is computed for only data objects in the candidate set and the top \(n\) declared outliers. The majority of non-outlier data objects are pruned by our algorithm resulting in reduced number of \(LSC\) computations which further enhances the performance of the algorithm. The main steps of the Enhanced Local Sparsity algorithm are given in Figure 8.

1. Compute \(k\)-distance\((p)\)
   i. Compute all distances from \(p\)
   ii. Select the first \(k\)-minimum distance from \(p\)
   iii. Select the maximum of the \(k\)-minimum distance as \(k\)-distance\((p)\)

2. Find \(k\)-neighbourhood distance of \(p\) \((N_k(p))\)

3. Compute the local sparsity ratio of \(p\) \((lsr_k(p))\)

\[
lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} dist \_N_k(p)}
\]

4. Compute the pruning factor \((Pf)\)
   // the minimum \(k\)-neighbourhood distance from data
   \(N_k(p).num: =\) total number of neighbours for data
   \(N_k(p).sum: =\) sum of all \(N_k(p)\)
   \(N_k(p).mean = N_k(p).sum / N_k(p).num\)
   // take the reciprocal to get the minimum bound on \(Lsr_k(p)\)
   \(Pf = 1 / N_k(p).mean\)

   If \(Lsr_k(p) < lsr\_bound\) // for any object
     Add object to Candidate set (Candset) // potential outlier
   Else
     Delete object from list
   End if.

5. Compute local sparsity coefficient of \(p\) for object in Candset

\[
LSC_k(p) = \frac{\sum_{o \in N_k(p)} lsr_k(o)}{|N_k(p)|}
\]

6. Rank \(LSC_k(p)\)

7. Declare the top \(n\) objects as outliers.

Figure 8: Enhanced Local Sparsity Coefficient Algorithm
The ELSC algorithm is applied to the sample NHL player statistic data shown in Table 5. The reason is to verify if ELSC gives the same results as LSC and LOF. We follow the steps in the algorithm to get the output.

1. **Computing k-distance(p)**

   The k-distances of the objects in the database are obtained as discussed in the earlier sections.

2. **Finding k-neighbourhood distance of p (N_k(p))**

   The k-distance neighbourhoods are obtained in the same way as discussed in the proceeding sections but this time we note the actual neighbourhood distances. For \( P_1 \), the k-nearest neighbours are \( \{P_1, P_4, P_5, P_8\} \) and their respective distances are \( \{32.3, 28.6, 25.5, 28.8\} \). Also, \( P_2 \) nearest neighbours are \( \{P_3, P_4, P_5, P_6\} \) with neighbourhood distances \( \{24.1, 22.9, 23.6, 23\} \). The neighbourhood distances of the other objects are obtained as described above.

3. **Computing local sparsity ratio for objects**

   The ratios of the absolute value of the nearest-neighbours of \( p \) to k-distance neighbourhood of \( p \) are then computed for the objects as follows:

   \[
   ls_{rk}(p) = \frac{\sum_{o \in N_k(p)} |N_k(p)|}{\text{dist}_N_k(p)}, \text{ where } |N_k(p)| \text{ is the number of objects in } p \text{ and } \text{dist}_N_k(p) \text{ consists of actual distances of the objects in } N_k(p).
   \]

   Using the neighbourhood distances shown in Table 8, the local sparsity ratios are computed as follows:

   \( Ls_{rk}(P_1) = 4/115.2 = 0.035, Ls_{rk}(P_2) = 4/93.6 = 0.043, Ls_{rk}(P_3) = 4/38 = 0.104, \)
   \( Ls_{rk}(P_4) = 4/34.4 = 0.116, Ls_{rk}(P_5) = 4/29.9 = 0.134, Ls_{rk}(P_6) = 4/43 = 0.093, \)
   \( Ls_{rk}(P_7) = 4/42.6 = 0.094, Ls_{rk}(P_8) = 4/78.5 = 0.051, Ls_{rk}(P_9) = 4/101.2 = 0.04 \) and \( Ls_{rk}(P_{10}) = 4/276.4 = 0.014. \)
4. Computing the pruning factor (Pf)

The pruning factor (Pf) is first computed and subsequently used to prune the entire data. The pruning factor is based on the assumption that the local sparsity ratio of an object p in a dataset should not be less than that of the entire data if that object is not an outlier. The pruning factor gives an upper bound for any object that is an outlier candidate. It is computed using the sample data.

