Western Kentucky University Year 2009

A FRAMEWORK FOR CONSISTENCY BASED FEATURE SELECTION

Pengpeng Lin
Western Kentucky University, pengpeng.lin574@wku.edu

A FRAMEWORK FOR CONSISTENCY BASED FEATURE SELECTION

A Thesis

Presented to

The Faculty of the Department of Mathematics & Computer Science

Western Kentucky University

Bowling Green, Kentucky

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Pengpeng Lin

May 2009

A FRAMEWORK FOR CONSISTENCY BASED FEATURE SELECTION

Date Recommended__May 05. 2009_________

___Huanjing Wang_________________
Director of Thesis

___Art Shindhelm_________________
Committee Member

___Qi Li_________________________
Committee Member

____
Dean, Graduate Studies and Research Date

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Huanjing Wang, for her continuous support. Dr. Wang was always there to listen and give me advice. She is responsible for leading me into the data mining area in the first place. She showed me the different ways to approach a data mining problem and the persistence to accomplish the research. I specially thank Dr. Art Shindhelm who taught me to look into a problem in every aspect in CS 541 (Mathematical Foundation of Computer Science), made me a better programmer in CS 544 (Compiler Theory and Design), had confidence in me when I doubted myself, and constantly gave me encouragement when I was anxiously waiting for my PHD admission.

I also would like to thank my committee member, Dr. Li, who offered guidance on various occasions. I would like to thank my parents who have been supportive of my higher education. I am also greatly indebted to my best friend, Gyuchoon Cho, who showed me a whole new area of programming. Gyuchoon is very experienced as a programmer, very responsible as person and very trustworthy as a friend. Last but not least, I wish to thank everyone who helped me, not only on my thesis and graduate study, but during my stay at WKU with pleasant memories.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

A FRAMEWORK FOR FEATURE SELECTION

Pengpeng Lin May 2009 48pages

Directed by: Huanjing Wang

Department of Mathematics & Computer Science Western Kentucky University

Feature selection is an effective technique in reducing the dimensionality of features in many applications where datasets involve hundreds or thousands of features. The objective of feature selection is to find an optimal subset of relevant features such that the feature size is reduced and understandability of a learning process is improved without significantly decreasing the overall accuracy and applicability. This thesis focuses on the consistency measure where a feature subset is consistent if there exists a set of instances of length more than two with the same feature values and the same class labels. This thesis introduces a new consistency-based algorithm, Automatic Hybrid Search (AHS) and reviews several existing feature selection algorithms (ES, PS and HS) which are based on the consistency rate. After that, we conclude this work by conducting an empirical study to a comparative analysis of different search algorithms.

CHAPTER 1: INTRODUCTION

The chapter provides the definition of data mining with several typical examples and gives a brief introduction to the four steps of the mining process: data preprocessing, data mining, pattern evaluation and knowledge presentation. Two aspects of data mining functionalities are described which are descriptive and predictive. The idea of feature selection and the concept of evaluation criterion is introduced. At the end of the chapter, we make a conclusion for the work done in this thesis and give a concise introduction to the contents of the rest of this thesis.

1.1 WHAT IS DATA MINING

By definition, data mining is a process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses, or other information repositories. Data as the target for data mining has skyrocketed dimensionally in the size and category each year. Large parts of those data are with inferior qualities such as irrelevancy, redundancy and unreliability. Data mining implements various algorithms on such data and tries to dig useful information by looking at a small fraction of a large amount of data. For instance, the Midwest Grocery Chain once used the data mining capacity of Oracle to analyze the local buying patterns. They discovered that when men bought diapers on Thursdays and Saturdays, they also tended to buy beers. This buying pattern could be used by the retailers. For example, beers could be displayed next to the diapers and they could be sold at full price on Thursdays and Saturdays

1.2 STEPS OF DATA MINING

According to [9], data mining consists of the following four steps:

1. Data preprocessing (where the data are prepared for mining);
2. Data mining (an essential process where intelligent methods are applied in order to extract data patterns);
3. Pattern evaluation (to identify the truly interesting patterns or evaluate the data's merits);
4. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user).

Data preprocessing can be divided into four steps: data cleaning, data integration, data selection and data transformation. Data cleaning is to remove noise and inconsistent data. Data integration combines multiple data sources. Data selection retrieves those data relevant to the analysis from database. Data transformations transform the data into appropriate forms for mining. After data preprocessing, the data is clear to go through data mining where a pattern may be discovered. In order to identify the pattern's value, some merit measurements are employed during pattern evaluation. Finally, the results are visualized and presented to users.

1.3 DATA MINING TASKS

Data mining tasks are used to specify the kind of patterns to be found in data mining [9]. It can be broadly unfolded into two aspects: descriptive and predictive. Descriptive mining tries to summarize the general properties of data, whereas predictive mining tasks try to make prediction on current data’s changing patterns or trends. A data mining system should be able to produce a general description of the characteristics of the target class. For example, when executing a conditional selection query such that selecting employees who’s monthly salary is above $7000, the data that meet the condition should be collected by mining the data characteristics. The target class of mining data can also be discriminated by a comparison between other features’ characteristics, or values. A prediction can then be made on the values of target based on the data discrimination. For example, people working for the weather bureau can collect data and analyze them in order to predict weather.

1.4 DATA REDUCTION

Data reduction as one of the data preprocessing techniques obtains a reduced representation of a dataset that is much smaller than the original dataset and similar data mining results can be obtained with the reduced dataset [10].

Data reduction includes:

- Feature selection, keeping only useful features and removing irrelevant and noisy information, at the same time improving efficiency without significantly reducing the accuracy of the classifier;
- Reducing the number of attribute values by grouping them into intervals or grouping values in cluster;
- Reducing the number of records.

1.5 FEATURE SELECTION

The raw data (data without proper data preprocessing) are usually with inferior qualities such as irrelevancy and redundancy, unreliability. Mining raw data for a given task has proved to be deteriorating in efficiency. There are many algorithms in data mining which can be used in dealing with redundant, irrelevant, or missing data. Before the mining process begins, those instances with missing value are considered insignificant to the result and thus can be eliminated from the dataset. The most challenging thing is to find out and eliminate redundant and irrelevant features. As part of the data preprocessing procedure, feature selection uses certain algorithms to keep only useful features and remove irrelevant and noisy information, at the same time improving efficiency without significantly reducing the accuracy of the classifier. Figure 1 shows the basic structure of feature selection.

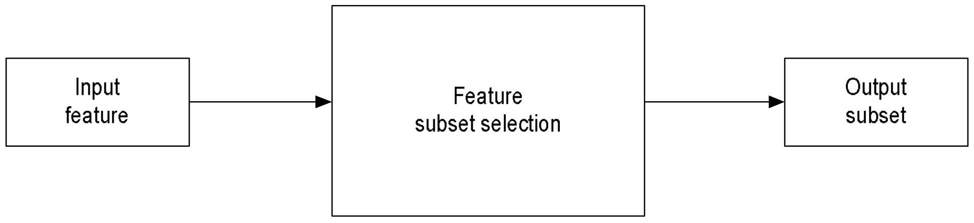


Figure 1. Basic structure of feature selection

1.6 EVALUATION CRITERION

In order to select a subset of relevant features, an evaluation criterion must be defined. Feature subset is selected using a search algorithm. After that, evaluation criterion is used to measure the goodness of selected features. A merit value is calculated for each selected feature subset. During the process of non-random searching feature selection, a threshold is usually setup at the beginning. After evaluation, the data with merit less than the threshold is eliminated. Those selected features are combined with the remaining features to form new patterns and undergoing another selection by evaluation criterion where some of them will again be eliminated. This process repeats itself until the desired candidate subsets (this is, several subsets with the same desired merit value) are selected. For probabilistic searching algorithms, a run time is specified at the beginning and feature subset is generated during each run time. The result obtained at the first running time is set to be the best feature subset and if later another feature subset with better merit value by evaluation criteria is obtained, the previous best feature subset is replaced. The probabilistic searching stops when it reaches a specified run time.

