B.Sc. in Computer Science and Engineering Thesis

Document Driven Decision Support System: A Practitioner's Approach

Submitted by

Sharmin Rahman
ID#200914056

Sharmin Islam
ID#200914015

Nusrat Sharmin
ID#200914057

Supervised by

Dr. Mohammad Lutfur Rahman
Asst. Professor, Faculty of Technical & Engineering Studies
Bangladesh University of Professional (BUP)

Department of Computer Science and Engineering
Military Institute of Science and Technology
Dhaka-1212, Bangladesh

DECEMBER 2012

CERTIFICATION

This thesis paper titled "Document Driven Decision Support System: A Practitioner's Approach"is submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering on December 2012.

Group Members:

Sharmin Rahman
Sharmin Islam
Nusrat Sharmin

Supervisor:

Dr. Mohammad Lutfur Rahman
Asst. Professor
Faculty of Technical & Engineering studies
Bangladesh University of Professional (BUP)
Dhaka-1216, Bangladesh.

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis paper is the outcome of the investigation and innovation carried out by the following students under the supervision of Dr. Mohammad Lutfur Rahman, Asst. Professor, Faculty of Technical & Engineering studies Bangladesh University of Professional (BUP), Dhaka, Bangladesh.

It is also declared that neither of this thesis paper nor any part of this has been submitted anywhere else for the award of any degree, diploma or other qualification.

_____

Sharmin Rahman
ID#200914056

_____

Sharmin Islam
ID#200914015

_____

Nusrat Sharmin
ID#200914057

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor Dr. Mohammad Lutfur Rahman, Asst. Professor, Faculty of Technical & Engineering studies Bangladesh University of Professional (BUP), Dhaka, Bangladesh.for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was great help in completing thesis.

We are especially grateful to the Department of computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing all out academic support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Dhaka
December 2012

Sharmin Rahman
Sharmin Islam
Nusrat Sharmin

ABSTRACT

Decision Support Systems (DSS) are computerized information system that helps decision makers with decision-making activities. DSS are interactive computer-based system that uses data, document, communication technologies, knowledge and model to support decision making process. In this thesis, we consider top two data mining algorithms in the research community: k-means and k-nearest neighbor (kNN). Given $C = \{c_1, ..., c_m\}$ is a set of pre-defined categories, an initial corpus $C_o = \{d_1, ..., d_s\}$ of documents previously categorized under the same set of categories and a training set $T_r = \{d_1, ..., d_g\}$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced and $D = \{\overline{d_1}, ..., \overline{d_n}\}$ is a set of documents to be categorized. The problem is to assign a value from $\{0, 1\}$ to each entry, a_{ij} where $1 \leq i \leq m$, $1 \leq j \leq n$ of the *decision matrix*. A value of 1 for a_{ij} is interpreted as a decision to file d_j under c_i, while a value of 0 is interpreted as a decision not to file d_j under c. A test set $T_e = d_{g+1}, ..., d_s$ will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in T_e will be fed to the classifiers and a measure of classification effectiveness will be based on how often the values for the a_{ij}'s obtained by the classifiers match the values for the ca_{ij}'s provided by the experts where ca_{ij} is the element of correct decision matrix and $1 \leq i \leq m$, $1 \leq j \leq s$. We implemented these two algorithms and perform cross-validation test to measure accuracy of them. It comes out that; kNN is more accurate than that of k-mean. Then we develop tracing of document-driven DSS to provide an explanation to improve the acceptance of decision makers, because decisions are based on both the inheritance among documents and acceptance of those advices for decision makers. So, we develop a tracing on the contents and the classification of interrelated documents to improve the explanation.

Keywords: Document-driven DSS, Document Categorization, Document Lineage, Explanation.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

DSS: Decision Support System

LS: Language System

PPS: Problem Process System

KS: Knowledge System

RS: Reasoning System

MBMS: Model Base Management System

KBMS: Knowledge Base Management System

DBMS: Data Base management system

LIST OF SYMBOLS

E : set of entities to be clustered

K : number of clusters

C : set of cluster centroids

$(\bar{x}, y)$: test object

(x, y) : training object

CHAPTER 1
INTRODUCTION

In 1970s, DSS was brought forward first by American scientist Scott Morton. And it had a great development in 1980s. A new subject and study has been produced. Its functions are the integration of making use of mass data, the organization of many patterns and the realization of scientific decision by assistance of all directors through man-machine alteration system. Intelligentizing is one of the developmental directions of the study of DSS.

Data mining is knowledge discovery in database. It is a non-common process that effective, novel and valuable data have been mined out among the mass data. It is that data mining is the mining of knowledge among the mass data.

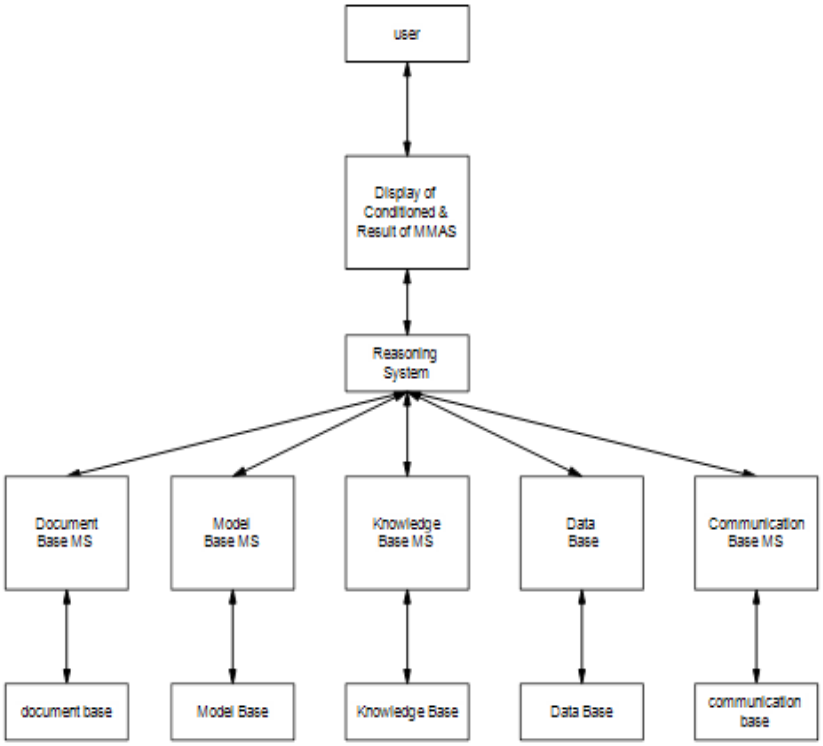

Figure 1.1: Configuration Block Drawing of DSS

As shown in figure 1.1, it is the general configuration of DSS. There are three departments of DSS. They are language system (LS), problem process system (PPS) and knowledge

system (KS). Problem process system is composed of reasoning system (RS), model base management system (MBMS), knowledge base management system (KBMS) and data base management system (DBMS). Knowledge system is composed of document, model base (MB), knowledge base (KB), data base (DB) and communication.

Forest Research [1] predicts that unstructured data, such as text, will become the predominant data type stored online. Also Gartner group [22] predicts that the amount of textual information double in every month. The data grows too fast. It has exceeded human capacity to retrieve and utilize. This is a problem that today's decision makers have to face. In order to get rid of the plight of this information explosion, decision makers often use decision support system (DSS). However, "decision makers choose to trust and use these forecast only if they believe these predictions are justifiable, relevant, and valuable if effectively managing the uncertainties about the future"[26]. In other word, a successful DSS depends not only on the accuracy of the results, but also on acceptance of these results by its users.

Teach and Shortliffe[36] show that "the users of a system that dogmatically offers advice are very likely to reject it, even if the system has impressive accuracy and the ability to provide reliable results". DSS should be able not only to provide accurate results but also to explain their own reasoning. Davis and Kotteman[9] carry on an experiment using a production planning DSS , and Lim and O'Connor[25] do the same thing using a forecasting DSS.

The results show that students trained continue to ignore the good advice given by the DSS even when they could see the DSS was outperforming their own efforts. It seems clear then that the key element in DSS design is to enhance the effective communication between the users and DSS. People want to use computer systems to solve semi structured or unstructured problems. Those problems are characterized by a complex decision-making process. There are no universal rules and models to solve them. Decision makers' subjective behaviors, such as knowledge, experience, intuition, common sense, insight, personal preferences and decision-making style, greatly affect the results. Semi-structured and non-structured problems cannot use a particular formula or fixed way to solve. The right way is to collect information relevant to the issue, and then classify, sort and establish associate relations between information. The aim is to arrange the information into a form of easy-to-view and understanding. Different types of information apply to the performance of different forms, such as text, images and sounds. Decision makers' subjective behavior could be affected by

the expression of information. Personal preferences and decision-making style makes them show the different interests for different forms of information. And decision makers have different sensitivity when they face the same information. So it is necessary to make information easier to accept when DSS organize them. According of information from different source, DSS can be divided into data-driven DSS, document-driven DSS, knowledge-driven DSS, and communication-driven DSS.

Power called such DSS use large amount of documents for decision as "document-driven decision support system"in his books [27]. Such DSS is characterized by "uses computer storage and processing technologies to provide document retrieval and analysis, a search engine is a primary decision aiding tool [28]". In order to get the better results, decision makers often need to collect all possible documents. Those documents contain the amount of information is often more than necessary. Attendant problem is that decision makers need to spend a lot of time to read these documents. So the main function of document-driven DSS is documents management, retrieval and tracking. The typical tools such as Google and Baidu search engine. These tools can locate and inquiry documents information, but they are lack of a means to analyze the association between the relevant documents, cannot explain the relationship in the information transmission. In the face of that "independent"information, decision maker's acceptance is relatively low. Led to this situation, there are two major factors:

(1)To decision maker's point of view, the advice generation process is a "black box". They cannot see the internal calculation, reasoning process, thus, decision maker find it is hard to believe that the advice is reliable.

(2)The forms of recommendations are diverse. There are graphics, text, sounds and images. Because of the existence of human cognitive bias, so that different people have different understanding for the same advice.

