

2015


Feature Selection and Classification Methods for Decision Making: A Comparative Analysis

Osiris Villacampa

Nova Southeastern University, osiris@nova.edu

This document is a product of extensive research conducted at the Nova Southeastern University [College of Engineering and Computing](#). For more information on research and degree programs at the NSU College of Engineering and Computing, please click [here](#).

Follow this and additional works at: http://nsuworks.nova.edu/gscis_etd

 Part of the [Business Intelligence Commons](#), [Management Information Systems Commons](#), [Management Sciences and Quantitative Methods Commons](#), [Technology and Innovation Commons](#), and the [Theory and Algorithms Commons](#)

Share Feedback About This Item

NSUWorks Citation

Osiris Villacampa. 2015. *Feature Selection and Classification Methods for Decision Making: A Comparative Analysis*. Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. (63)
http://nsuworks.nova.edu/gscis_etd/63.

This Dissertation is brought to you by the College of Engineering and Computing at NSUWorks. It has been accepted for inclusion in CEC Theses and Dissertations by an authorized administrator of NSUWorks. For more information, please contact nsuworks@nova.edu.

Feature Selection and Classification Methods for Decision Making: A
Comparative Analysis

By
Osiris Villacampa

A dissertation submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy
In
Information Systems

College of Engineering and Computing
Nova Southeastern University

2015

We hereby certify that this dissertation, submitted by Osiris Villacampa, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Junping Sun, Ph.D.
Chairperson of Dissertation Committee

Date

Easwar Nyshadham, Ph.D.
Dissertation Committee Member

Date

Steven Zhou, Ph.D.
Dissertation Committee Member

Date

Approved:

Amon B. Seagull, Ph.D.
Interim Dean, College of Engineering and Computing

Date

College of Engineering and Computing
Nova Southeastern University

2015

An Abstract of a Dissertation Report Submitted to Nova Southeastern University in
Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Feature Selection and Classification Methods for Decision Making: A Comparative Analysis

by
Osiris Villacampa

August 2015

The use of data mining methods in corporate decision making has been increasing in the past decades. Its popularity can be attributed to better utilizing data mining algorithms, increased performance in computers, and results which can be measured and applied for decision making. The effective use of data mining methods to analyze various types of data has shown great advantages in various application domains. While some data sets need little preparation to be mined, whereas others, in particular high-dimensional data sets, need to be preprocessed in order to be mined due to the complexity and inefficiency in mining high dimensional data processing. Feature selection or attribute selection is one of the techniques used for dimensionality reduction. Previous research has shown that data mining results can be improved in terms of accuracy and efficacy by selecting the attributes with most significance. This study analyzes vehicle service and sales data from multiple car dealerships. The purpose of this study is to find a model that better classifies existing customers as new car buyers based on their vehicle service histories. Six different feature selection methods such as; Information Gain, Correlation Based Feature Selection, Relief-F, Wrapper, and Hybrid methods, were used to reduce the number of attributes in the data sets are compared. The data sets with the attributes selected were run through three popular classification algorithms, Decision Trees, k-Nearest Neighbor, and Support Vector Machines, and the results compared and analyzed. This study concludes with a comparative analysis of feature selection methods and their effects on different classification algorithms within the domain. As a base of comparison, the same procedures were run on a standard data set from the financial institution domain.

Dedication

This dissertation is gratefully dedicated to my late mother Sylvia Villacampa. Thanks for teaching me that all goals are attainable with hard work and dedication.

Acknowledgements

I would like to extend my sincere and heartfelt gratitude towards all the individuals who have supported me in this endeavor. Without their guidance, advice, and encouragement, I would have never finished.

I would like thank my advisor, Dr. Sun, for his guidance, direction, patience, and continuous encouragement throughout the time of my dissertation research. He was always there to answer my questions and set time to meet with me whenever needed. I will be forever grateful.

I would also like to thank my committee members, Dr. Nyshadham and Dr. Zhou, whose comments, advice, and suggestions from idea paper to final report were valued greatly.