1.7 FEATURE SELECTION MODELS

Feature selection algorithms designed with different evaluation criteria generally fall into three categories: filter model, wrapper model and hybrid model [3]. The filter model evaluates feature subsets based on general characteristics of data without involving any algorithm. The wrapper model requires evaluation criterion with a predetermined learning algorithm. The hybrid model is a combination exploiting advantages from both filter and wrapper model. The model for the new algorithm presented in this thesis is classified as a hybrid model since it not only employs consistency measure as an evaluation criterion which evaluates each subset by its general characteristics but also involves a learning process.

1.8 OVERVIEW

In this thesis, we review an evaluation criterion called the consistency measure. Using the consistency measure, feature selection is formalized as finding the small set of features that can distinguish classes as if with the full set [1]. Three exiting algorithms (Exhaustive Search (ES), Heuristic Search (HS) and Probabilistic Search (PS)) are depicted, examined and implemented and a new algorithm is proposed based on them. The Exhaustive Search (ES) or Heuristic Search (HS) such as $FocusM$ and $SetCover$ [1] starts either from an empty feature subset or a full set. New subsets are selected by either adding or deleting one feature at a time. Other search strategies such as probabilistic or random search begin the searching process with an arbitrary feature set – an example is LVF [1]. The new searching algorithm is named Automatic Hybrid Search (AHS) and is implemented. In Chapter 3, we give more details about the consistency rate evaluation. We introduce an important property of the consistency rate: monotonic property. We describe step by step how to compute the consistency rate and give its formula. In Chapter 4, we elaborate on the three most typical feature selection search algorithms which are ES, HS, and PS. In Chapter 5, a new search algorithm named AHS is introduced and explained in detail. In Chapter 6, we record our empirical results for the new algorithm and other aforementioned algorithms and give a comparative analysis for their performances. In the conclusion, we summarize our work done for this thesis, and make a blueprint for further researc

CHAPTER 2: FEATURE SELECTION

2.1 FEATURE SELECTION PROCESS

Feature selection is to find a subset of features according to the given evaluation criterion. Each feature subset is a feature combination. By evaluating each selected subset, we can reduce the number of possible feature combinations by eliminating the irrelative features and thus simplifying the classifier. Many existing evaluation criteria are accurate only for discrete data. A preprocessing step is taken to discretize data values before applying these criteria. In this thesis we use discretized data.

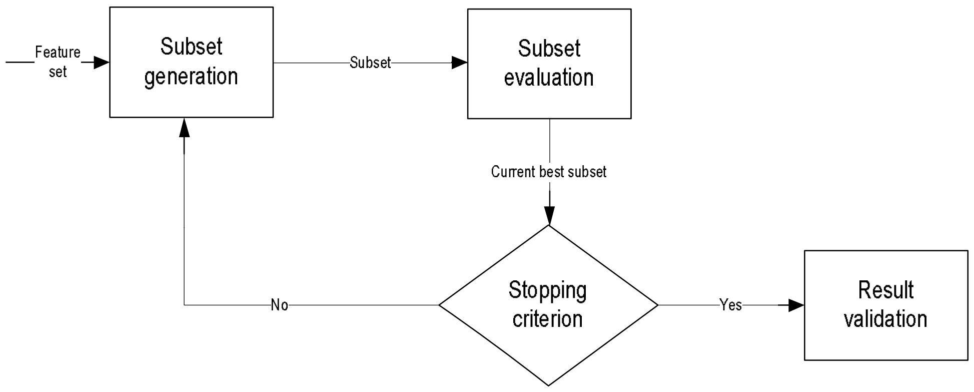


Figure 2. Traditional feature selection process

Traditionally, feature selection consists of four parts [1] (Figure 2): a subset generation procedure, an evaluation function or criterion, a stopping criterion and a validation procedure. Subset generation procedure usually uses certain searching strategy to produce candidate feature subset. Each selected subset is evaluated by a criterion for its merit and is compared with the previous best result. If the new selected subset has better merit than the previous best result, the previous best subset is replace by the new subset. The process of subset generation and evaluation is repeated until a stopping criterion is met. In the end, the validation procedure tests the validity of the selected subset. In general, we need a search algorithm sweeping through the feature sets' and an evaluation function selecting qualified features. Given a feature subset S for a data set D, the evaluation function F generates the goodness G of that subset. The equation (1) is the relation between search algorithms and evaluation functions.

$$\text{Evaluation function:} \qquad G \leftarrow F(S) \qquad (1)$$

2.2 SEARCH PROCESS

The search process usually focuses on three aspects: where to start, how to produce the next candidate subset and when to stop. Based on these three aspects, the search strategies can be categorized as: complete, sequential and random search [3]. A complete search is also called an exhaustive or naïve search and generates every possible combination of subsets. A complete search is accurate thus can produce an optimal result at the cost of computation expenses. Some examples are $FoucsM$, ABB [1]. Sequential search adds or deletes one or n features at a time where n is less than the number of features in the dataset. The sequential search does not check every possible combination of subsets and risks losing an optimal subset. They are preferable in applications where efficiency is an important matter. Some examples are $SetCover$ or QBB [1]. Random search randomly selects the subset. The selected subset combined with the previous best subset in order to produce better results. The use of randomness helps to escape local optimal in the search space, and optimality of the selected subset depends on the resources available [3].

2.3 EVALUATION

Evaluation criterions are focused on selecting relevant features and eliminating irrelevancy and redundancy. As described in [4], the definitions of feature relevance are classified into three categories: strongly relevant, weakly relevant and irrelevant. If a feature is strongly relevant, it indicates that the feature is to determine an optimal subset. Removing it will affect the class distribution. Weak relevant features are not always needed to obtain optimal subset. Irrelevant features should not be considered and are removed by evaluation criteria. In this thesis, we focus on a popular evaluation criterion called the consistency measure.

CHAPTER 3: EVALUATION CRITERION

3.1 INCONSISTENCY RATE

To be consistent and concise, we denote $CCON$ as the consistent count, INC as the inconsistent count, $CCONR$ as the consistency rate, $INCR$ as the inconsistency rate.

The goal of feature selection is to find as minimum as possible a number of feature subsets that can consistently discriminate classifier as if using the full set of features. The consistency measure as an evaluation criterion is used to determine which feature should be eliminated and thus reducing the size of the feature set. As depicted in [1], the consistency rate is defined by the an inconsistency rate where two instances are considered inconsistent if they have the same feature values but different class labels. To compute the inconsistency rate, first we have to compute inconsistency count. Assuming that the target feature has three different values (labels): T_1, T_2, T_3. For a feature subset S with M number of instances, there are h number of patterns such that $M = P_1+P_2 ... + P_h$. A pattern P_i ($1 \leq i \leq h$) appears in totally N instances out of which N_1 number of instances are labeled T_1, N_2 labeled T_2 and N_3 labeled T_3 where $N = N_1+N_2+N_3$. If N_1 is the largest among the three, the inconsistency count for the pattern P_i is $INC_i = N - N_1$. The inconsistency rate is the summation of all the inconsistent counts over all patterns divided by total number of instances in the dataset. The inconsistency rate $INCR$ can be expressed as (1).

$$INCR = \frac{\sum_{i=1}^{h} INCi}{M} \qquad (2)$$

In earlier works such as [1] and [2], the inconsistency rate is applied into the search algorithms. A threshold R is usually defined at the beginning. For each feature subset F selected by the search algorithm, the inconsistency rate $INCR$ is calculated. If the result meets the condition $INCR \leq R$, then the subset F is considered to be consistent. The original threshold R is updated by better results. Otherwise, the subset is eliminated along with its features.

3.2 CONSISTENCY RATE

In this work, we use the consistency rate instead of inconsistency rate. The consistency rate is similar to inconsistency rate except that consistent count $CCON$ is computed. Instead of subtracting N_1 from N to get INC, we consider N_1 as consistent count ($CCON$). Thus the consistency rate can be expressed as (2):

$$CCONR = \frac{\sum_{i=1}^{h} CCONi}{M} \qquad (3)$$

The purpose of using the consistency rate is to reduce the computation overhead. By comparison between (2) and (3) above, the consistency rate has one step less to compute than that of inconsistency rate and yet both of them are able to evaluate the merits of a feature subset.