To resolve the problem of advice acceptability in DSS, the main proposal is to make the necessary "explanations"for those advices. "Explanation is the procedure of describing the operation of a system and designing computer programs so that they can be explained [34]". "Provision of an explanation may constitute one of the most effective methods for improving the acceptance of forecasts or other decision support advice [24]". This shows that the explanation is an important method to improve the acceptability of advices, thereby improving

decision-making effects.

The research of explanation is started from Expert System (ES), and the research methods are followed in DSS. L. Richard Ye and E. Johnson discussed the content of explanations needs to be studied in their paper [30], there are three main categories:

(1)Trace, this kind of explanation provides a perspective about the internal outcome procedure that is used by the DSS. Thus a trace explanation usually answers the question "how".

(2)Justification, these explanations often taps into the deep knowledge that is behind a generated outcome, and they provide the rational behind a decision. They generally answer the question "why".

(3)Strategy, these explanations provide an insight about the overall problem-solving strategy used by system.

They try to form a global perspective on the operation of the system by tapping into the planning.Explanations should be provided throughout the entire decision-making process. Different levels of decision making need different explanations. In the document-driven DSS, the inheritance and reference relations exist between documents. Just like blood system in biological systems, there are some lineage relationships in documents. These relationships play an important role to explain the document's trace and cluster related documents.

1.1 Contribution of this thesis

Our first discussed algorithm was k-mean algorithm. Regarding computational complexity, the k-means clustering problem for observations in d dimensions is: If k and d are fixed, the problem can be exactly solved in time $O(n^{dk+1}\log n)$, where n is the number of entities to be clustered. And the second discussed algorithm was *kNN* algorithm. The computational load of the *kNN* classification into k clusters is $O(knd)$, where d is the dimension of feature vectors and n is the number of training samples. We tried to find out a experimental result about which algorithm can classify all observations into user specified clusters. After implementing the algorithms and then performing accuracy test we observed that *kNN* algorithm has a better correction rate than k-mean algorithm. Then we developed a document linage pattern among the documents of k clusters. This patter helps to find an explanation of a

decision.

1.2 Organization of this thesis

The organization of the rest of the thesis is as follows. In Chapter 2.1, we present previous works on DSS. In Chapter 3, we give preliminary concepts. In Chapter 4, we focus on the performance of the algorithms. In Chapter 5, we describe our experimental result. Finally we conclude in Chapter 6.

CHAPTER 2
LITERATURE REVIEW

2.1 Previous Work

The concept of a Decision Support System (DSS) is extremely broad and its definitions vary depending upon the author's point of view [12]. A DSS can take many different forms and the term can be used in many different ways [2]. On the one hand, [13] and others defined it broadly as "a computer-based system that aids the process of decision making". In a more precise way, Turban (1995) defines it as "an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a nonstructured management problem for improved decision making. It utilizes data, provides an easy-to-use interface, and allows for the decision maker's own insights". On the other hand, Schoff [31] quoted Keen [21] ("there can be no definition of decision support systems, only of decision support") as claiming that it is impossible to give a precise definition including all the facets of the DSS. Nevertheless, according to Power (1997), the term decision support system remains a useful and inclusive term for many types of information systems that support decision making. He humorously adds that every time a computerized system is not an on-line transaction processing system (OLTP), someone will be tempted to call it a DSS.

In the absence of an all-inclusive definition, we focus on the brief history of DSS based on Power's point of view. Power [29] made his comments: According to Keen and Scott Morton, the concept of decision support has evolved from two main areas of research: the theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early 1960s, and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s. It is considered that the concept of DSS became an area of research of its own in the middle of the 1970s, before gaining in intensity during the 1980s. In the middle and late 1980s,

executive information systems (EIS), group decision support systems (GDSS), and organizational decision support systems (ODSS) evolved from the single user and model-oriented DSS. Beginning in about 1990, data warehousing and on-line analytical processing (OLAP) began broadening the realm of DSS.

As the turn of the millennium approached, new Web-based analytical applications were introduced. Chen Wenwei in China proposes an integrated DSS containing three main parts. The first part is the integration of model-base system and database system. It is the basic for decision support. It provides supported information for the quantitative analysis (model calculation) of decision-making. The second part is the data warehouse and OLAP, which extracts integrated data and information from the data warehouse. These data and information reflect the inherent nature of a large amount of data. The third part is the integration of expert system and data mining. Data mining explores data from the database and data warehouse and put them into the knowledge base of the expert system to make knowledge-based reasoning for quantitative analysis of decision aid. The decision aid system which uses the combination of the first and third parts (expert system and data mining) is an intelligent DSS. The decision aid system which uses the second part (data warehouse and OLAP) is the new DSS. In OLAP, related models in the model base can be used to improve the data analysis ability of OLAP. Combination of the three parts, namely combining the three parts with the component of "problem integration and interaction system", forms the integrated DSS. Many authors define the architecture of a DSS in terms of various components (Sprague 1980; Bonczek, Holsapple and Whinston, 1981). Turban(1995) mentions that the various components of a traditional DSS and/or its users do not have to be in one location. They can be dispersed both geographically and organizationally throughout an organization and its environment. As mentioned in the Introduction, E-business (E-government) was born in the environment with the trend of globally development of trade and economy, global share of information knowledge, and global distribution of organizations. This makes it possible for the components of DSS and/or its users to spread all over the world in e-business environment. Internet is undoubtedly the best tool for the realization of information globalization. As a powerful driving force, Web-based E-business makes DSS expand from LAN to Internet. With this background, this paper proposes the distributed framework of DSS in ebusiness environment, and makes it possible to be implemented on the J2EE platform by using Internet-based Wed technologies.

CHAPTER 3
PRELIMINARIES

Document categorization may be seen as the task of determining an assignment of a value from $0, 1$ to each entry of the decision matrix shown in figure 3.1.

	d_1	...	...	d_j	...	...	d_n
c_1	a_{11}	...	...	a_{1j}	...	...	a_{1n}
...	...	...	...	...	...	...	...
c_i	a_{i1}	...	...	a_{ij}	...	...	a_{in}
...	...	...	...	...	...	...	...
c_m	a_{m1}	...	...	a_{mj}	...	...	a_{mn}

Figure 3.1: Decision matrix

Where $C = c_1, ..., c_m$ is a set of pre-defined categories, and $D = \overline{d}_1, ..., \overline{d}_n$ is a set of documents to be categorized. A value of 1 for a_{ij} is interpreted as a decision to file djunder c_i, while a value of 0 is interpreted as a decision not to file d_j under c_i.

3.1 Definition

3.1.1 Feature Space

In pattern recognition a feature space is an abstract space where each pattern sample is represented as a point in n-dimensional space. Its dimension is determined by the number of features used to describe the patterns. Similar samples are grouped together, which allows the use of density estimation for finding patterns.

The concept is a most used one in classification techniques like k nearest neighbors or support vector machines.

Feature Space (Nearest Neighbor Analysis)

The feature space chart is an interactive graph of the feature space (or a subspace, if there are more than 3 features). Each axis represents a feature in the model, and the location of points in the chart show the values of these features for cases in the training and holdout partitions.

Keys: In addition to the feature values, points in the plot convey other information. Shape indicates the partition to which a point belongs, either Training or Holdout. The color/shading of a point indicates the value of the target for that case; with distinct color values equal to the categories of a categorical target, and shades indicating the range of values of a continuous target. The indicated value for the training partition is the observed value; for the holdout partition, it is the predicted value. If no target is specified, this key is not shown. Heavier outlines indicate a case is focal. Focal cases are shown linked to their k nearest neighbors.

Controls and Interactivity: A number of controls in the chart allow one to explore the Feature Space.

One can choose which subset of features to show in the chart and change which features are represented on the dimensions.

“Focal cases”are simply points selected in the Feature Space chart. If one specified a focal case variable, the points representing the focal cases will initially be selected. However, any point can temporarily become a focal case if one selects it. The “usual”controls for

point selection apply; clicking on a point selects that point and deselects all others; Control-clicking on a point adds it to the set of selected points. Linked views, such as the Peers Chart, will automatically update based upon the cases selected in the Feature Space. One can change the number of nearest neighbors (k) to display for focal cases.

Hovering over a point in the chart displays a tooltip with the value of the case label, or case number if case labels are not defined, and the observed and predicted target values.

A "Reset"button allows one to return the Feature Space to its original state.

3.1.2 Cross Validation Test

Cross validation divides the sample into a number of subsamples, or folds. Tree models are then generated, excluding the data from each subsample in turn. The first tree is based on all of the cases except those in the first sample fold; the second tree is based on all of the cases except those in the second sample fold, and so on. For each tree, misclassification risk is estimated by applying the tree to the subsample excluded in generating it. One can specify a maximum of 25 sample folds. The higher the value, the fewer the number of cases excluded for each tree model.

Cross validation produces a single, final tree model. The cross validated risk estimate for the final tree is calculated as the average of the risks for all of the trees.

Split-Sample Validation

With split-sample validation, the model is generated using a training sample and tested on a hold-out sample.

One can specify a training sample size, expressed as a percentage of the total sample size, or a variable that splits the sample into training and testing samples.

If one uses a variable to define training and testing samples, cases with a value of 1 for the variable are assigned to the training sample, and all other cases are assigned to the testing sample. The variable cannot be the dependent variable, weight variable, influence variable, or a forced independent variable. One can display results for both the training and testing samples or just the testing sample. Split-sample validation should be used with caution on

small data files (data files with a small number of cases). Small training sample sizes may yield poor models, since there may not be enough cases in some categories to adequately grow the tree.

Cross Validation Subcommand (*kNN* command)

The CROSSVALIDATION subcommand specifies settings for performing V-fold cross-validation to determine the "best"number of neighbors.

V-fold cross validation divides the data into V folds. Then, for a fixed k, it applies nearest neighbor analysis to make predictions on the v-th fold (using the other $V-1$ folds as the training sample) and evaluates the error. This process is successively applied to all possible choices of v. At the end of V folds, the computed errors are averaged. The above steps are repeated for various values of k. The value achieving the lowest average error is selected as the optimal value for k.

If multiple values of k are tied on the lowest average error, then the smallest k among those that are tied is selected.