\[ N_k(p)_{\text{mean}} = N_k(p)_{\text{sum}} / N_k(p)_{\text{num}}, \]
where \( N_k(p)_{\text{mean}}, N_k(p)_{\text{sum}}, N_k(p)_{\text{num}} \)
are the mean, sum and the number of neighbourhood distances respectively.

\[ N_k(p)_{\text{mean}} = 853.4 / 40 = 21.3275. \]
The pruning factor (Pf) is the inverse of \( N_k(p)_{\text{mean}}. \) Thus, \( Pf = 1 / 21.3275 = 0.04685. \) Applying the pruning factor to the sample results obtained in 3 above, only data objects \( P_2, P_{10}, P_1 \) and \( P_9 \) qualify as outlier candidates. Since \( P_1, P_2, P_9 \) and \( P_{10} \) are the only data objects with local sparsity ratios less than 0.04685. The candidate set = \{P_1, P_9, P_2 and P_{10}\} is tested further for outliers but not the entire dataset.

5. Computing \( LSC_k(p) \) for objects in the candidate set

The local sparsity coefficients are computed for objects in the candidate set. The candidate set for the sample data is \{P_1, P_9, P_2 and P_{10}\}. The local sparsity coefficients are calculated as shown below:

\[ LSC_k(P_1) = \frac{(l_{sr}(P_3) + l_{sr}(P_4) + l_{sr}(P_5) + l_{sr}(P_9))}{l_{sr}(P_1)} / 4 \]
\[ = \frac{(0.104 + 0.116 + 0.134 + 0.051)}{0.035} / 4. \]
\[ = 2.893 \]

\[ LSC_k(P_2) = \frac{(l_{sr}(P_3) + l_{sr}(P_4) + l_{sr}(P_5) + l_{sr}(P_9))}{l_{sr}(P_2)} / 4 \]
\[ = \frac{(0.104 + 0.116 + 0.134 + 0.04)}{0.043} / 4. \]
\[ = 2.291 \]

\[ LSC_k(P_9) = \frac{(l_{sr}(P_2) + l_{sr}(P_3) + l_{sr}(P_6) + l_{sr}(P_{10}))}{l_{sr}(P_9)} / 4 \]
\[ = \frac{(0.043 + 0.134 + 0.093 + 0.094)}{0.04}/4 \]
\[ = 2.375 \]

\[ LSC_k(P_{10}) = \frac{(l_{sr}(P_2) + l_{sr}(P_3) + l_{sr}(P_4) + l_{sr}(P_9))}{l_{sr}(P_{10})} / 4 \]
\[ = \frac{((0.043 + 0.104 + 0.166 + 0.093)}{0.014}/4 \]
\[ = 5.536 \]
6. Ranking $LSC_k(p)$ and picking top $n$ as outliers

Finally, the computed local sparsity coefficients ($LSC_k(p)$) are ranked and the top $n$ objects with the highest values are declared outliers. Clearly, we have our outliers in increasing order of outlying as $P_2, P_1, P_9$ and $P_{10}$. The same results were obtained using the previous algorithms ($LSC$ and $LOF$).

3.4 Defining Bounds on Local Sparsity Ratio and Local Sparsity Coefficient

In this section, we define the acceptable range of values for local sparsity coefficient and local sparsity ratio of an object $p$. By definition local sparsity coefficient of $p$ ($LSC_k(p)$) is defined in relation (1), where all the symbols have their usual meanings.

$$LSC_k(p) = \frac{\sum_{o \in N_k(p)} lsr_k(o)}{|N_k(p)|} \quad (1)$$

Let $lsr_k(o).lower$ be the lower bound on $lsr_k(o)$, then by taking the lower bound on the local sparsity ratio of relation (1), we obtain relation (2).

$$LSC_k(p) > \frac{lsr_k(o).lower}{|N_k(p)|} \quad (2)$$

Also, if $lsr_k(o).upper$ is the upper bound on $lrs_k(o)$, then by taking the upper bound on the local sparsity coefficient of relation (1), we get relation (3).

$$LSC_k(p) > \frac{lsr_k(o).upper}{|N_k(p)|} \quad (3)$$

Since $|N_k(p)|$ is a positive constant, combining relation (1) and relation (2), we obtain relation (4) as follows:

$$\frac{lsr_k(o).upper}{lsr_k(p)} < LSC_k(p) < \frac{lsr_k(o).lower}{lsr_k(p)} \quad (4)$$