3.3 MONOTONIC PROPERTY

Consistency rate has the monotonic property. An evaluation criterion is monotonic if for a data D and a measure M, there exists feature sets S_i and S_j where $S_i \subseteq S_j$, then $M(S_i, D) \leq M(S_j, D)$. The monotonic property of consistency rate can be used in feature selection. For example, at the beginning, we consider full feature set as the optimal feature set and calculate consistency rate δ for it. According to the monotonic property, no feature subset can have a larger consistency rate than the full feature set. For each generated feature subset, if the corresponding $CCONR$ is equal to δ and the size of the feature subset is smaller; we say that this feature subset can be used for classification as if using the full feature set.

The property gave us the following facts:

- The full feature set has the highest consistency rate δ, that is, the consistency rate of any feature subset is less than or equal to δ.
- The superset of a consistent feature subset is also consistent.
- If $CR(S_i, D) \leq CR(S_j, D)$, then $CR(S_i \cap f, D) \leq CR(S_j \cap f, D)$ where f is a single feature.

CHAPTER 4: SEARCH METHODOLOGIES

Search techniques are very important. A good search strategy can reduce the computational cost while improving the accuracy at the same time. According to different searching nature, search strategies include exhaustive search (ES), heuristic search (HS), probabilistic search (PS) and hybrids of the above three.

The relation between searching techniques and evaluation measures can be illustrated in *Figure 3.*

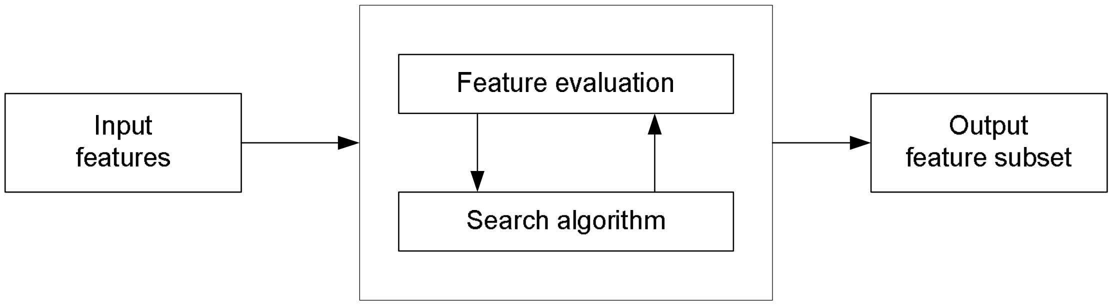


Figure 3. Search algorithm

4.1 EXHAUSTIVE SEARCH (ES)

The exhaustive search uses the algorithm to generate every possible combination of feature subset and compute consistency rate for each of them. A threshold is set up at the beginning according to the consistency rate calculated for the first selected feature subset which is also set to be the current best subset. As the algorithm proceeds, the current best subset may be replaced by the one with higher consistency rate and smaller size of feature set in which case the threshold is also replaced. The exhaustive search can start with either a set with one feature and carry out by adding features into set or with a full feature set and subtract features from it.

It is evident that exhaustive search is time consuming and computational expensive as it calculates every combination many of which may be redundant and could be avoided if a learning process is involved. The efficiency is deteriorating fast with the size of the search space. For example, if a testing data has n features, the number of combinations is $\sum_{i=1}^{n} C_n^i$ which means that there will be $\sum_{i=1}^{n} C_n^i$ times of calculation for consistency rate. In conclusion, the exhaustive search is inefficient and costly for a high feature size. An example for the exhaustive search is $Focus$, or $FocusM$.

Below is the exhaustive search algorithm where $ConCal()$ is a function to calculate the consistency rate for a given dataset D. S is the full feature set for the D.

Algorithm: Exhaustive Search

Input: Data D, Feature set S

Output: Consistent Feature Subset

```
1.  $BestSet = \emptyset$
2.  $BestConsistencyRate = -\infty$
3.  $For\ i = 1\ to\ |S|$
4.  $Begin$
5.      $\forall S'\ where\ |S'| = i\ and\ S' \in S$
6.      $CCONR = ConCal(S', D)$
7.      $If\ CCONR \geq BestConsistencyRate$
8.          $If\ |BestSet| < |S|$
9.              $BestSet = S'$
10.         $BestConsistencyRate = CCONR$
11. $End$
12. $Return\ BestSet$
```

Figure 4. Exhaustive search algorithm (ES)

4.2 HEURISTIC SEARCH (HS)

There are two fundamental goals in computer algorithms: finding a way to use a shorter running time and to produce an optimal solution. A heuristic algorithm is used when there is no known way to find an optimal solution in which case the goal is to develop a simple process with a provable better running time and an improved solution. Since exhaustive search algorithms take a significant amount of unnecessary time and computationally costly, the heuristic algorithm is a good alternative to quickly conduct and return a decent result.

There are many heuristic search techniques in practice such as $SetCover$. $SetCover$ is depicted in [1] to be the most time efficient, close to optimal and deterministic heuristic algorithm in comparison to genetic algorithms and simulated annealing. The original idea for $SetCover$ is that two instances with different class labels are said to be "covered" when there exists at least one feature which takes different values for the two instances [6]. In other words, two instances with two different class labels are considered to be consistent if two instances have at least one distinctive feature value between them. This is actually a very similar concept to consistency rate in the inverse order. For consistency rate, the two instances with the same feature pattern are considered to be consistent if they have identical value for each of their features.

In the thesis, we apply Johnson's algorithm [7] with the consistency rate as the evaluation. The following is its pseudo-code

Heuristic Search Algorithm

Input: Data D, full feature set S

Output: Consistent Feature Subset

```
1.    $BestConRate = ConCal(S, D)$
2.    $SelectedFeatureSet = \emptyset$
3.    $S' = \emptyset$
4.    $LargestConRate = -\infty$
5.    $\forall\ feature\ f \in S$
      $begin$
6.        $S'' = S'\ appends\ f$
7.        $TempConRate = ConCal(S'', D)$
8.        $If\ \ TempConRate = BestConRate$
9.          $Return\ S''$
10.       $Elseif\ TempConRate > LargestConRate$
11.         $LargestConRate = TempConRate$
12.         $SelectedFeatureSet = f$
13.       $S' = S'\ append\ f$
14.       $S = S - f$
      $end$
```

Figure5. Heuristic search algorithm (HS)

4.3 PROBABILISTIC SEARCH (PS)

As aforementioned, a traditional feature selection process consists of four components: generation procedure, evaluation function, stopping criterion and validation procedure. The third component, stopping criterion, usually has a stopping point by which an algorithm's completion is determined. For feature selection, the stopping point is reached when the smallest feature subset with the highest merit is found and thus the search algorithm stops searching process. In a probabilistic search, the stopping criterion can be defined in many ways. Some probabilistic search techniques combine with heuristic search algorithms to identify the most useless features in each iteration and generate a better candidate subset by eliminating the weakest feature from a combination list. The probabilistic searches are often given a number as a parameter to specify how many times the search is going to run. During each iteration, a new subset is generated randomly from the remaining features which had the most useless one trimmed off during the last iteration. As the time of the iteration increases, the accuracy is achieved at the cost of a more expensive computational time. A typical example is genetic algorithm (GA) where the stopping criterion is set to stop the algorithm at a certain amount of running times. Genetic algorithm adopts the idea of mutation and crossover from biology to generate new candidate feature sets and uses a fitness function to evaluate their merit. The consistency rate can be used as a fitness function for genetic algorithm and it has to be given an integer for the number of generations as a stop criterion. For example, if the number of generations for GA is set to 11, the program will stop on running the algorithm 11 times. The number of generations should be predefined and sufficient to obtain a good result. Some people tend to set the number of generations to be much more than necessary in order to get an optimal result. Others trade off accuracy for efficiency.