I would also like to thank my sister, Dulce. She will always be my role model and muse. Thanks to my children, Alexis and Osiris, who were always supporting me and reminding me of the ultimate goal.

Finally, I would like to thank my wife, Maria. Her words of encouragement and support are what kept me going all these years.

Table of Contents

Abstract iii
List of Tables viii
List of Figures xi

Chapters

1. Introduction 1
Background 1
Statement of Problem and Goal 3
Relevance and Significance 5
Barriers and Issues 6
Definition of Terms 7
Organization of the Remainder of the Dissertation Report 7

2. Review of Literature 9
Feature Selection 12
Data Mining 18
Performance Measures 25

3. Methodology 34
Introduction 34
Data Mining Process 34
Data 37
Data Acquisition 37
Data Pre-Processing 40
Feature Selection 44
Classification Algorithms 52
Performance Evaluation 74
Apparatus 84
Data Sets 85
Summary 87

4. Results 89
Introduction 89
Bank Data Set 89
Service Data Set 109
Summary 124

5. Conclusions, Implications, Recommendations, and Summary 125

Introduction 125

Conclusions 126

Implications 128

Recommendations 128

Summary 129

Appendixes

A. Data Schema 132

B. Data Dictionaries 133

C. WEKA Functions and Parameters 149

D. All Classification Results 155

Reference List 161

Bibliography 170

List of Tables

Tables

1. Confusion Matrix 26
- 2a. Confusion Matrix for a Perfect Classifier 28
- 2b. Confusion Matrix for Worst Classifier 28
- 2c. Confusion Matrix for an Ultra-Liberal Classifier 28
- 2d. Confusion Matrix for an Ultra-Conservative Classifier 29
3. Quinlan (1986) Golf Data Set 47
4. Top 10 Attributes Ranked by Relief-F Using the UCI Bank Data Set 49
5. Performance Matrix Template 51
6. Web Data Set 55
7. Web Data Set with Distance Calculated 56
8. Data after Normalization 57
9. Data Set with Weighted Distance 58
10. Data after Discretization 61
11. Confusion Matrix for a Binary Classification Model 74
12. Bank Data Set Attributes 86
13. Top 15 Attributes Ranked by Information Gain by Using the UCI Bank Data Set 90
14. Top 15 Attributes Ranked by Relief-F by Using the UCI Bank Data Set 91
15. Attributes Selected by CFS by Using the UCI Bank Data Set 92
16. Results after Applying the Wrapper Selection by Using the UCI Bank Data Set 93
17. Performance of the Decision Tree Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics 94

18. Confusion Matrix for Decision Tree Algorithm Using J48 Wrapper Data Set 96
19. Performance Measures for Decision Tree Algorithm Using J48 Wrapper Data Set 96
20. Performance of the J48 Classifier across the UCI Bank Data Set in Terms of Accuracy AUC, TP Rate, and FP Rate Evaluation Statistics for Different Confidence Factors 97
21. Performance of the Decision Tree Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics Using a Parameter Search 98
22. Confusion Matrix for Decision Tree with Optimized Parameter Wrapper Data Set 99
23. Performance Measures for Decision Tree with Optimized Parameter Wrapper Data Set 99
24. Performance of the K-NN Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics 100
25. Confusion Matrix for Nearest Neighbor Using $k = 5$ 101
26. Performance Measures for Nearest Neighbor Using $k = 5$ 102
27. Performance of the Lib-SVM Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, True Positive Rate, and True Negative Rate Evaluation Statistics 103
28. Average Overall Classification Accuracy on Bank Data Set Based on Individual Runs 105
29. Accuracy Results Using Optimized Parameters by Using Experimenter 106
30. AUC Results Using Optimized Parameters by Using Experimenter 106
31. F-Measures Results for Bank Data Set Using Optimized Parameters by Using Experimenter 107
32. Attributes Ranked Using Information Gain Method across the Service Data Set 109
33. Attributes Ranked by the Relief-F Method across the Service Data Set 110
34. Attributes Selected by CFS Method by Using the Service Data Set 111