Hence, relation (4) defines the upper and lower bounds on local sparsity coefficient in terms of the local sparsity ratios.
Similarly,
\[ lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} dist_{-N_k(p)}} \],
where \(|N_k(p)|\) is the cardinality of the
neighbourhood distance of p and \(dist_{-N_k(p)}\) contains the actual distances of objects
within the neighbourhood of p. Since \(lsr_k(p)\) is a ratio of the absolute value of
the neighbourhood distance to the sum of the individual elements in that neighbourhood. The
lower and upper bounds on \(lsr_k(p)\) is obtained by taking the maximum and minimum of
the actual distances of the objects within the neighbourhood of p. Thus, \(lsr_k(p)\) is
bounded by:
\[ \frac{1}{\max(dist_{-N_k(p)})} < lsr_k(p) < \frac{1}{\min(dist_{-N_k(p)})} . \]

3.5 Comparing LSC and EULSC with LOF
The comparison of Local Sparsity Coefficient and Enhanced Local Sparsity Coefficient
algorithms with Local Outlier algorithm is done based on how the algorithms are
implemented. We did not make use of either indexing or materialization, instead we
recomputed the neighbourhood distances whenever they were needed. In some cases,
specific distances were also recomputed when needed. In comparing the three algorithms,
we consider only the most dominant steps such as:

1. Computing the distance of every object from each other results in \((n/2) (n-1)\)
   number of comparisons, yielding a time complexity of \(O(n^2)\), where \(n\) is the size
   of data. This is because each record is paired with every other record in the data
   set before the entire distances are obtained. The time complexity for a data of size
   \(n\) with k-dimension is thus \(O(kn^2)\). The distance computation is done by all the
   three algorithms.

2. The next major task is the computation of reachability distance, which compares
   and selects the maximum of the k-distance of \(o\) and the distance between \(p\) and \(o\),
   where \(p\) and \(o\) are objects in a given dataset. This computation results in at least
   \(k \times n\) number of comparisons with a time complexity of \(O(kn)\), where \(k\) is the
   minimum number of neighbours of each record and \(n\) is the size of data. Since \(k\) is
supposed to be very large for all density-based algorithms, this step can be very expensive when \( n \) is also large. The reachability distance computation is done for only the Local Outlier Factor algorithm.

3. The last most important activity is the sorting/ranking of outliers. This step has a time complexity of \( O(n\log n) \) where \( n \) is the size of data.

The dominant cost is \( O(kn^2) \) for all the three algorithms where \( n \) is the size of data and \( k \) is the number of dimensions, thus, all the three algorithms have the same time complexity of \( O(kn^2) \).

The Local Sparsity Coefficient and Enhanced Local Sparsity Coefficient algorithms still perform better than the Local Outlier Factor algorithm with respect to response time even though they all have a quadratic time complexity. The reason is that in LSC and ELSC the computations and comparisons involved in obtaining reachability distances and reachability densities are avoided. Similarly, ELSC performs better than LSC because in ELSC, the computation of local sparsity coefficient is done for only data objects that are potential outlier candidates and not for the entire data.
4. EXPERIMENTAL EVALUATION

This chapter presents the performance analysis of Local Sparsity Coefficient (LSC), Enhanced Local Sparsity Coefficient (ELSC) and Local Outlier Factor (LOF) algorithms. All the experiments were performed on a 1.8GHz Intel Pentium 4 PC with 256 megabytes main memory, running Windows 2000 Professional Edition. The programs are written in Java. The data sets were stored in text files and retrieved using Java file processing methods.

Two sets of data were used to test the algorithms for correctness and execution (response) time. The one for testing correctness is obtained from the National Hockey League (NHL) player statistics for 1995 whereas the response time dataset was generated using a simple code. The set up of the experiments and the results obtained are discussed in subsequent sections.

4.1 Testing for Correctness

The National Hockey League (NHL) player statistics of 1995 used for this experiment had 805 records with unknown number of outliers. Four fields (name, total score, plus/minus, and penalty minutes) were chosen from the entire statistics to constitute our data set. In addition, each record was assigned a unique ID purposely for evaluating the results. This experiment was to determine how many of the outliers determined by LOF are also correctly identified by LSC and ELSC. In particular, we were interested in the number of top-n records determined as outliers, as well as their positional rankings in the three algorithms. The experimental setup requires the minimum number of neighbours (MinPts) to be considered in determining outliers as well as the top-n outliers needed. The three algorithms were run using different MinPts of 50, 100 and 200 with top-n outliers set to 100. Next, we discuss the results of the experimental runs.