Other probabilistic searches employ two stopping criterions combined. One is the generation time, the other one is a traditional stopping point where the searching algorithm is stopped when the best subset is found. Such a combination guarantees accuracy and the search is stopped not only based on running times but also on the result, this combination effectively avoids unnecessary lavish expenditures. As the probabilistic search proceeds, the feature subsets are randomly generated with equal probability, once a consistent feature subset is selected that satisfies the threshold, the search will stop regardless of the specified running time. In other cases where data set may have a large number of features and instances, if the general purpose is to find a result with a certain amount of tolerance, setting the running time as the main stopping criterion is the most reasonable way.

In the following, we implemented a probabilistic search algorithm called LVF (Las Vegas Filter) [8] as an example.

Probabilistic Search

Input: Data D, feature-set S, Running-Time T

Output: Consistent Feature Subset

```
1.   $BestConRate = ConCal(S, D)$
2.   $TempSet = S$
3.   $For\ i = 1\ to\ T$
4.       $Begin$
5.         $S' = Random(S)$
6.         $If\ |S'| \leq |TempSet|$
7.             $If\ ConCal(S', D) = BestConRate$
8.                 $If\ |S'| < |TempSet|$
9.                     $TemsSet = S'$
10.                $else$
11.                    $TempSet = TempSet\ appends\ S'$
12.      $End$
13.  $Return\ TempSet$
```

Note: The function $Random()$ above randomly chooses a subset from S and return it.

Figure 6. Probabilistic search algorithm (PS)

CHAPTER 5: PROPOSED ALGORITHM (AHS)

5.1 PROPOSED ALGORITHM

The proposed algorithm is based on the aforementioned search algorithms. The full feature set has the highest consistency rate. It can also be proven by its monotonic property that a superset of a consistent feature subset is also consistent. This guarantees that the best consistency rate can be calculated from the beginning and hence helps setting up the threshold for a stopping criterion. However, our goal is more than to just calculate the highest consistent rate. Reducing the dimensionality of a data set becomes increasingly important since today's data expands exponentially. Classification to the large data depends heavily on feature selection techniques to perform quickly and make an accurate result.

The algorithm works as follows: the consistency rate δ of full feature set is computed at the beginning and serves as a threshold. The searching algorithm starts from a feature subset of size 1. The searched features with local highest consistency rate are selected. These selected features will be used to generate a super set. This process repeats itself until it finds feature subsets that have consistency rate equals to δ or the full feature set is reached. If more than one feature subset is generated (multiple selected subsets with consistency rate δ), a classifier called C4.5 [11] will be used to discriminate the selected feature subsets again by their error rate.

Automatic Hybrid Search

Input: Data D, feature-set S;

Output: consistent feature subsets L, Consistent Feature Sub-Set

```
1.  $L = S$ ;
2.  $\delta \leftarrow ConCal(S, D)$ ;
3.  $T \leftarrow \forall S'$ $in$ $S$ $where$ $|S'| = 1$ ;
4.  max $\leftarrow -\infty$ ;
5.  $while \forall |T'| in\ T\ and < |S|$ ;
6.      $tempSet = \emptyset$ ;
7.      $for\ each\ T'\ in\ T$ {
8.          $tempCal = ConCal(T', D)$
9.          $if\ (max < tempCal)\ then$ {
10.             $max = tempCal$ ;
11.             $tempSet = \emptyset$ ;
12.             $add\ T' to\ tempSet$ ;
13.         }
14.         $if\ (max == tempCal)\ then$
15.             $add\ T' to\ tempSet$ ;
16.         }
17.     $if\ (max \geq \delta)\ then$ {
18.         $L = tempSet$ ;
19.         $return\ L$ ;
20.         }
21.     $else\ if\ (|tempSet| == |T|)\ then$ {
22.         $T \leftarrow CombinationSet(T,\ size + 1)$ ;
23.         }
24.     $else$ {
25.         $for\ any\ set\ tempSet' in\ tempSet$
26.             $append\ tempSet' with\ f\ where\ f\ is\ any$
                $feature\ ins\ S, not\ in\ tempSet'$ ;
27.         $T = tempSet$;
28.         }
29.     }
30. Return L;
```

Figure 7. Automatic Hybrid search algorithm (step1)

Step2:

Input: L, consistent feature subsets from step1.

Output: T, selected feature subset.

Method:

```
1.  $\min = \infty$ ;
2.  $T = \emptyset$ ;
3.  $for\ each\ feature\ subset\ L'in\ L$ {
4.      $calculate\ error\ rate\ r\ using\ C4.5\ with\ L'$ ;
5.      $if(r < min)then${
6.          $min = r$ ;
7.          $T = L'$
8.      }
9.  }
10. $return\ T$ ;
```

Figure 8. C4.5 error rate checking (step2)

5.2 AHS VS. HS

AHS and HS are both using the monotonic property to setup the threshold from the onset. The full feature set is used in both algorithms to generate the best merit possible by the consistency rate evaluation criterion. The most obvious distinction between AHS and HS is that HS only selects one feature subset in each iteration whereas AHS selects all the feature subsets with the same highest merit. For example, if in one iteration, N feature subsets are selected according to the consistency rate measure, HS will only pick the first one. AHS, on the other hand, picks all the qualified candidates. After the selection, the selected feature patterns are kept and used to combine the remaining features. AHS keeps more patterns than HS in each loop since AHS uses the entire feature subset. This greatly reduced the further possible computation cost since there are less features left to generate new combinations in AHS. AHS is also more accurate than HS. If the result contains more than one feature subsets, *C4.5* is used to check their error rate. The one with the best result is chosen and others are eliminated. In conclusion, AHS is more accurate in comparison to HS and able to reduce computational costs.

5.3 FLOW CHART OF AHS

AHS returns the entire qualified feature subset and involves more of a learning process. It is more complex to implement since it has more conditions to consider. The following is the flow chart for it.

T is a feature set holder that initially contains the whole features;

L is the consistent feature subsets;

D is dataset;

Max is initialized $-\infty$;