Cross-validation is not used when /MODEL NEIGHBORS=FIXED or when /MODEL FEATURES=AUTO. It is invalid to specify both the FOLDS and VARIABLE keywords on the CROSSVALIDATION subcommand.

3.1.3 Decision Boundary

In a statistical-classification problem with two classes, a decision boundary or decision surface is a hyper-surface that partitions the underlying vector space into two sets, one for each class. The classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class. If the decision surface is a hyper-plane, then the classification problem is linear, and the classes are linearly separable.

Decision boundaries are not always clear cut. That is, the transition from one class in the feature space to another is not discontinuous, but gradual. This effect is common in fuzzy logic based classification algorithms, where membership in one class or another is ambiguous.

In general, a pattern classifier carves up (or tesselates or partitions) the feature space into volumes called decision regions. All feature vectors in a decision region are assigned to the same category. The decision regions are often simply connected, but they can be multiply connected as well, consisting of two or more non-touching regions shown in figure 3.2.

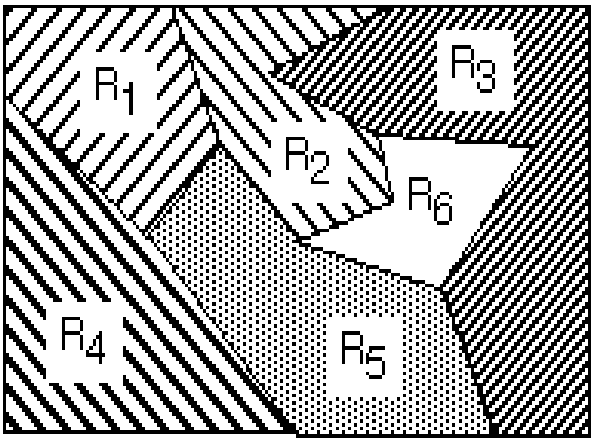

Figure 3.2: Decision Region

The decision regions are separated by surfaces called the decision boundaries. These separating surfaces represent points where there are ties between two or more categories.

For a minimum-distance classifier, the decision boundaries are the points that are equally distant from two or more of the templates. With a Euclidean metric, the decision boundary between Regions i and Region j is on the line or plane that is the perpendicular bisector of the line from m_i to m_j. Analytically, these linear boundaries are a consequence of the fact that the discriminant functions are linear. (With the Mahalanobis metric, the decision boundaries are quadratic surfaces, such as ellipsoids, paraboloids or hyperboloids.)

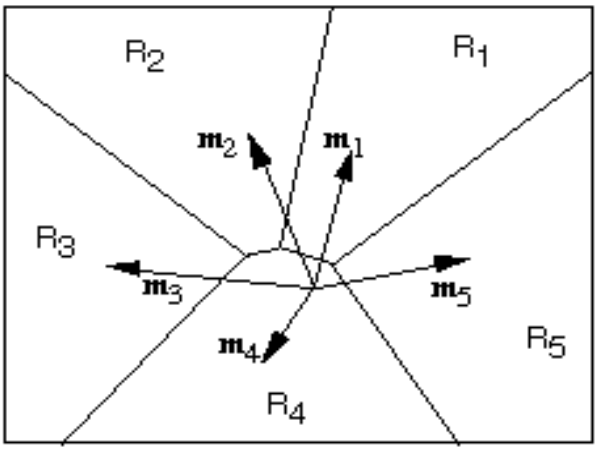

Figure 3.3: Nearest-template decision boundaries

How well the classifier works depends upon how closely the input patterns to be classified resemble the templates. In the example sketched below, the correspondence is very close, and one can anticipate excellent performance. However, things are not always this good in practice, and one should understand the limitations of simple classifiers.

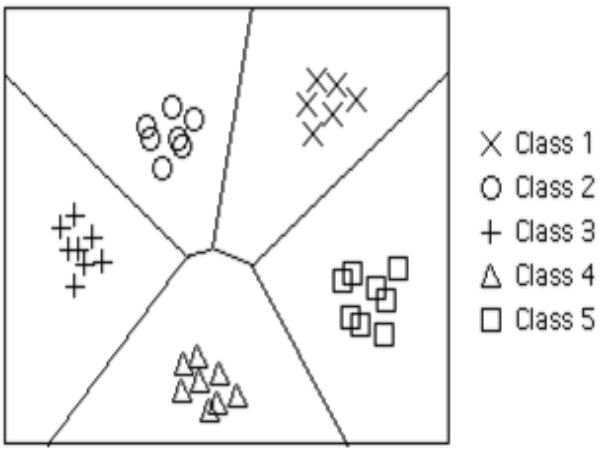

Figure 3.4: Result of Classification

3.1.4 Lazy Learning

In artificial intelligence, lazy learning is a learning method in which generalization beyond the training data is delayed until a query is made to the system, as opposed to in eager learning, where the system tries to generalize the training data before receiving queries. The main advantage gained in employing a lazy learning method, such as Case based reasoning, is that the target function will be approximated locally, such as in the *k*-nearest neighbor algorithm. Because the target function is approximated locally for each query to the system, lazy learning systems can simultaneously solve multiple problems and deal successfully with changes in the problem domain.

The disadvantages with lazy learning include the large space requirement to store the entire training dataset. Particularly noisy training data increases the case base unnecessarily, because no abstraction is made during the training phase. Another disadvantage is that lazy learning methods are usually slower to evaluate, though this is coupled with a faster training phase. Lazy classifiers are most useful for large datasets with few attributes.

3.2 Problem Description

Document categorization relies on the existence of a an initial corpus $C_o = d_1.....d_s$ of documents previously categorized under the same set of categories $C = c_1, ..., c_m$ with which the categorizer must operate. This means that the corpus comes with a correct decision matrix in figure 3.5.

	Training set				Test set			
	d_1	...	...	d_g	d_{g+1}	...	...	d_s
c_1	ca_{11}	...	...	ca_{1g}	$ca_{1(g+1)}$	...	...	ca_{1s}
...	...	...	...	...	...	...	...	...
c_i	ca_{i1}	...	...	ca_{ig}	$ca_{i(g+1)}$	...	...	ca_{is}
...	...	...	...	...	...	...	...	...
c_m	ca_{m1}	...	...	ca_{mg}	$ca_{m(g+1)}$	...	...	ca_{ms}

Figure 3.5: Correct decision matrix

A value of 1 for ca_{ij} is interpreted as an indication from the expert to file d_j under c_i, while a value of 0 for is interpreted as an indication from the expert not to file d_j under c_i. A document d_j is often referred to as a positive example of c_i if $ca_{ij} = 1$, a negative example of c_i if $ca_{ij} = 0$. For evaluation purposes, in the first stage of classifier construction the initial corpus is typically divided into two sets, not necessarily of equal size:

* A *training set* $T_r = d_1, ..., d_g$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced.

* A *test set* $T_e = d_{g+1}, ..., d_s$. This set will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in T_e will be fed to the classifiers, and the classifier decisions compared with the expert decisions; a measure of classification effectiveness will be based on how often the values for the a_{ij}'s obtained by the classifiers match the values for the ca_{ij}'s provided by the experts.

Note that in order to give a scientific character to the experiment the documents in T_e cannot participate in any way in the inductive construction of the classifiers; if this condition were not to be satisfied, the experimental results obtained would typically be unrealistically good.

However, it is often the case that in order to optimize a classifier, its internal parameters should be tuned by testing which value of the parameter yields the best effectiveness. In order for this to happen and, at the same time, safeguard scientific standards, the set $d_1, ..., d_g$ may be further split into a "true"training set $T_r = d_1, ..., d_f$, from which the classifier is induced, and a *validation testset* $V_a = d_{f+1}, ..., d_g$, on which the repeated tests of the induced classifier aimed at parameter optimization are performed.

Finally, one may define the generality $g_{Co}(c_i)$ of a category c_i relative to a corpus C_o as the percentage of documents that belong to c_i, i.e.:

$$g_{Co}(c_i) = |\{d_j \in C_o | ca_{ij} = 1\}| / |\{d_j \in C_o\}| \ldots\ldots\ldots (1)$$

The training set generality $g_{Tr}(c_i)$, validation set generality $g_{va}(c_i)$, and test set generality $g_{Te}(c_i)$ of a category c_i may be defined in the obvious way by substituting T_r, V_a, or T_e, respectively, to C_0 in Equation 1. Then we developed tracing of document-driven DSS to provide an explanation to improve the acceptance of decision makers.

CHAPTER 4
PROCEDURE

4.1 The k-means algorithm

The K-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, k. This algorithm has been discovered by several researchers across different disciplines, most notably Lloyd (1957, 1982) [32], Forgey (1965), Friedman and Rubin (1967), and McQueen (1967). Detailed histories of k-means along with descriptions of several variations are given in [20]. Gray and Neuhoff [17] provide a nice historical background for k-means placed in the larger context of hill-climbing algorithms.

4.1.1 Description of the algorithm

The algorithm operates on a set of d-dimensional vectors, $D = x_i | i = 1...N$, where $x_i \in R_d$ denotes the ith data point. The algorithm is initialized by picking k points in R^d a as the initial k cluster representatives or "centroids". Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data k times. Then the algorithm iterates between two steps till convergence:

Step 1: Data Assignment

Each data point is assigned to its closest centroid, with ties broken arbitrarily. This results in a partitioning of the data.

Step 2: Relocation of "means"

Each cluster representative is relocated to the center (mean) of all data points assigned to it. If the data points come with a probability measure (weights), then the relocation is to the expectations (weighted mean) of the data partitions. The algorithm converges when the

assignments (and hence the C_j values) no longer change. The algorithm execution is visually depicted from figure. Note that each iteration needs $N \times k$ comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on N, but as a first cut, this algorithm can be considered linear in the dataset size. One issue to resolve is how to quantify "closest"in the assignment step. The default measure of closeness is the Euclidean distance, in which case one can readily show that the non-negative cost function,

$$\sum_{i=1}^{n} \left(argmin ||x_i - c_i||^2 \right)$$

will decrease whenever there is a change in the assignment or the relocation steps, and hence convergence is guaranteed in a finite number of iterations. The greedy-descent nature of k-means on a non-convex cost also implies that the convergence is only to a local optimum, 1 and indeed the algorithm is typically quite sensitive to the initial "centroid"locations. The local minima problem can be countered to some extent by running the algorithm multiple times with different initial centroids, or by doing limited local search about the converged solution.