MinPts of 50

The results obtained from running the three algorithms on the test data with MinPts of 50 are presented in Table 9. The results show that all the three algorithms identified the
same records for 8 out of the top 10 outliers. In addition, all the three algorithms agreed on the same positional rankings for the top 7 outliers. The strongest outlier is record with Record# 97 with LOF of 10.798 and LSC of 15.712 for LOF and LSC/ELSC algorithms respectively. Also LSC and ELSC produced the same number of outliers.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Rec. No.</th>
<th>LOF</th>
<th>Rec. No.</th>
<th>LSC</th>
<th>Rec. No.</th>
<th>ELSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97</td>
<td>10.798</td>
<td>97</td>
<td>15.712</td>
<td>97</td>
<td>15.712</td>
</tr>
<tr>
<td>2</td>
<td>547</td>
<td>7.111</td>
<td>547</td>
<td>10.314</td>
<td>547</td>
<td>10.314</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>5.548</td>
<td>16</td>
<td>6.504</td>
<td>16</td>
<td>6.504</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>4.435</td>
<td>43</td>
<td>6.504</td>
<td>43</td>
<td>6.504</td>
</tr>
<tr>
<td>5</td>
<td>223</td>
<td>3.18</td>
<td>223</td>
<td>4.64</td>
<td>223</td>
<td>4.64</td>
</tr>
<tr>
<td>6</td>
<td>576</td>
<td>2.78</td>
<td>576</td>
<td>3.604</td>
<td>576</td>
<td>3.604</td>
</tr>
<tr>
<td>7</td>
<td>241</td>
<td>2.609</td>
<td>241</td>
<td>3.51</td>
<td>241</td>
<td>3.51</td>
</tr>
<tr>
<td>8</td>
<td>23</td>
<td>2.148</td>
<td>441</td>
<td>3.068</td>
<td>441</td>
<td>3.068</td>
</tr>
<tr>
<td>9</td>
<td>247</td>
<td>1.946</td>
<td>23</td>
<td>2.897</td>
<td>23</td>
<td>2.897</td>
</tr>
<tr>
<td>10</td>
<td>640</td>
<td>1.88</td>
<td>366</td>
<td>2.455</td>
<td>366</td>
<td>2.455</td>
</tr>
</tbody>
</table>

Table 9: Experimental Result on Test Data with MinPts = 50

**MinPts of 100**

Table 10 shows the results from running the algorithms on the test data with MinPts of 100. The three algorithms agreed on the same number of records for the top 10 outliers. They also agreed on the same positional rankings for 8 out of the 10 outliers. Two outliers were however interchanged in their positional rankings. LOF identified records with Record# 43 and Record# 16 as the 3rd and 4th strongest outliers respectively, while LSC/ELSC identified Record# 43 and 16 as the 4th and 3rd strongest outliers respectively. The strongest outlier was Record# 97 and Record# 676 as the weakest.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Rec. No.</th>
<th>LOF</th>
<th>Rec. No.</th>
<th>LSC</th>
<th>Rec. No.</th>
<th>ELSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97</td>
<td>10.598</td>
<td>97</td>
<td>15.173</td>
<td>97</td>
<td>15.173</td>
</tr>
<tr>
<td>2</td>
<td>547</td>
<td>6.977</td>
<td>547</td>
<td>10.0</td>
<td>547</td>
<td>10.0</td>
</tr>
<tr>
<td>3</td>
<td>43</td>
<td>5.394</td>
<td>16</td>
<td>7.631</td>
<td>16</td>
<td>7.631</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>5.35</td>
<td>43</td>
<td>6.143</td>
<td>43</td>
<td>6.143</td>
</tr>
<tr>
<td>5</td>
<td>223</td>
<td>3.547</td>
<td>223</td>
<td>5.066</td>
<td>223</td>
<td>5.066</td>
</tr>
<tr>
<td>6</td>
<td>576</td>
<td>3.157</td>
<td>576</td>
<td>4.5</td>
<td>576</td>
<td>4.5</td>
</tr>
<tr>
<td>7</td>
<td>241</td>
<td>2.738</td>
<td>241</td>
<td>3.943</td>
<td>241</td>
<td>3.943</td>
</tr>
<tr>
<td>8</td>
<td>23</td>
<td>2.439</td>
<td>23</td>
<td>3.469</td>
<td>23</td>
<td>3.469</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>2.232</td>
<td>93</td>
<td>2.697</td>
<td>93</td>
<td>2.697</td>
</tr>
<tr>
<td>10</td>
<td>676</td>
<td>2.202</td>
<td>676</td>
<td>2.659</td>
<td>676</td>
<td>2.659</td>
</tr>
</tbody>
</table>