Q is the best consistency rate calculated by full set

$tempSet$ is a temporary feature set holder

S is the full feature set

Figure 9. Notations of AHS flow chart

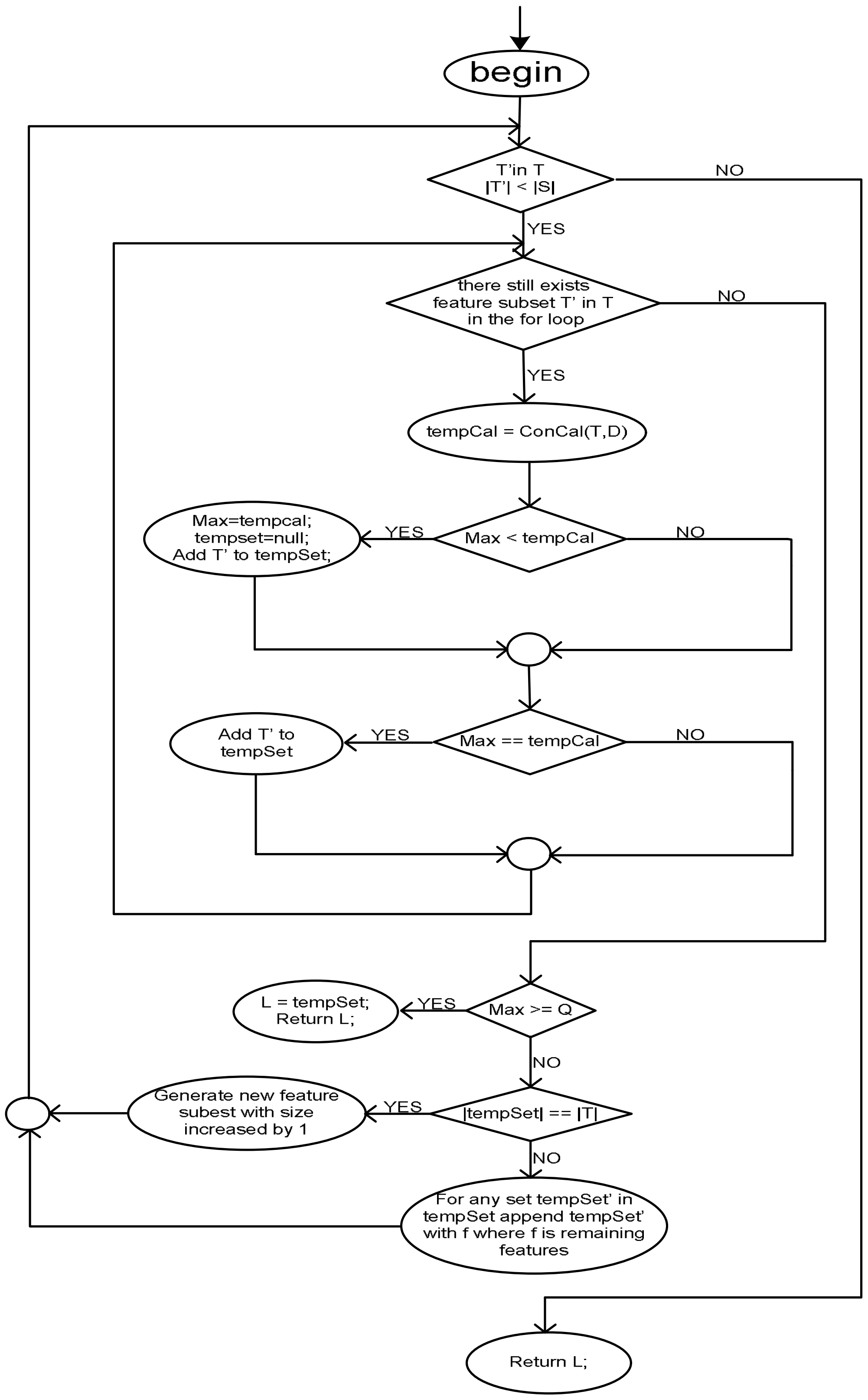

begin
T'in T
|T'| < |S|
NO
YES
there still exists feature subset T' in T in the for loop
NO
YES
tempCal = ConCal(T,D)
Max=tempcal; tempset=null; Add T' to tempSet;
YES
Max < tempCal
NO
Add T' to tempSet
YES
Max == tempCal
NO
L = tempSet; Return L;
YES
Max >= Q
NO
Generate new feature subest with size increased by 1
YES
|tempSet| == |T|
NO
For any set tempSet' in tempSet append tempSet' with f where f is remaining features
Return L;

Figure 10. AHS flow chart

CHAPTER 6: EXPERIMENTAL EVALUATION

6.1 DATE SET DESCRIPTION

In order to test the performance and accuracy of the AHS algorithm and make a comparison to the algorithms we discussed in chapter 4 and chapter 5, we selected three typical datasets. They are the Credit Approval dataset [12], the SPECT Heart dataset [13] and the Weather dataset [14].

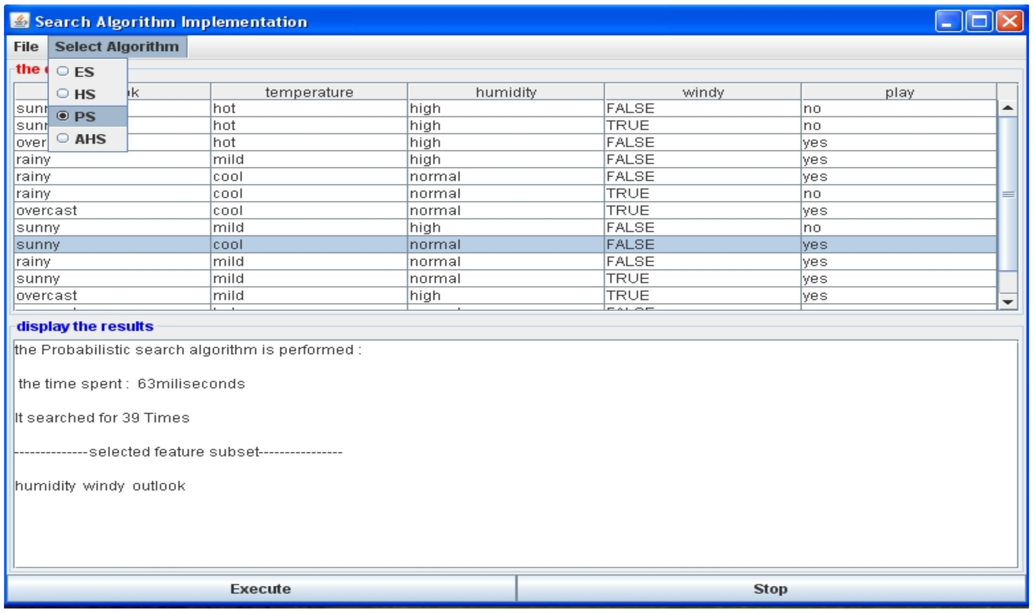
The Credit Approval dataset is interesting because it contains a mix of attributes - continuous, nominal with small numbers of values, and nominal with larger numbers of values. This fact allows us to test whether our program can implement on different attribute values. There are also a small amount of instances with missing data which are removed because dealing with missing data is out of the scope of this thesis. The Credit Approval dataset is about credit card applications. To protect the confidentiality, the domain for each attribute is replaced by meaningless symbols. The SPECT Heart dataset describes diagnosing of Cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The Weather dataset is small but a very typical dataset which has been used as sample dataset by many major data mining tools such as WAKA. The target attribute is 'play' which consists of two classes: no and yes. All three datasets are described in the table 1 below.

Name	# of instances	# of features
Credit Approval	673	15
SPECT Heart	80	22
Weather	15	4

Table 1. Experimental datasets

6.2 COMPARISONS

Figure 11 shows the application interfaces where the four search algorithms are implemented using Java. The four algorithms are compared with running time and accuracy. Ten runs were done for each search algorithm and the running time is based on their average outcomes. From table 2, we can see that ES is more time consuming than HS, PS and AHS. AHS took a bit longer than HS and PS since AHS is a hybrid algorithm and involved learning and selecting feature subset. The execution time differences are insignificant between the four algorithms when the feature size is small and significant when the feature size is large i.e. tens or hundreds of thousands of features.



k	temperature	humidity	windy	play
sunr	hot	high	FALSE	no
sunr	hot	high	TRUE	no
over	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes

```
the Probabilistic search algorithm is performed :

 the time spent :  63miliseconds

It searched for 39 Times

--------------selected feature subset---------------

humidity  windy  outlook
```

Figure 11. Interface to implement different search algorithms

Feature Selection Methods	Dataset		
	Credit Approval	SPECT Heart	Weather
ES	13761219	27110657	31
HS	2043	1109	25
PS	4931	226	52
AHS	4234	24562	29.8

Table 2. Run-time comparisons (in milliseconds)

Feature Selection Methods	Dataset		
	Credit Approval	SPECT Heart	Weather
ES	3	11	3
HS	3	15	3
PS	4	15	3
AHS	3	11	3

Table 3. Size of consistent feature subset

Table 3 shows the number of features selected by each algorithm. The result for AHS is the same as ES for all dataset. As ES searches for every possible dataset combination, the result by ES is usually more accurate. This again shows us that the AHS can get more accurate results with much less time overhead.

We also employed a well known classification algorithm C4.5 [11] to evaluate the degree that each result affects the classification and their error rate. Table 4 shows the error rate of the decision tree for each feature selection method. As seen in the empirical results, AHS performed the same as HS and PS with smaller size of feature subset.

Method	**Error rate**
ES	21.25%
HS	18.75%
PS	18.75%
AHS	18.75%

Table 4. Results for the C4.5 Algorithm

CHAPTER 7: CONCLUSIONS

In this thesis, we have reviewed the framework of feature selection and explained the basic concept of different feature selection model: filter, wrapper and hybrid model. After that, we gave a brief review on several evaluation criterions and centered our attention on the consistency rate measurement. We examined its properties such as the monotonic property. Three typical search algorithms are investigated in this thesis which are : the exhaustive (ES), the heuristic (HS) and the probabilistic search (PS). Based on the studies on those search algorithms, we also introduced a new hybrid searching algorithm, AHS (Automatic Hybrid Search Algorithm).