Regarding computational complexity, the k-means clustering problem for observations in d dimensions is: If k and d are fixed, the problem can be exactly solved in time $O(n^{dk+1} \log n)$, where n is the number of entities to be clustered.

4.1.2 k-mean Algorithm

In this section we have described k-mean algorithm in Algorithm 1.

4.1.3 Limitations

In addition to being sensitive to initialization, the k-means algorithm suffers from several other problems. First, observe that k-means is a limiting case of fitting data by a mixture of k Gaussians with identical, isotropic covariance matrices ($\sum = \sigma^2 I$), when the soft assignments of data points to mixture components are hardened to allocate each data point solely to the most likely component. So, it will falter whenever the data is not well described by reasonably separated spherical balls, for example, if there are non-covex shaped clusters in

Algorithm 1 k-mean Algorithm

Input: $E = e_1, e_2, ..., e_n$ (set of entities to be clustered)
k(number of clusters)
$MaxIters$ (limit of iterations)

output: $C = c_1, c_2, ..., c_k$ (set of cluster centroids)
$L = l(e)|e = 1, 2, ..., n$ (set of cluster levels of E)

for each $c_i \in C$ do
$c_i \leftarrow e_j \in E$ (random selection)
end

for each $e_i \in E$ do
$l(e_i) \leftarrow argminDistance(e_i, , c_j)\ j \in 1, 2, ..., k$
end

$changed \leftarrow false$
$iter \leftarrow 0$

repeat
for each $c_i \in C$ do
$UpdateCluster(c_i)$
end
for each $e_i \in E$ do
$minDist \leftarrow argminDistance(e_i, , c_j)\ j \in 1, 2, ..., k$
if $minDist \neq l(e_i)$ then
$l(e_i) \leftarrow minDist$
$changed \leftarrow true$
end
end
$iter++$
$untilchanged = trueanditer \leq MaxIter$

the data. This problem may be alleviated by rescaling the data to "whiten"it before clustering, or by using a different distance measure that is more appropriate for the dataset. For example, information-theoretic clustering uses the KL-divergence to measure the distance between two data points representing two discrete probability distributions. It has been recently shown that if one measures distance by selecting any member of a very large class of divergences called Bregman divergences during the assignment step and makes no other changes, the essential properties of k-means, including guaranteed convergence, linear separation boundaries and scalability, are retained [3]. This result makes k-means effective for a much larger class of datasets so long as an appropriate divergence is used.

K-means can be paired with another algorithm to describe non-convex clusters. One first clusters the data into a large number of groups using k-means. These groups are then agglomerated into larger clusters using single link hierarchical clustering, which can detect complex shapes. This approach also makes the solution less sensitive to initialization, and since the hierarchical method provides results at multiple resolutions, one does not need to pre-specify k either.

The cost of the optimal solution decreases with increasing k till it hits zero when the number of clusters equals the number of distinct data-points. This makes it more difficult to (a) directly compare solutions with different numbers of clusters and (b) to find the optimum value of k. If the desired k is not known in advance, one will typically run k-means with different values of k, and then use a suitable criterion to select one of the results. For example, SAS uses the cube-clustering-criterion, while X-means adds a complexity term (which increases with k) to the original cost function and then identifies the k which minimizes this adjusted cost. Alternatively, one can progressively increase the number of clusters, in conjunction with a suitable stopping criterion. Bisecting k-means [33] achieves this by first putting all the data into a single cluster, and then recursively splitting the least compact cluster into two using 2-means. The celebrated LBG algorithm [17] used for vector quantization doubles the number of clusters till a suitable code-book size is obtained. Both these approaches thus alleviate the need to know k beforehand.

The algorithm is also sensitive to the presence of outliers, since "mean"is not a robust statistic. A preprocessing step to remove outliers can be helpful. Post-processing the results, for example to eliminate small clusters, or to merge close clusters into a large cluster, is also

desirable. Ball and Hall's ISODATA algorithm from 1967 effectively used both pre- and post-processing on k-means.

4.1.4 Generalizations and connections

As mentioned earlier, *k*-means is closely related to fitting a mixture of *k* isotropic Gaussians to the data. Moreover, the generalization of the distance measure to all Bregman divergences is related to fitting the data with a mixture of *k* components from the exponential family of distributions. Another broad generalization is to view the "means"as probabilistic models instead of points in Rd. Here, in the assignment step, each data point is assigned to the most likely model to have generated it. In the "relocation"step, the model parameters are updated to best fit the assigned datasets. Such model-based *k*-means allow one to cater to more complex data, e.g. sequences described by Hidden Markov models. One can also "kernelize"*k*-means [11]. Though boundaries between clusters are still linear in the implicit high-dimensional space, they can become non-linear when projected back to the original space, thus allowing kernel k-means to deal with more complex clusters. Dhillon et al. [11] have shown a close connection between kernel *k*-means and spectral clustering. The K-medoid algorithm is similar to k-means except that the "centroids"have to belong to the data set being clustered. Fuzzy *c*-means is also similar, except that it computes fuzzy membership functions for each clusters rather than a hard one. Despite its drawbacks, *k*-means remains the most widely used partitioned clustering algorithm in practice. The algorithm is simple, easily understandable and reasonably scalable, and can be easily modified to deal with streaming data. To deal with very large datasets, substantial effort has also gone into further speeding up *k*-means, most notably by using kd-trees or exploiting the triangular inequality to avoid comparing each data point with all the centroids during the assignment step. Continual improvements and generalizations of the basic algorithm have ensured its continued relevance and gradually increased its effectiveness as well.

4.2 k-nearest neighbor classification

One of the simplest and rather trivial classifiers is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of the test object match

one of the training examples exactly. An obvious drawback of this approach is that many test records will not be classified because they do not exactly match any of the training records. A more sophisticated approach, k-nearest neighbor (kNN) classification [14, 35], finds a group of k objects in the training set that are closest to the test object, and bases the assignment of a label on the predominance of a particular class in this neighborhood. There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance or similarity metric to compute distance between objects, and the value of k, the number of nearest neighbors. To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class labels of these nearest neighbors are then used to determine the class label of the object.

4.2.1 Description of the algorithm

Given a training set D and a test object $x = (\overline{x}, y)$, the algorithm computes the distance (or similarity) between z and all the training objects $(x, y) \in D$ to determine its nearest-neighbor list, D_z. (x is the data of a training object, while y is its class. Likewise, $\overline{x}$ is the data of the test object and $\overline{y}$ is its class.) Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

Majority Voting:

$$\overline{y} = \underset{v}{argmax} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

Where v is a class label, y_i is the class label for the ith nearest neighbors, and I (.) is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

4.2.2 kNN Algorithm

Algorithm 2 kNN Algorithm

Input: D the set of training objects and test object $z = (\bar{x}, \bar{y})$

Process:

Compute $d(\bar{x}, x)$, the distance between and every object, $(x, y) \in D$
Select, $D_z \subseteq D$ the set of closest training objects to z.

output:

$$\bar{y} = \underset{v}{argmax} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

4.2.3 Analysis of the Algorithm

There are several key issues that affect the performance of *k*NN. One is the choice of k. If k is too small, then the result can be sensitive to noise points. On the other hand, if k is too large, then the neighborhood may include too many points from other classes.

Another issue is the approach to combining the class labels. The simplest method is to take a majority vote, but this can be a problem if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the class of the object. A more sophisticated approach, which is usually much less sensitive to the choice of k, weights each object's vote by its distance, where the weight factor is often taken to be the reciprocal of the squared distance: $w_i = 1/d(\bar{x}, x_i)^2$.This amounts to replacing the last step of the k-NN algorithm with the following:
Distance-Weighted Voting: $\bar{y} = \underset{v}{argmax} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$.

The choice of the distance measure is another important consideration. Although various measures can be used to compute the distance between two points, the most desirable distance measure is one for which a smaller distance between two objects implies a greater likelihood of having the same class. Thus, for example, if *k*NN is being applied to classify documents, then it may be better to use the cosine measure rather than Euclidean distance.

Some distance measures can also be affected by the high dimensionality of the data. In particular, it is well known that the Euclidean distance measure become less discriminating as the number of attributes increases. Also, attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.

For example, consider a data set where the height of a person varies from 1.5 to 1.8m, the weight of a person varies from 90 to 300 lb, and the income of a person varies from $10,000 to $1,000,000. If a distance measure is used without scaling, the income attribute will dominate the computation of distance and thus, the assignment of class labels. A number of schemes have been developed that try to compute the weights of each individual attribute based upon training set [18].

In addition, weights can be assigned to the training objects themselves. This can give more weight to highly reliable training objects, while reducing the impact of unreliable objects. The PEBLS system by Cost and Salzberg [6] is a well-known example of such an approach.

Classifying unknown objects is relatively expensive since it requires the computation of the *k*-nearest neighbors of the object to be labeled. This, in general, requires computing the distance of the unlabeled object to all the objects in the labeled set, which can be expensive particularly for large training sets. A number of techniques have been developed for efficient computation of *k*-nearest neighbor distance that make use of the structure in the data to avoid having to compute distance to all objects in the training set. These techniques, which are particularly applicable for low dimensional data, can help reduce the computational cost without affecting classification accuracy.

The computational load of the *k*NN classification of a single test point is O (knd), where d is the dimension of feature vectors and n is the number of training samples. O (knd) is a lot of time, particularly if n is large. The difference to most other techniques for classification is that with *k*NN training points are needed during the classification when usually they are needed only during the training. Since training is performed only once and the classification many times, we are more concerned about the time consumption of classification.

4.2.4 Impact

kNN classification is an easy to understand and easy to implement classification technique. Despite its simplicity, it can perform well in many situations. In particular, a well known result by Cover and Hart [7] shows that the error of the nearest neighbor rule is bounded above by twice the Bayes error under certain reasonable assumptions. Also, the error of the general *kNN* method asymptotically approaches that of the Bayes error and can be used to approximate it. kNN is particularly well suited for multi-modal classes as well as applications in which an object can have many class labels. For example, for the assignment of functions to genes based on expression profiles, some researchers found that *kNN* outperformed SVM, which is a much more sophisticated classification scheme [23].