Table 10: Experimental Result on Test Data with MinPts = 100

MinPts of 200

Finally, the results from running the three algorithms on the test data with MinPts of 200 are shown in Table 11.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Rec. No.</th>
<th>LOF</th>
<th>Rec. No.</th>
<th>LSC</th>
<th>Rec. No.</th>
<th>ELSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97</td>
<td>10.711</td>
<td>97</td>
<td>15.065</td>
<td>97</td>
<td>15.065</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>7.113</td>
<td>16</td>
<td>9.91</td>
<td>16</td>
<td>9.91</td>
</tr>
<tr>
<td>3</td>
<td>547</td>
<td>7.077</td>
<td>547</td>
<td>7.49</td>
<td>547</td>
<td>7.49</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>5.344</td>
<td>43</td>
<td>7.481</td>
<td>43</td>
<td>7.481</td>
</tr>
<tr>
<td>5</td>
<td>223</td>
<td>4.212</td>
<td>223</td>
<td>5.887</td>
<td>223</td>
<td>5.887</td>
</tr>
<tr>
<td>6</td>
<td>241</td>
<td>3.621</td>
<td>241</td>
<td>5.074</td>
<td>241</td>
<td>5.074</td>
</tr>
<tr>
<td>7</td>
<td>576</td>
<td>3.605</td>
<td>576</td>
<td>4.393</td>
<td>576</td>
<td>4.393</td>
</tr>
<tr>
<td>8</td>
<td>23</td>
<td>3.063</td>
<td>23</td>
<td>3.761</td>
<td>23</td>
<td>3.761</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>2.703</td>
<td>93</td>
<td>3.393</td>
<td>93</td>
<td>3.393</td>
</tr>
<tr>
<td>10</td>
<td>676</td>
<td>2.651</td>
<td>676</td>
<td>3.329</td>
<td>676</td>
<td>3.329</td>
</tr>
</tbody>
</table>

Table 11: Experimental Result on Test Data with MinPts = 200
The three algorithms produced the same outliers and the same positional rankings for all the outliers. The strongest outlier is Record# 97 with LOF of 10.711 and LSC of 15.065. The weakest outlier is Record# 676. Since all the three algorithms identified the same outliers for different neighbourhood distances, we conclude that the proposed algorithms are correct. This experiment also confirms that the algorithms achieve better and more accurate results when the neighbourhood distance is large.

### 4.2 Testing for Response Time

The data used for testing response time was generated using a simple code. The generated data set contained 16,000 records with structure similar to the NHL player statistics data used for the analysis in section 4.1. The data set contained unknown number of outliers. Two sets of experiments were conducted for response time. The first was to find the effect of MinPts on response time while the other was to ascertain the effect of data size on response time for the three algorithms. The execution times for each of the three algorithms were recorded. The results of the experimental runs are discussed in subsequent sections.

#### 4.2.1 Effect of MinPts on Response Time

This experiment uses 10,000 out of the 16,000 generated records. The experimental setup requires the user to provide MinPts and the number of top-n outliers needed. The test was conducted using 8 different MinPts 50, 100, 200, 300, 400, 500, 600 and 700. The number of top-n outliers needed was set to 100 in each case. The times taken for each of the algorithms to produce the needed outliers were recorded for each neighbourhood distance. The results obtained are presented in Table 12 and represented graphically in Figure 9.

It can be observed from Figure 9 that ELSC and LSC perform better than LOF with ELSC having the least execution time for different MinPts. The execution time of ELSC increases slightly with increasing MinPts whereas LSC has almost a constant execution time with increasing MinPts. The execution time of LOF increases rapidly with
increasing MinPts owing to the large number of comparisons that are made during the computation of reachability distances.