Instead of stopping on finding only one result, the new algorithm is able to find all the feature subsets with the same highest merit measured in consistency rate. C4.5 is integrated into the algorithm in step two to evaluate and select again in order to further ensure the result to be more accurate with a slight trade off with time. Finally, we developed a tool to run ES, PS, HS and AHS and an empirical evaluation of these measures is given. Their pros and cons on accuracy, time consumption and the reduction of the number of features were discussed.

CHAPTER 8: FUTURE DIRECTION

Future work will focus on experimental analysis on more datasets with broad feature spaces. The AHS algorithm will be tested on more datasets with various backgrounds. The comparison between different algorithms in performance and accuracy can be conducted more thoroughly and the outcomes will be more convincing as more experimental results will be available. The different evaluation criteria would be employed to test the AHS and make a comparison with consistency rate. We can also change the different error checking algorithms for the second part of AHS algorithm.

In addition, different data mining functions could be added to our programs such as eliminating missing values or data discretization. Numerical dataset could be discretized into an appropriate form by the program before the feature selection procedure. Different algorithms for discretizing dataset would be implemented. It is interesting to explore the measures that can make several related data mining techniques working together and be able to handle all types of data.

REFERENCES

[1] Manoranjan Dash and Huan Liu. Consistency-based search in feature selection. Artificial Intelligence 151 (2003) 155-176.

[2] Huanjing Wang, AllenParrish, Randy K.Smith and Susan Vrbsky. Variable Selection and Ranking for Analyzing Automobile Traffic Accident Data . Proceedings of the 2005 ACM symposium on Applied computing. (2005)

[3] Huan Liu and Lei Yu. Toward integrating Feature Selection Algorithms for Classification and Clustering. Knowledge and Data Engineering, IEEE Transactions. Page(s): 491 – 502, (2005)

[4] R. Kohavi and G. John, "The wrapper approach", *Feature Selection for Knowledge Discovery and Data Mining*, pp. 33–50, Kluwer Academic Publishers, New York (1998).

[5] Lei Yu, Huan Liu. Efficient Feature Selection via Analysis of Relevance and Redundancy. Journal of Machine Learning Research 5 (2004) 1205–1224.

[6] A. L. Oliveira, A.S. Vincentelli, Constructive induction using a non-greedy strategy for feature selection, in: Proceedings of Ninth International Conference on Machine Learning, Aberdeen, Scotland, Morgan Kaufmann, San Mateo, CA, (1992),pp.355-360.

[7] D. S. Johnson, Approximation algorithms for combinatorial problems, J. Comput. System Sci. 9(1974)256-278.

[8] H. Liu, R. Setiono, Feature selection and classification – A probabilistic wrapper approach, in: Proceedings of Ninth International Conference on Industrial and Engineering Applications of AI and ES, (1996), pp.419-424.

[9] Jiawei Han, Micheline Kamber, Data Minging Concepts and Techniques, Elsevier Inc. (2000).

[10] Pengpeng lin, Huanjing Wang, A Novel Hybrid Search Algorithm for Feature Selection, SEKE.(2009).

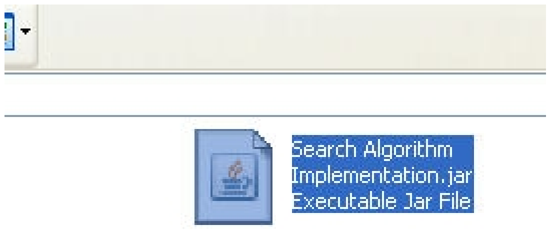
[11] J. R. Quinlan, C4.5: programs for machine learning, Los Altos, California: Morgan Kanfmann, (1993).

[12] UCI Repository of Machine Learning Databases,
http://archive.ics.uci.edu/ml/

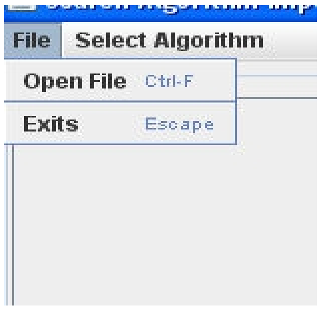
[13] http://www.cs.waikato.ac.nz/ml/weka/

TOOL SPECIFICATION

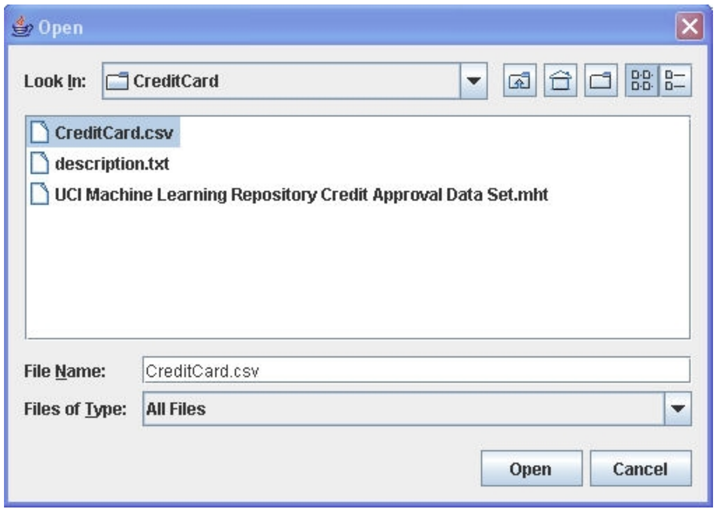
Step 1: Double click the program icon labeled with “Search Algorithm Implementation”:

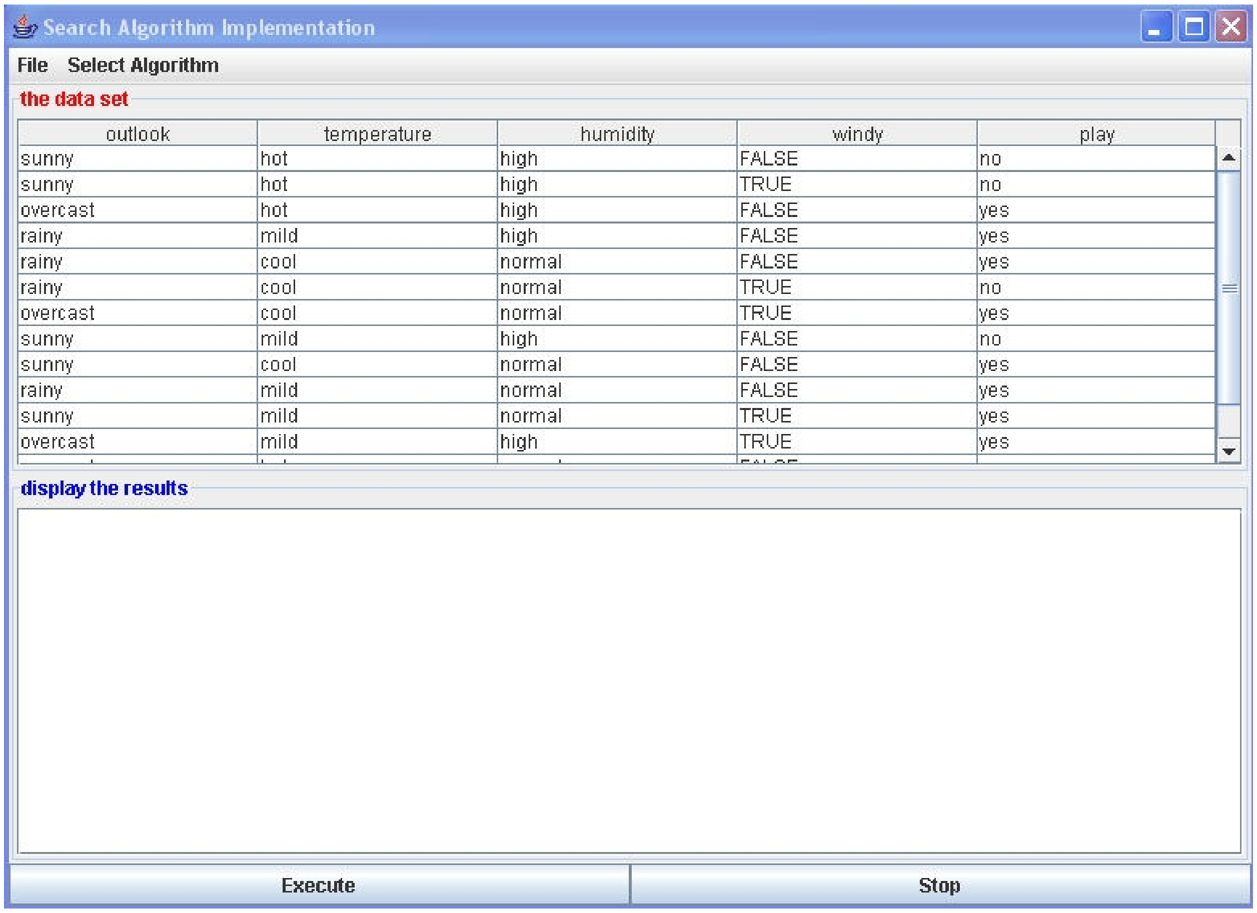


Step 2: Go to menu “file” at the leftmost corner and click “open file”:



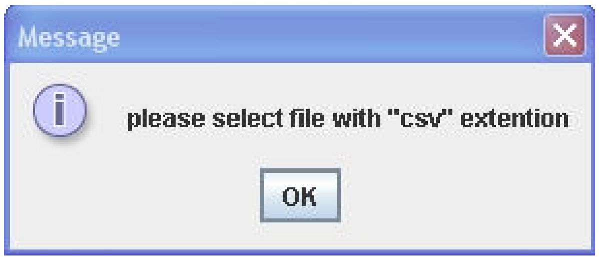
Step 3: Select data file with "CSV" extension:





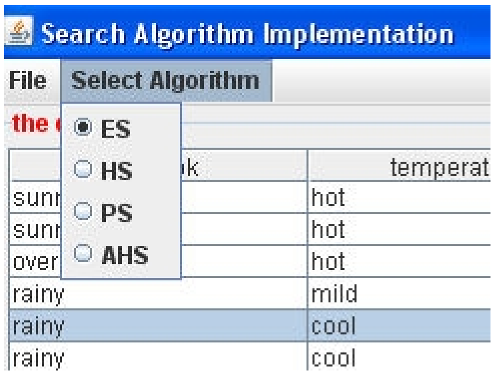
outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes

If you accidentally select file with different extension, the program will pop a window:



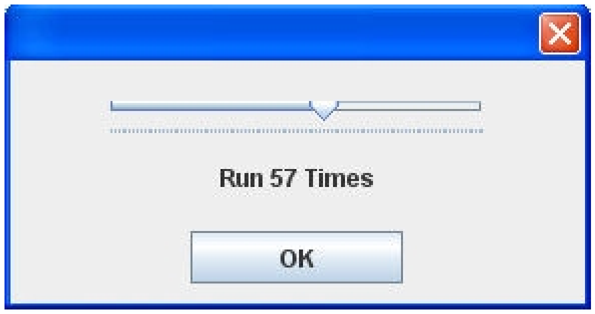
please select file with "csv" extention

Step 4: Choose an algorithm:

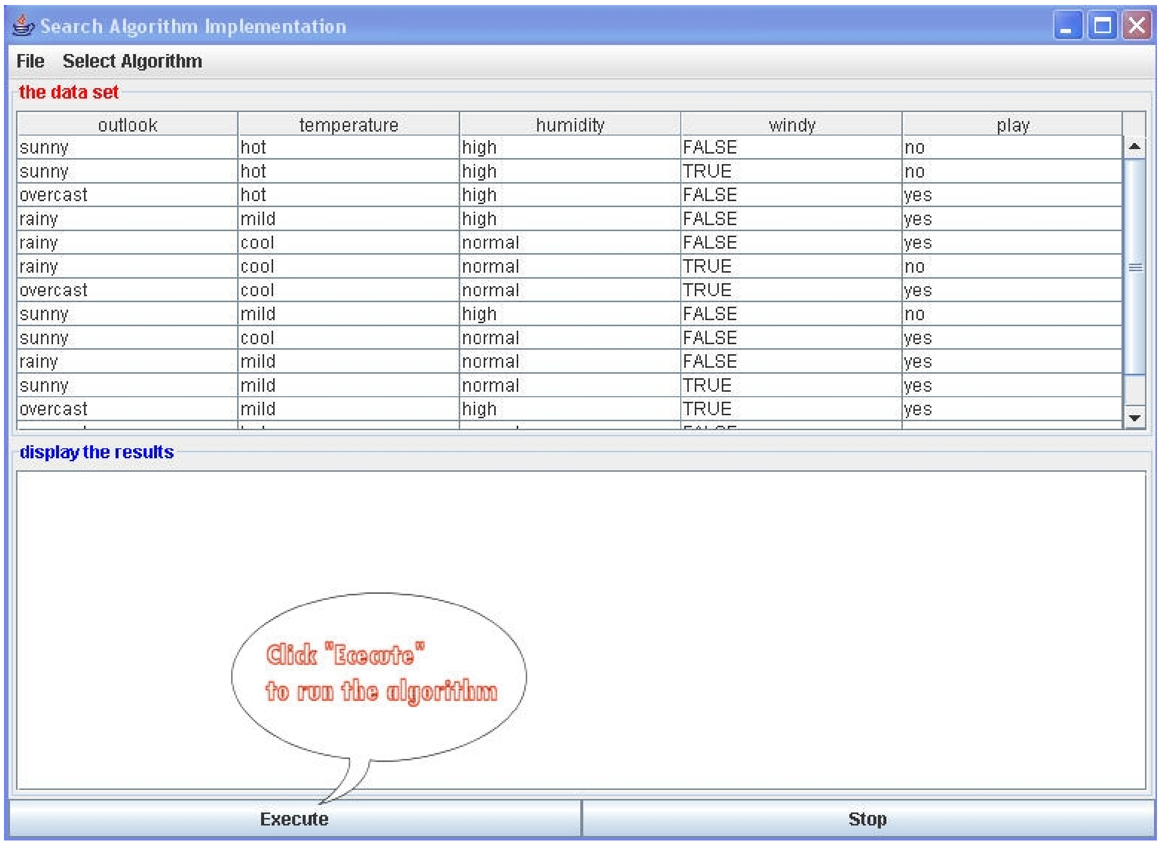


	temperat
sunr	hot
sunr	hot
over	hot
rainy	mild
rainy	cool
rainy	cool

If you select PS algorithm, you have to specify the running time:

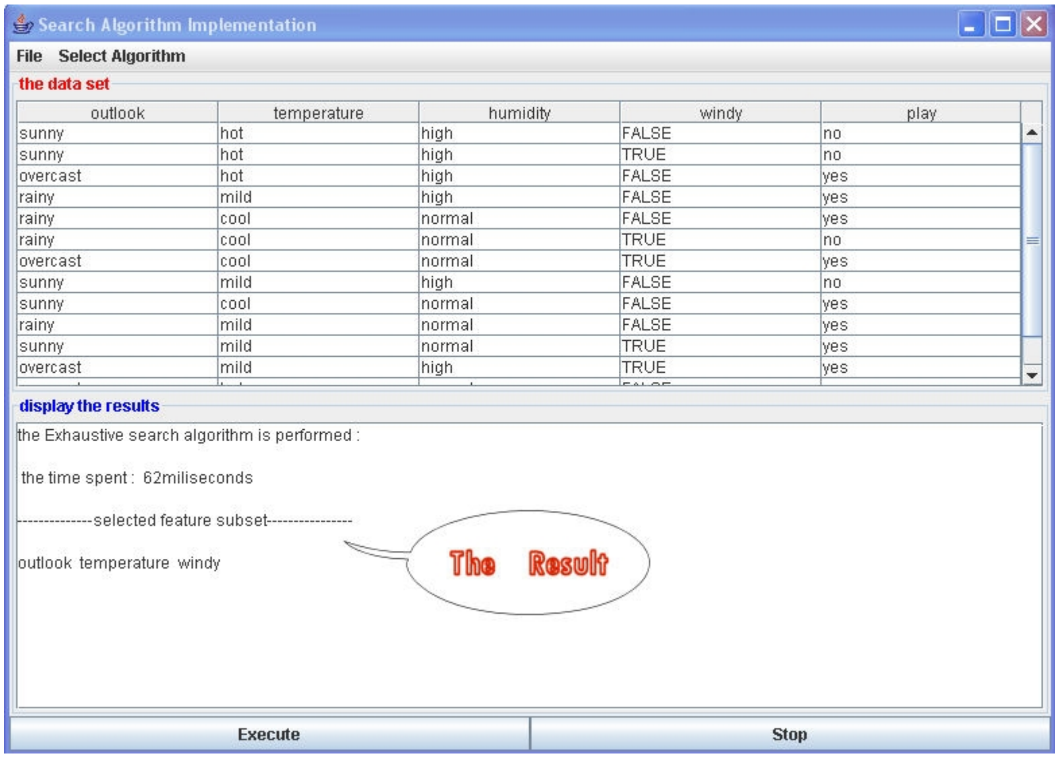


Step 5: Hit the button "Execute":