4.2.5 Current and future research

Although the basic *kNN* algorithm and some of its variations, such as weighted kNN and assigning weights to objects, are relatively well known, some of the more advanced techniques for *kNN* are much less known. For example, it is typically possible to eliminate many of the stored data objects, but still retain the classification accuracy of the kNN classifier. This is known as 'condensing'and can greatly speed up the classification of new objects [19]. In addition, data objects can be removed to improve classification accuracy, a process known as "editing" [39]. There has also been a considerable amount of work on the application of proximity graphs (nearest neighbor graphs, minimum spanning trees, relative neighborhood graphs, Delaunay triangulations, and Gabriel graphs) to the *kNN* problem. Recent papers by Toussaint [38, 37], which emphasize a proximity graph viewpoint, provide an overview of work addressing these three areas and indicate some remaining open problems. Other important resources include the collection of papers by Dasarathy [8] and the book by Devroyeet al. [10]. Finally, a fuzzy approach to kNN can be found in the work of Bezdek [4].

4.3 The concept of document lineage

Document is carrier of information. It inherits from the original documents, and innovates the current information. This process allows information to pass along. Document lineage is generated in this process. Document lineage is a relationship among documents. This relationship is congenital and unalterable.

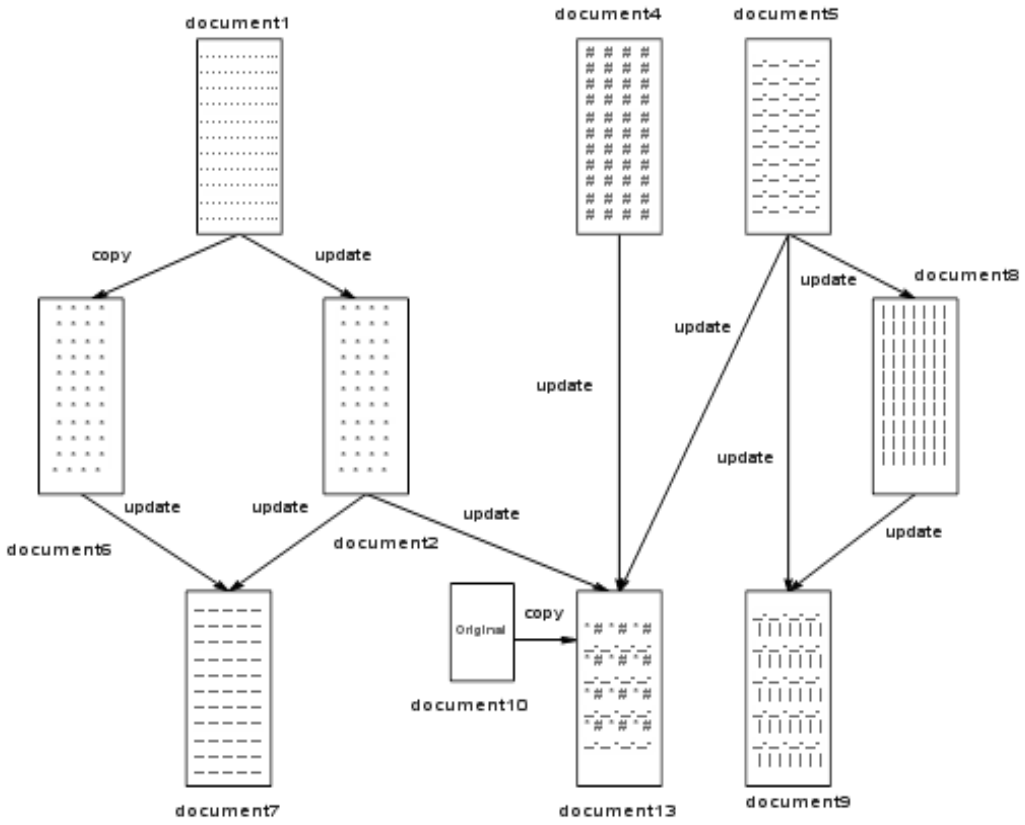

Figure 4.1: Document lineage diagram

In the documents transfer process, the new document is produced by inheritance and innovation. As shown in Figure 4.1: *document1*, *document4* and *document5* are the original documents, *document2* is the result of updates to *document1*, *document6* is a copy of *document1*, so *document2* and *document6* are descendants of document1. *Document7* contains the contents of *document2* and *document6*, *document3* is a combination of four documents, its contents are from *document2*, *document4*, *document5* and *document10*. There is also original part. *Document9* is a combination of *document5* and *document8*.

4.3.1 “Parent-child”Concept:

“Parents-child”relationship between documents, the direct inheritance relationship between thedocuments. In this relationship, the document has beeninherited is called the parent document, the document inherits the contents of the parent document is called child document [5]. For example in Figure 4.1, there are such relationships between *document1* and *document2*, between *document2* and *document3*, between *document5* and *documnet7*.

A document can be both “parent document”and “child document”. It depends on them to assume the role of lineage relationship, as in Figure, *document2* is *document1*’s childdocument, and *document2* is the parent document of *document3*. In the documents lineage system, all documents relationships can be described by these three relationships. So, we can form the “lineage pattern”which describes the relationship of relevant documents set. The significance of this map is to establish a link between separate documents. Such a link can express the source of document, and it is a interpretability for the “trace”in explanation.

4.3.2 Route Generation

In order to explain decision advice, we need to do some work at the three levels: Trace, Justification and Strategy. The “Trace”can help decision makers to know the source of advice; the “Justification”can prove the advice; and the “Strategy”is a strategic perspective to describe the effect of the advice. M. SinanGnl has such a view: “the explanations (especially the traces) can increase in visibility facilitates the participants’ acceptance of the system as being logical, and thus, acceptance of its advice as being justified and useful [15]”.

In document-driven DSS, decision makers obtain information by reading documents, and the references between documents constitute the routes of document lineage transmission. When the advice’s reliability needs to be audited, we can trace the origin of it and get the routes.

CHAPTER 5
EXPERIMENTAL RESULTS

We run the two algorithms for same data set and same number of observation.

5.1 Environment

All experimental test cases were run in a notebook with following specification:

5.1.1 Experimental setup

The experiments were performed using Intel CORE*i*3 CPU at 2.13 GHz processor having 4*GB* RAM running Windows 7 operating system.

5.1.2 IDE

We used **MATLAB R2011a. MATLAB** (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

5.1.3 Code Specification

The correctness of our approach is fully dependent on the code. We tried our best to implement both algorithms very carefully. We tried to make the code a compact one for having

the exact result. Then we performed cross-validation test to measure the accuracy both of the classifier.

5.2 Result of Classification

For each algorithm we used 150 observations ($n = 150$) and classified them in 9 categories ($k = 9$).Then we measured the performance of each classifier in best, average and worst case.

5.2.1 Performance of k-mean Classifier

Worst Case

Worst case for k-mean classifier (k=9) is shown in figure 5.1 and its performance is described in Table 5.1.

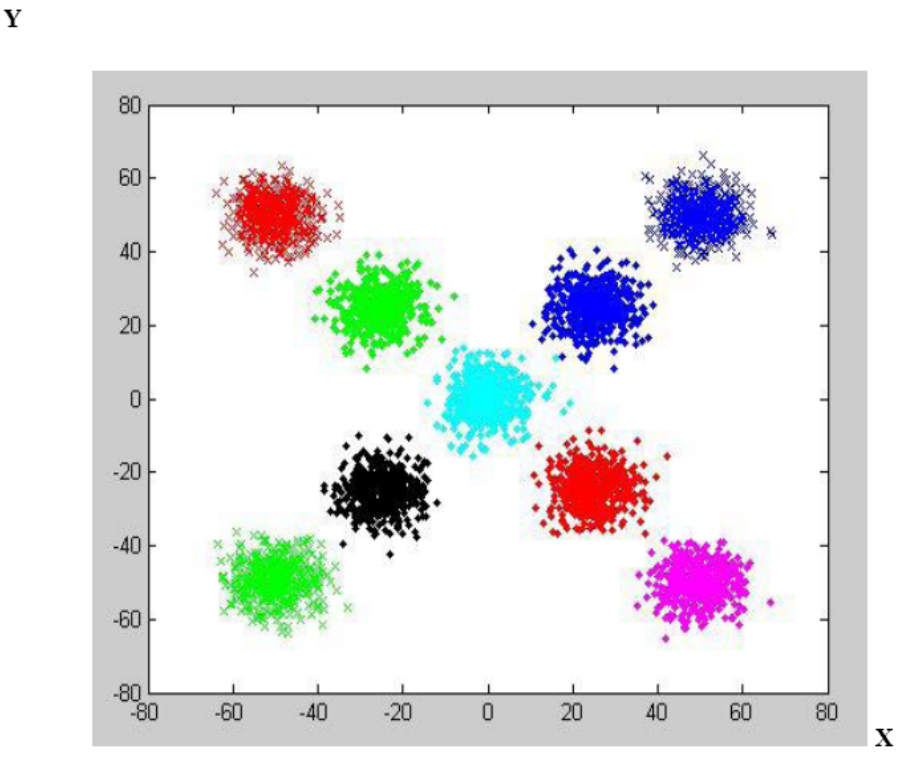

Figure 5.1: worst case for k-mean classifier (k=9)

Table 5.1: Worst case performance of k-mean algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	1
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.8667
ErrorRate	0.1333
LastCorrectRate	0.8667
LastErrorRate	0.1333
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

Average Case

Average case for *k*-mean classifier (k=9) is shown in figure 5.2 and its performance is described in Table 5.2.