<table>
<thead>
<tr>
<th>MinPts</th>
<th>LOF</th>
<th>LSC</th>
<th>ELSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>4145</td>
<td>4100</td>
<td>3390</td>
</tr>
<tr>
<td>100</td>
<td>4215</td>
<td>4100</td>
<td>3490</td>
</tr>
<tr>
<td>200</td>
<td>4275</td>
<td>4090</td>
<td>3587</td>
</tr>
<tr>
<td>300</td>
<td>4415</td>
<td>4100</td>
<td>3599</td>
</tr>
<tr>
<td>400</td>
<td>4600</td>
<td>4090</td>
<td>3620</td>
</tr>
<tr>
<td>500</td>
<td>4740</td>
<td>4100</td>
<td>3620</td>
</tr>
<tr>
<td>600</td>
<td>4966</td>
<td>4060</td>
<td>3600</td>
</tr>
<tr>
<td>700</td>
<td>5405</td>
<td>4050</td>
<td>3650</td>
</tr>
</tbody>
</table>

Table 12: Effect of MinPts on Response Time

Figure 9: Effect of MinPts on Response Time
4.2.2 Effect of Data Size on Response Time

This test uses all the 16,000 generated records. The test was conducted separately for two different MinPts 50 and 500 with data size varying from 2,000 to 16,000. The rationale was to find out how the algorithms behave with smaller and larger MinPts as data size increases. The number of top-n outliers needed was set to 100 in each case. The time taken for each algorithm to produce the needed results on the test data were noted for the different MinPts.

MinPts of 50

The results from running the algorithms on the test data with MinPts of 50 are shown in Table 13 and depicted in Figure 10. The experimental results show no difference in execution (response) time between LOF and LSC irrespective of the size of data when MinPts is small as depicted in Figure 9. ELSC however, shows a remarkable difference from LOF and LSC. The difference becomes pronounced with increasing data size. ELSC thus demonstrates superiority over LOF and LSC for all data sizes.

<table>
<thead>
<tr>
<th>No. of Records</th>
<th>Local Outlier Factor (LOF)</th>
<th>Local Sparsity Coefficient (LSC)</th>
<th>Enhanced Local Sparsity Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2k</td>
<td>165</td>
<td>160</td>
<td>132</td>
</tr>
<tr>
<td>4k</td>
<td>768</td>
<td>760</td>
<td>620</td>
</tr>
<tr>
<td>6k</td>
<td>1550</td>
<td>1520</td>
<td>1275</td>
</tr>
<tr>
<td>8k</td>
<td>2630</td>
<td>2620</td>
<td>2055</td>
</tr>
<tr>
<td>10k</td>
<td>4035</td>
<td>3915</td>
<td>3370</td>
</tr>
<tr>
<td>12k</td>
<td>5600</td>
<td>5530</td>
<td>4800</td>
</tr>
<tr>
<td>14k</td>
<td>7745</td>
<td>7710</td>
<td>6360</td>
</tr>
<tr>
<td>16k</td>
<td>10350</td>
<td>10285</td>
<td>8750</td>
</tr>
</tbody>
</table>

Table 13: Effect of Data Size on Response Time (MinPts =50)
Figure 10: Effect of Data Size on Response Time (MinPts = 500)

MinPts of 500

The results from running the algorithms on the test data with 16000 records and a MinPts of 500 are shown in Table 14 and represented graphically in Figure 11.

<table>
<thead>
<tr>
<th>No. of Records</th>
<th>Local Outlier Factor (LOF)</th>
<th>Local Sparsity Coefficient (LSC)</th>
<th>Enhanced Local Sparsity Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2k</td>
<td>212</td>
<td>222</td>
<td>203</td>
</tr>
<tr>
<td>4k</td>
<td>785</td>
<td>760</td>
<td>545</td>
</tr>
<tr>
<td>6k</td>
<td>1830</td>
<td>1475</td>
<td>1260</td>
</tr>
<tr>
<td>8k</td>
<td>3020</td>
<td>2630</td>
<td>2305</td>
</tr>
<tr>
<td>10k</td>
<td>4760</td>
<td>3942</td>
<td>3415</td>
</tr>
<tr>
<td>12k</td>
<td>6735</td>
<td>5733</td>
<td>5150</td>
</tr>
<tr>
<td>14k</td>
<td>9243</td>
<td>7862</td>
<td>7263</td>
</tr>
<tr>
<td>16k</td>
<td>12150</td>
<td>10605</td>
<td>10000</td>
</tr>
</tbody>
</table>

Table 14: Effect of Data Size on Response Time (MinPts = 500)
The algorithms do not show much difference in execution time for smaller data sizes, but there are remarkable differences as the size of data increases. Once again, ELSC has the least execution time for all data sizes. The execution time of LOF increases rapidly with increasing data size whereas LSC and ELSC show steady increases with increasing data size. In addition, ELSC is, thus, very practicable irrespective of size of data or MinPts. ELSC performs better than LOF and LSC under all conditions.