Search Algorithm Implementation
File
Select Algorithm
the data set
display the results
Click "Execute" to run the algorithm
Execute
Stop

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes

Output:



outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes

```
the Exhaustive search algorithm is performed :

 the time spent :  62miliseconds

--------------selected feature subset---------------

outlook  temperature  windy
```

APPENDIX

Source code of AHS

```
import javax.swing.*;

import java.lang.*;

import java.io.*;

import java.util.*;

public class LP1 {

   private readData toRead;

   private inConCal toInCon;

   private CombinationGenerator X;

   float BestResult;

   List<String[]> BestSet;

   List<String[]> LeftSet;

   private String S[];//feature set

   private List<String[]> TestSet;

   public LP1() {

      BestSet = new ArrayList<String[]>();

      TestSet = new ArrayList<String[]>();

      LeftSet = new ArrayList<String[]>();

      toRead = new readData();

      toInCon = new inConCal();

   }
```

```
  public List<String[]> result(File fileName)
     {
          if(S!=null&&TestSet!=null)
          toClear();
          toRead.toRead(fileName);
          S = new String[toRead.toGetTable().getColumnCount()];
          for(int i=0; i<toRead.toGetTable().getColumnCount();i++)
          S[i]=toRead.toGetTable().getColumnName(i);
          BestSet.add(S);
          BestResult = toInCon.ConCal(S,toRead.toGetTable());
           for(String aa:S)
           {
              String[]bb=new String[1];
              bb[0]=aa;
              TestSet.add(bb);
           }
           TestSet.remove(S.length-1);
           String Target = S[S.length-1];
           for(int i=0;i<TestSet.size();i++)
              LeftSet.add((String[])TestSet.get(i));
           float T = Integer.MIN_VALUE;
Stop:{
```

```
while(((String[])TestSet.get(0)).length<S.length)
{
   List<String[]> set = new ArrayList<String[]>();
   set.clear();
   for (Object aa : TestSet) {
      String[] input = addTarget((String[]) aa, Target);
      float SubResult = toInCon.ConCal(input, toRead.toGetTable());
      for (String a1 : input)
         System.out.print(a1+"  ");
      System.out.println();
      System.out.println(SubResult);
      if (T < SubResult) {
         T = SubResult;
         set.clear();
         set.add((String[]) aa);
      } else if (T == SubResult) {
         set.add((String[]) aa);
            }
      }
      if (T >= BestResult) {
      if (((String[]) set.get(0)).length <
         ((String[]) BestSet.get(0)).length) {
         BestSet = set;
```

```
            break Stop;
                }
                }
        else if(set.size()==TestSet.size())
        {
        int length1 = set.get(0).length;//to get current candidate's length
        TestSet = ToSieveDuplication(set);
        if(TestSet.size()<=length1)
                return BestSet;
        TestSet = CombineSet(TestSet,length1+1);
        }
        else{
        updateLeftSet(LeftSet,S);
        LeftSet = DeleteElement(LeftSet, set);
        if (LeftSet.isEmpty())
        {
                BestSet = set;
                return BestSet;
        }
        TestSet = AppendElement(LeftSet, set);
        }
    }
}
```

```
     return BestSet;
}
 private List<String[]> DeleteElement(List<String[]> LeftSet,List<String[]> Set)
 {
    List<String[]> returner = new ArrayList<String[]>();
    if(Set.size()==0)
        returner = LeftSet;
    else
    {
      List<String[]> newSet =(List<String[]>) ToSieveDuplication(Set);
     stop2:for (Object aa : LeftSet) {
          for (Object bb : newSet) {
              if (IsArrayEqual((String[]) aa, (String[]) bb))
                  continue stop2;
          }
          returner.add((String[]) aa);
      }
    }
    return returner;
 }
 private List<String[]> AppendElement(List<String[]> LeftSet,List<String[]> Set)
 {
    List<String[]> returner = new ArrayList<String[]>();
```

```
int Length=0;
for(Object aa:Set)
{
  for(Object bb:LeftSet)
  {
    String S1[] = (String[])aa;
    String S2[] = (String[])bb;
    String S3[] = new String[S1.length+S2.length];
    Length = S3.length;
    List<String> newList = new ArrayList<String>();
    for(int i=0;i<S1.length;i++)
    {
      newList.add(S1[i]);
    }
    for(int j=0;j<S2.length;j++)
    {
      newList.add(S2[j]);
    }
    for(int ij=0;ij<S3.length;ij++)
    {
      S3[ij]=newList.get(ij);
    }
    returner.add(S3);
```

```
          newList.clear();
        }
     }
     List<String[]> newSet =(List<String[]>) ToSieveDuplication(Set);
     if(Length!=0)
     {
        if(newSet.size()>=Length)
        {
        List<String[]> tempList = CombineSet(newSet,Length);
        for(Object ob:tempList)
        returner.add((String[])ob);
        }
     }
     return returner;
  }
   private List<String[]> CombineSet(List<String[]> set,int length)
   {
      List<String> temp = new ArrayList<String>();
      List<String[]> returner = new ArrayList<String[]>();
      int[] indices;
      CombinationGenerator X = new CombinationGenerator(set.size(),length);
      while(X.hasMore())
      {
        indices = X.getNext();
        for(int i=0;i<indices.length;i++)
        {
           String[]a1 = (String[])set.get(indices[i]);
           temp.add(a1[0]);
        }
        String[] tempString = new String[temp.size()];
        for(int i=0;i<tempString.length;i++)
        {
           tempString[i]=temp.get(i);
        }
        returner.add(tempString);
        temp.clear();
     }
     return returner;
  }
  private List<String[]> ToSieveDuplication(List<String[]> set)
  {
     List<String[]> SievedSet = new ArrayList<String[]>();

     Set<String> toSieve = new HashSet<String>();
     for(Object aa:set)
     {
            String[] temp = (String[])aa;
            for(String bb:temp)
                toSieve.add(bb);
        }
        for(Object aa : toSieve)
        {
            String[] temp2=new String[1];
            temp2[0]=(String)aa;
            SievedSet.add(temp2);
        }
        return SievedSet;
    }
    private String[] addTarget(String[] main, String target)
        {
            String[] newArray = new String[main.length+1];
            for(int i=0;i<main.length;i++)
            {
                newArray[i] = main[i];
            }
            newArray[newArray.length-1]=target;
            return newArray;
    }
    private boolean IsArrayEqual(String[] a,String[] b)
```

```
{
    boolean bb = true;
    for(int i=0; i<a.length;i++)
    {
       if(!(a[i].equalsIgnoreCase(b[i])))
       {
          bb=false;
          break;
       }
    }
    return bb;
}
  private void toClear()
  {
    BestResult=0;
    BestSet.clear();
    TestSet.clear();
    for(String aa : S)
       aa=null;
  }
  private void updateLeftSet(List<String[]> Set,String[] FeatureString){
       Set.clear();
       for(String aa:FeatureString)
```

```
        {
            String[]bb=new String[1];
            bb[0]=aa;
            Set.add(bb);
        }
            Set.remove(FeatureString.length-1);
    }
}
```