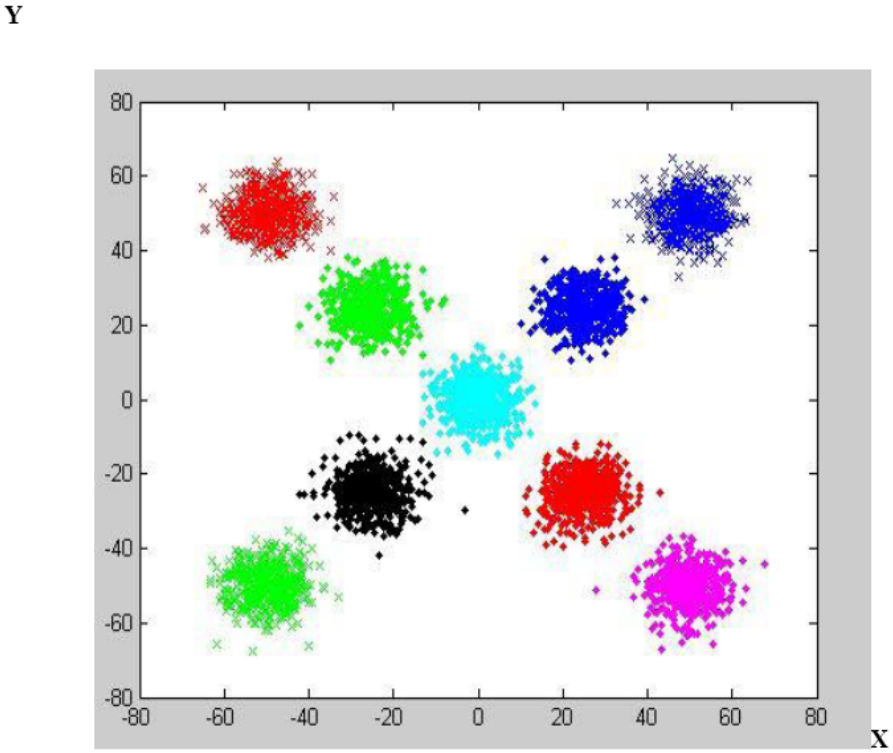

Figure 5.2: Average case for k-mean classifier (k=9)

Best Case

Best case for *k*-mean classifier (k=9) is shown in figure 5.3 and its performance is described in Table 5.3.

5.2.2 Performance of KNN Classifier

Worst Case

Worst case for *KNN* classifier (k=9) is shown in figure 5.4 and its performance is described in Table 5.4.

Table 5.2: Average case performance of k-mean algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	3
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.9111
ErrorRate	0.0889
LastCorrectRate	0.8667
LastErrorRate	0.1333
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

Table 5.3: Best case performance of k-mean algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	1
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.9333
ErrorRate	0.0667
LastCorrectRate	0.9333
LastErrorRate	0.0667
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

Table 5.4: Worst case performance of *kNN* algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	1
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.9333
ErrorRate	0.0667
LastCorrectRate	0.9333
LastErrorRate	0.0667
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

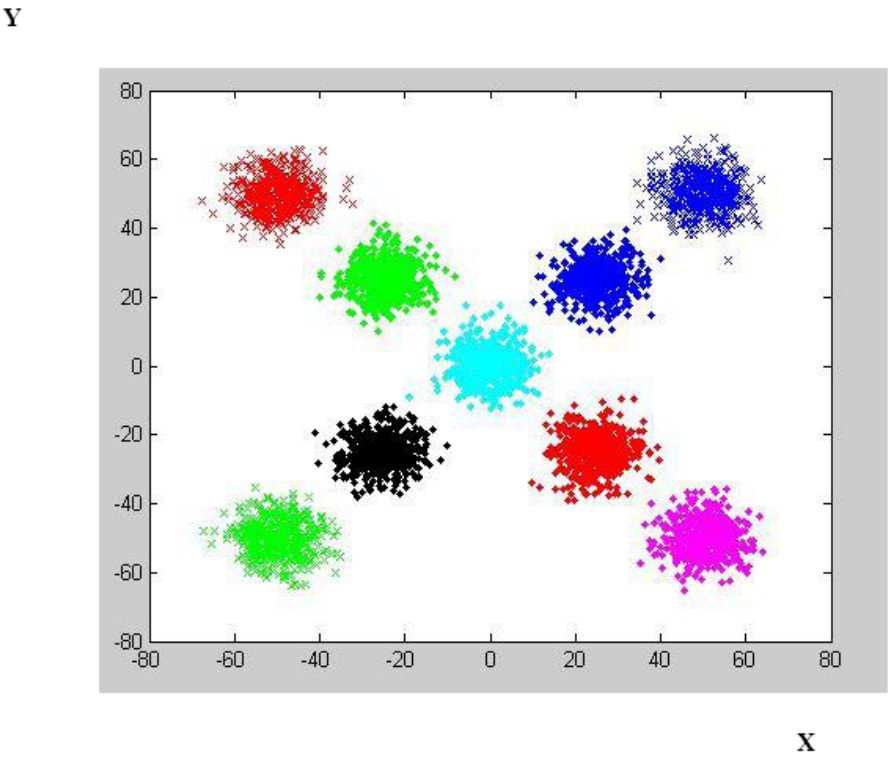

Figure 5.3: Best case for k-mean classifier (k=9)

Average Case

Average case for *KNN* classifier (k=9) is shown in figure 5.5 and its performance is described in Table 5.5.

Best Case

Best case for *KNN* classifier (k=9) is shown in figure 5.6 and its performance is described in Table 5.6.

5.3 Developing a Path among Classified Documents

Formal expression of paths of document lineage can be describedas follows:

Path $<$ Terminal Point, StartingPoint $\geq$ (catID,DocID,Action,ParentCatID,ParentDocId)

Thus we finally developed a linage patters among the classified documents which is shown in figure 5.12

Table 5.5: Average case performance of *kNN* algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	10
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.9400
ErrorRate	0.0600
LastCorrectRate	0.8667
LastErrorRate	0.1333
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

Table 5.6: Best case performance of *kNN* algorithm

ClassLabels	{3x1 cell}
GroundTruth	[150x1 double]
NumberOfObservations	150
ControlClasses	[2x1 double]
TargetClasses	1
ValidationCounter	1
SampleDistribution	[150x1 double]
ErrorDistribution	[150x1 double]
SampleDistributionByClass	[3x1 double]
ErrorDistributionByClass	[3x1 double]
CountingMatrix	[4x3 double]
CorrectRate	0.9800
ErrorRate	0.0200
LastCorrectRate	0.9800
LastErrorRate	0.0200
InconclusiveRate	0
ClassifiedRate	1
Sensitivity	1
Specificity	1
PositivePredictiveValue	1
NegativePredictiveValue	1
PositiveLikelihood	NaN
NegativeLikelihood	0
Prevalence	0.3333
DiagnosticTable	[2x2 double]

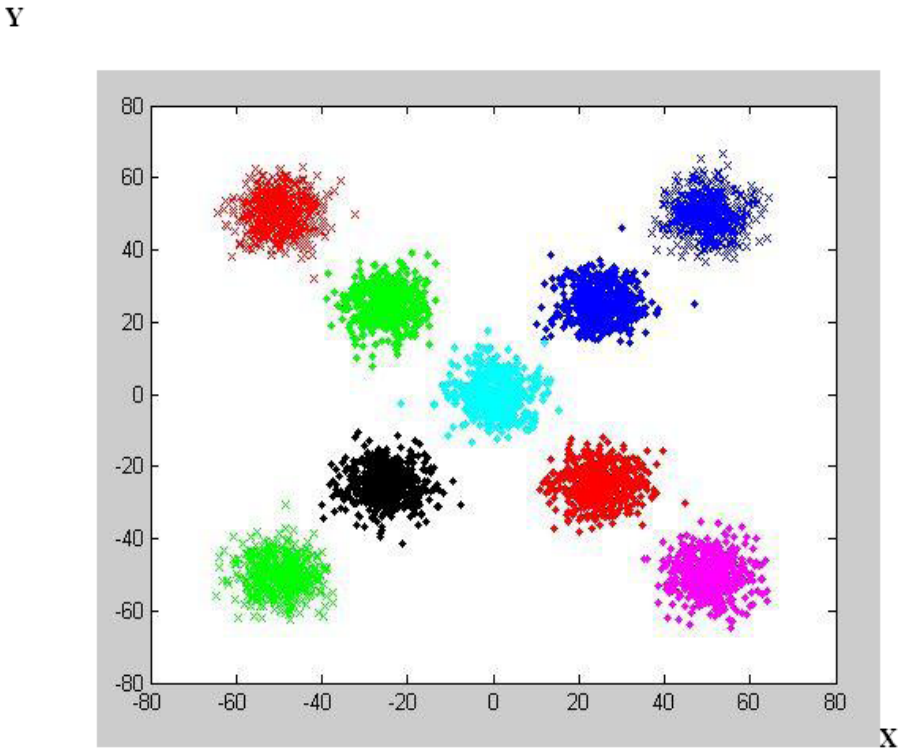

Figure 5.4: Worst case for *KNN* classifier (k=9)

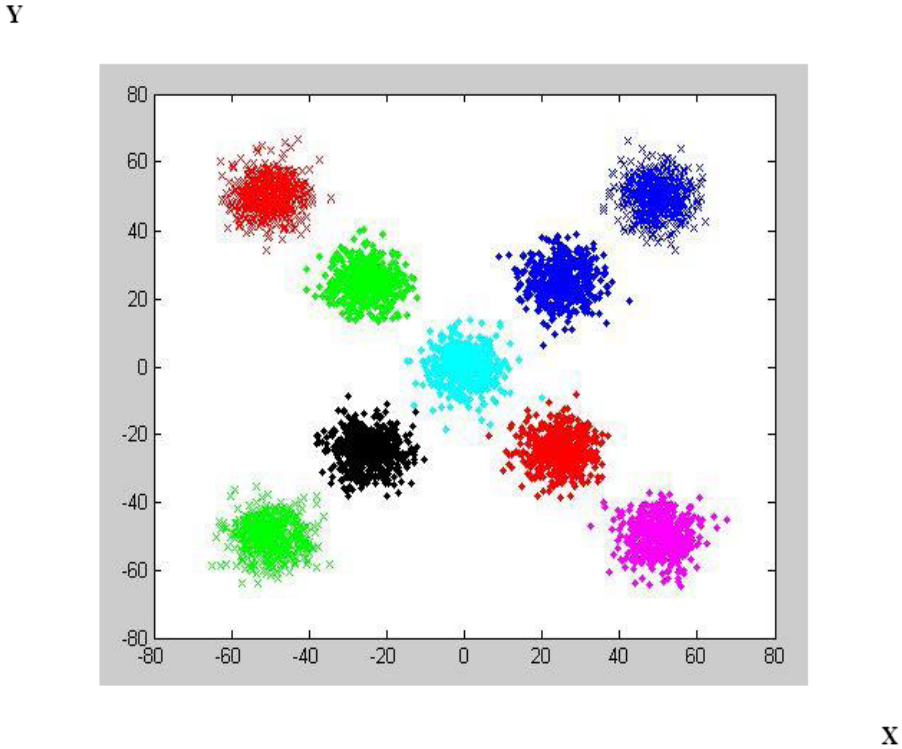

Figure 5.5: Average case for *kNN* classifier (k=9)