Figure 11: Effect of Data Size on Response Time (MinPts = 500)
5. CONCLUSIONS AND FUTURE WORK

Outlier mining is very important data mining activity which has not received much attention in the research community. But, finding rare activities such as detecting credit card fraud or cellular phone cloning is likely to be more interesting than finding how often a regular customer visits the ATM. Density-based approach to outlier-mining makes monitoring customer activities even more interesting since every customer activity has a potential of being exceptional.

This thesis contributes an enhancement to the Local Outlier Factor (LOF) algorithm called Local Sparsity Coefficient (LSC) algorithm. LSC improves upon the response time of LOF by eliminating the computation of reachability distances and local reachability density and replacing them with local sparsity ratio, which is less expensive. Another contribution of this thesis is Enhanced Local Sparsity Coefficient (ELSC) algorithm, which is a further enhancement to LSC. Since outliers constitute a small percentage of the entire data, ELSC prunes (removes) data objects that are not potential outlier candidates based on their neighbourhood distances as soon as they are detected. This process further reduces the number of possible computations resulting in additional improvement on the response time.

Experimental results show that ELSC performs better than LOF and LSC with respect to response time for all sizes of data. The three algorithms (LOF, LSC and ELSC) do not show any significant difference in response time for smaller MinPts. However, there is a remarkable difference in response time when MinPts is large. ELSC has the least response time followed by LSC with LOF having the worse response time. This order is true irrespective of data size and MinPts value. In terms of correctness, all the three algorithms identified the same records for the top-n with slight differences in positional ranking for the identified outliers for smaller MinPts. However, with larger values of MinPts, all the three algorithms identified the same record for with the same positional rankings for all the identified outliers. Areas of future research include finding what fraction of the entire data should be assigned to MinPts, running the proposed algorithms
on clusters of data to establish the performance of the algorithms on different clusters. Another area of interest includes investigating the possibility of applying the proposed algorithms together with some of the algorithms discussed in the literature to text data.
REFERENCES


[CLP+00] Cahill M.H., Lambert D., Pinheiro J.C., Sun D.S. "Detecting Fraud in the Real World." A technical Report, Bell Labs, Lucent technologies, 2000

2nd Int. Conf. on knowledge Discovery and Data Mining, Portland, OR 1996, pp 226-231


[Keywords] discordancy test, normal distribution, outliers, test statistic, p-value, deviation, chi square distribution, regression, multivariate outliers.

[HC00] Hung E., Cheung D.W.: "Parallel Algorithms for Mining Outliers in Large Databases" In Proc. of the 4th PAKDD 2000, Kyoto, Japan

[HLL+00] Hussain F., Liu H., Suzuki E., Lu H “Exception Rule Mining with a Relative Interestingness Measure” Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2000


[JTH01] Jin W., Tung A.K.H, Han J. “Mining Top-n Local Outliers in Large Databases” KDD 2001 San Francisco, Callifornia USA.


[MRA95] Mehta M., Rissanen J., Agrawal A. “MDL-based Decision Tree Pruning” In Intl. Conf. on Knowledge Discovery and Data Mining (KDD-95) Montreal, Canada, August 1995


[Jus01] Jussi T. "Outliers in Non-linear Time Series Econometrics" PhD Dissertation, June 2001 (University of Turkey)


[Van01] Van Dijk D. J. C. "Smooth Transition Models: Extensions and Outlier Robust Inference" PhD Dissertation, June 2001 (Erasmus University of Rotterdam)


VITA AUCTORIS

NAME       Malik Agyemang

YEAR OF BIRTH  1972

PLACE OF BIRTH  Kumasi, Ghana

EDUCATION

M.Sc., Computer Science
University of Windsor
Windsor, Ontario
Canada
2001 – 2002

B.Sc., (Second Class Upper Division Honors)
Computer Science and Statistics
University of Ghana, Legon
Accra
Ghana
1993 – 1997