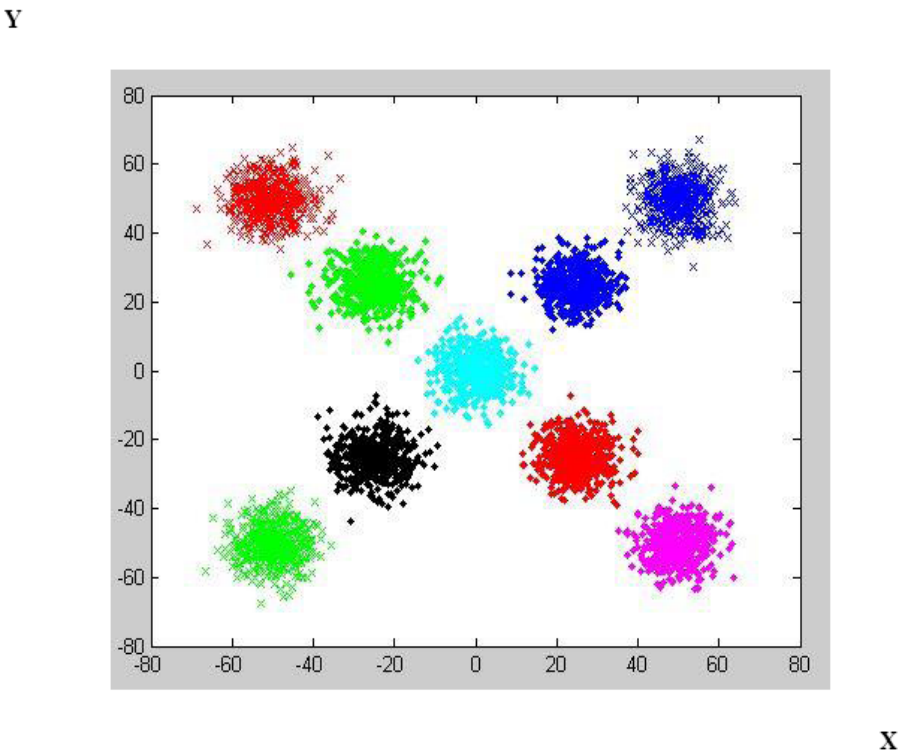

Figure 5.6: Best case for *kNN* classifier (k=9)

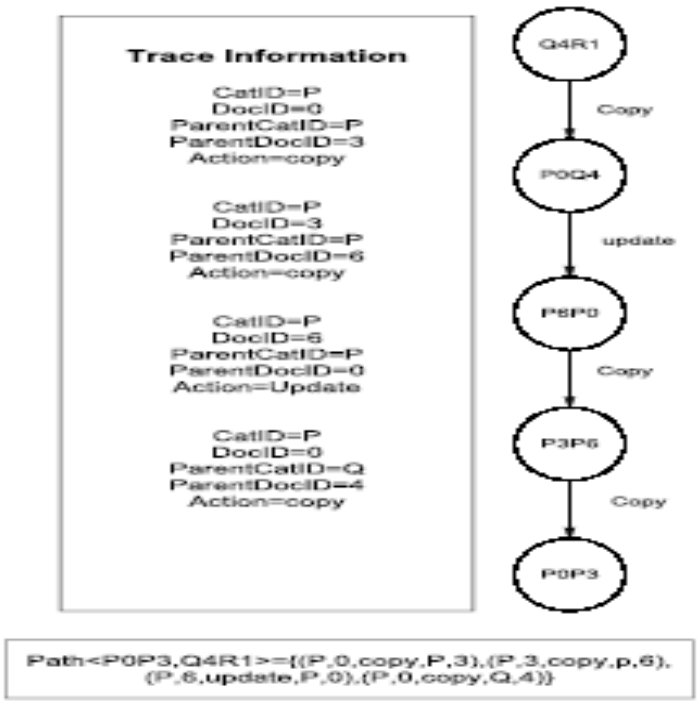

Figure 5.7: Path from *document* $Q4R1$ to *document* $P0P3$

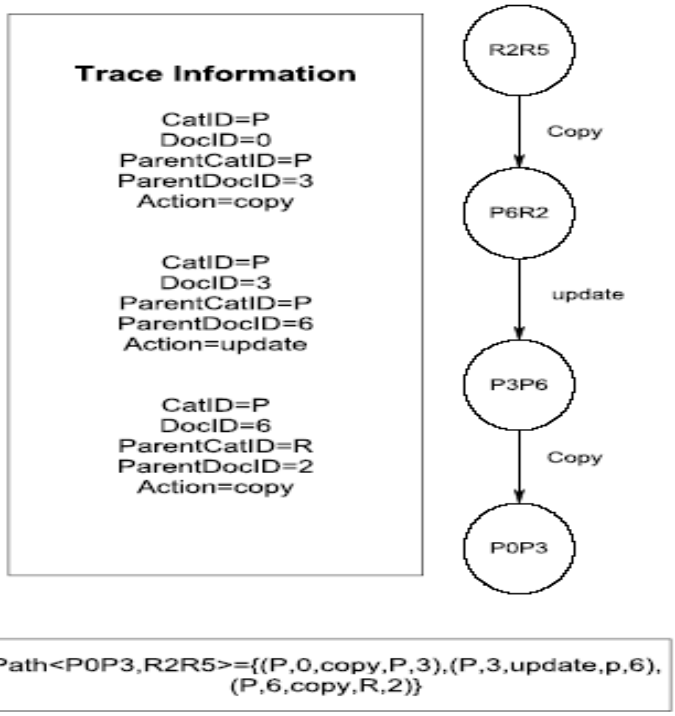

Figure 5.8: Path from *document R*2*R*5 to *document P*0*P*3

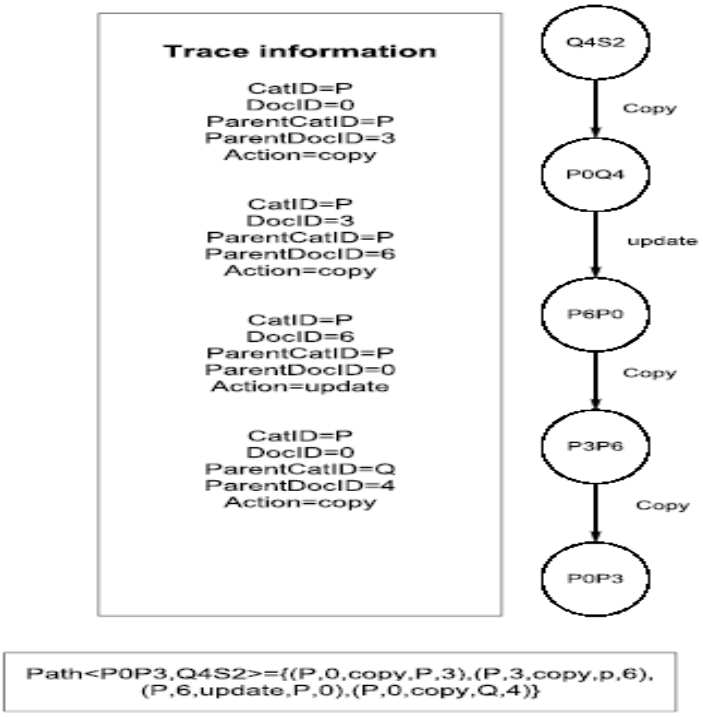

Figure 5.9: Path from *document Q*4*S*2 to *document P*0*P*3

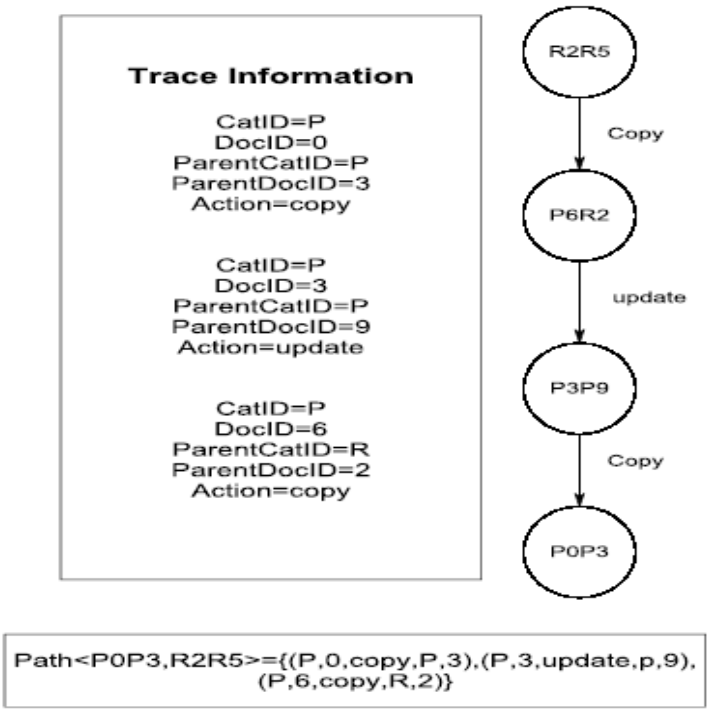

Figure 5.10: Path from *document R*2*R*5 to *document P*0*P*3

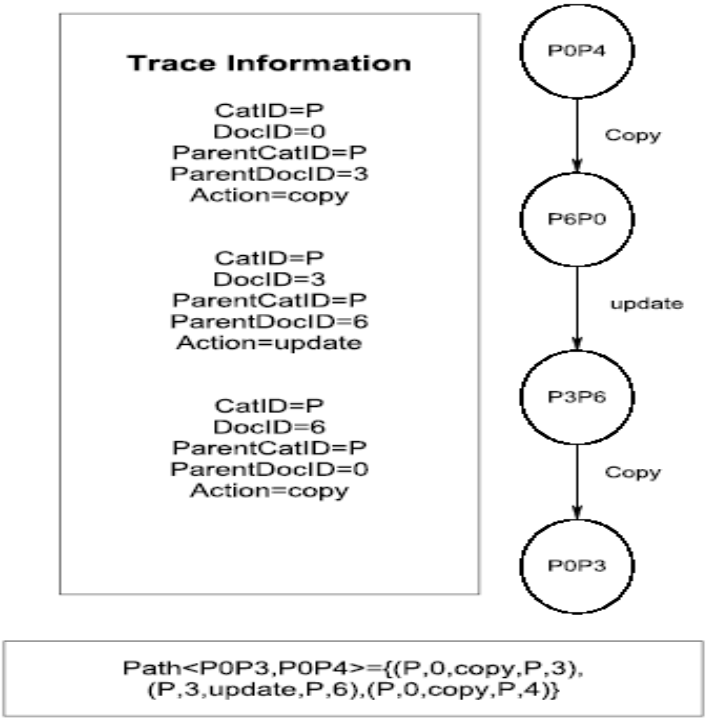

Figure 5.11: Path from *document P*0*P*4 to *document P*0*P*3

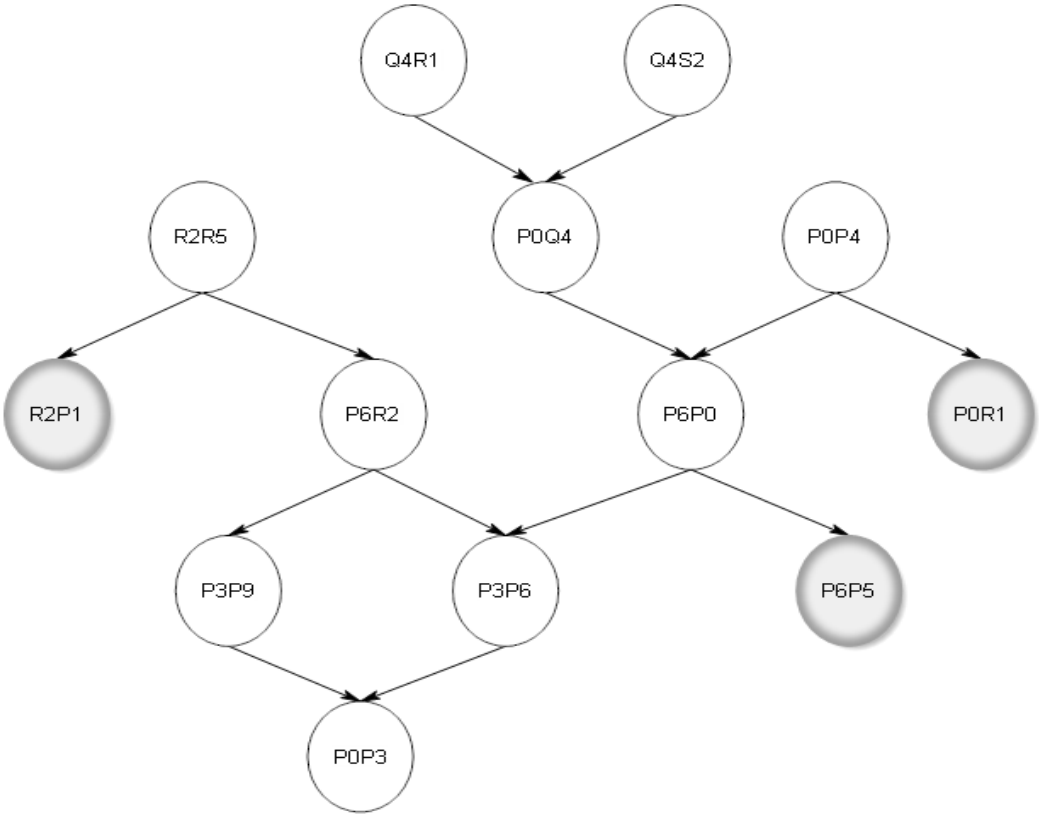

Figure 5.12: Lineage Patterns in documents

CHAPTER 6
CONCLUSION

Our thesis is finally concluded in this chapter where we emphasize on what can be done based on this thesis. We have presented our experimental result for two document classification algorithm.For same number of observations and clusters we performed cross-validation test on each algorithm to generate the correction rate of the classifier and based on that result we made a comparison. Then we developed a route through the classified observations for providing theexplanation which may improve the acceptance of the decision-maker. Thus we developed trace routes among documents to improve the explanation. In future we can try to find out a way to obtain better classification.

REFERENCES

[1] Coping with complex data. the forrest report. In *Forrester Research*, 1995.

[2] S.L. Alter. Decision support systems: Current practice and continuing challenge. In *MA*. Addison-Wesley, 1988.

[3] A Banerjee, S Merugu, I Dhillon, and J Ghosh. Clustering with bregman divergences. *J Mach Learn Res*, 6:1705–1749, 2005.

[4] JC Bezdek, SK Chuah, and D Leep. Generalized k-nearest neighbor rules. fuzzy sets syst. *34th symposium on computing and statistics*, 3:237–256, 1986.

[5] Zhi Chen. The effects of documents lineage on use of explanation in document-driven dss.

[6] S Cost and S Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. In *Mach Learn*, 1999.

[7] T Cover and P Hart. Nearest neighbor pattern classification. *IEEE Trans Inform Theory*, 1:21–27, 1967.

[8] BV Dasarathy. Nearest neighbor (nn) norms: Nn pattern classification techniques. In *IEEE Computer Society Press*, 1991.

[9] F. Davis and J. Kottemann. Determinants of decision rule use in a production planning task. In *Organizational Behavior and Human Decision Processes*, volume 63, pages 145–157, 1995.

[10] L Devroye, L Gyorfi, and G Lugosi. A probabilistic theory of pattern recognition. In *Springer, New York*, 1996.

[11] IS Dhillon, Y Guan, and B Kulis. Kernel k-means: spectral clustering and normalized cuts. *KDD*, pages 551–556, 2004.

[12] M. J. Druzdzel and R. R. Flynn. Decision support systems. In *Encyclopedia of Library and Information Science*. Allen Kent (ed.), Marcel Dekker, Inc., 1999.

[13] P. Finlay. Introducing decision support systems. In *Oxford:Blackwell*, 1994.

[14] E Fix and JL Hodges Jr. Discriminatory analysis, nonparametric discrimination. *USAF School of Aviation Medicine, Randolph Field*, 4:128–131, 1951.

[15] M. Sinan Gonul. The effects of structural characteristics of explanations on use of a dss. *Decision Support Systems*, 42, 2006.

[16] M. Sinan Gonul. The effects of structural characteristics of explanations on use of a dss. In *Decision Support Systems*, 2006.

[17] RM Gray and DL Neuhoff. Quantization. *IEEE Trans Inform Theory*, 4:2325–2384, 1998.

[18] E Han. Text categorization using weight adjusted k-nearest neighbor classification. In *PhD thesis, University of Minnesota*, 1999.

[19] P Hart. The condensed nearest neighbor rule. *IEEE Trans Inform Theory*, 14:515–516, 1968.

[20] AK Jain and RC Dubes. Algorithms for clustering data. In *Prentice-Hall, Englewood Cliffs*, 1988.

[21] P.G.W. Keen. Decision support systems: A research perspective, in decision support systems: Issues and challenges. In *Pergamon Press*, 1981.

[22] HANA KOPACKOVA. Decision making with textual and spatial information. March 2008.

[23] M Kuramochi and G Karypis. Gene classification using expression profiles: A feasibility study. *Int J Artif Intell Tools*, 4:641–660, 2005.

[24] M. Lawrence, L. Davies, M. O'Connor, and P. Goodwin. Improving forecast utilization by providing explanations. In *21st International Symposium on Forecasting, Atlanta-USA*, 2001.

[25] J.S. Lim and M.J. O Connor. Judgmental adjustment of initial forecasts. In *Journal of Behavioral Decision Making*, volume 8, pages 149–168, 1995.

[26] P.Goodwin, D. nkal Atay, M.E. Thomson, A.C. Pollock, and A. Macaulay. Feedback-labelling synergies in judgmental stock price forecasting. In *Decision Support Systems*, volume 37, pages 175–186, 2004.

[27] D. J. Power. Decision support systems: Concepts and resources for managers. In *Westport, CT: Greenwood/Quorum Books*, 2002.

[28] D. J. Power. A brief history of decision support systems. In *DSSResources.COM*, 2007.

[29] D.J. Power. A brief history of decision support systems. 1999.

[30] L. Richard, Ye. Paul, and E. Johnson. The impact of explanation facilities on user acceptance of expert system advice. 1997.

[31] A. Schroff. An approach to user oriented decision support systems, inaugural dissertation nr. 1208. In *Druckerei Horn, Bruchsal*, 1998.

[32] Lloyd SP. Least squares quantization in pcm. unpublished bell lab. *IEEE Trans Inform Theory (Special Issue on Quantization)*, 28:129–137, 1982.

[33] M Steinbach, G Karypis, and V Kumar. A comparison of document clustering techniques. In *Proceedings of the KDD Workshop on Text Mining*, 2000.

[34] W. R. Swartout. Explanation.in s. c. shapiro. *Encyclopedia of artificial intelligence, New York Wiley*, 1:298–300, 1990.

[35] P-N Tan, M Steinbach, and V Kumar. Introduction to data mining. In *Pearson Addison-Wesley*, 2006.

[36] R.L. Teach and E.H. Shortliffe. An analysis of physicians attitudes. In *Computers in Biomedical Research*, volume 14, pages 542–558, 1981.

[37] GT Toussaint. Open problems in geometric methods for instance-based learning. *JCDCG*, pages 273–283, 2002.

[38] GT Toussaint. Proximity graphs for nearest neighbor decision rules: recent progress. *34th symposium on computing and statistics*, 2:17–20, 2002.

[39] DL Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans Syst Man Cyberne*, 2:408–420, 1972.