35. Features Selected by Wrapper Selection by Using Service Data Set 112
36. Performance of the Decision Tree Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics 113
37. Decision Tree Confusion Matrix 114
38. Decision Tree Performance Measures 115
39. Performance of the K-NN Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics 116
40. Confusion Matrix for Nearest Neighbor Using $k = 1$ and Wrapper Data Set 117
41. Performance Measures for Nearest Neighbor Using $k = 1$ and Wrapper Data Set 118
42. Performance of the LIB-SVM Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics with Default Parameters 118
43. Performance of the Lib-SVM Classifier across the Service Data Set in Terms of Accuracy AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics Grid Search Parameters 119
44. Average Overall Classification Accuracy on Service Data Set across All Classifiers and Feature Selection Methods through Individual Testing 120
45. Average Overall Classification Accuracy by Using Service Data Set across All Classifiers and Feature Selection Methods Using Optimized Parameters 121
46. Average Overall AUC on Service Data Set across All Classifiers and Feature Selection Methods Using Optimized Parameters 122
47. F-Measures Results for Service Data Set with Optimized Parameters Using Experimenter 123

List of Figures

Figures

1. KDD Process 11
2. Feature Selection Process 13
3. K-Nearest Neighbor with $k = 3$ 19
- 4a. Linear Support Vector Machine 24
- 4b. Radial Support Vector Machine 24
5. Receiver Operating Characteristic Curve Points 30
6. Receiver Operating Characteristic Curves 31
7. Area Under Receiver Operating Characteristic Curve (AUC) 32
8. Framework Used in this Research 36
9. Data Acquisition Flow 39
10. K-NN Visualization with $k = 3$ 53
11. Final Decision Tree 62
12. Subtree Raising - Subtree D is Replaced by a Single Node 64
13. Support Vector Machine for a Binary Class 67
- 14a. SVM Optimized by Grid Search of Parameters 70
- 14b. Grid Search with a Decrease in C Value 71
15. 5-Fold Cross-Validation 73
16. Classification Flow with WEKA's KnowledgeFlow 78
17. WEKA Multiple ROC Workflow 80
18. Multiple Receiver Operating Characteristic Curves for IBK – KNN, J48 – C4.5 Decision Tree, and LibSVM – SVM Classifiers 82
19. Area Under Curve 84

20. Performance of J48 Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 95
21. Decision Tree Pruning. Lower Confidence Factor Indicates Higher Pruning 98
22. Performance of K-NN Classifier across Different Feature Sets Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 101
23. Performance of SVM Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 104
24. ROC Curve for Bank Data Set 107
25. Classification Performance on Bank Data across Data Sets 108
26. Performance of J48 Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 114
27. Performance of K-NN Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 117
28. Performance of SVM Classifier across Different Feature Sets Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics 120
29. Service Data ROC Curves 122
30. Service Data Set Classification Performance Measures across Data Sets 123

Chapter 1

Introduction

Background

Businesses are constantly looking for more methodologies to keep them competitive in today's marketplace. The low cost of disk space and ease of data capture (i.e., barcodes, Radio Frequency Identification - RFID tags, and credit card swipes) have led for the storage of enormous amounts of data. The data comes from various systems in the enterprise such as, Point of Sale (POS) systems, web sites, Customer Relationship Management (CRM) software, and more. In past decades this type of information has been stored in data-warehouses and mostly used to produce trending and historical reports using tools such as Online Analytical Processing (OLAP) and Structured Query Language (SQL) (Watson & Wixom, 2009).

Today, as computing power increases and becomes more affordable, a new trend playing an important role is to mine the data for unknown patterns and to extract data that is previously unknown that may be useful to improve the decision making process (Fayyad, Piatetsky-Shapiro, & Smyth, 1996).

Chen, Han, and Yu (1996) state: Data mining, which is also referred to as knowledge discovery in databases, means a process of nontrivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from databases. (p.866) with a goal of making it ultimately understandable.

This discovery of knowledge has been used by financial institutions, to detect fraud and Index prices (Major & Riedinger, 1992); in medical research, such as heart disease prediction (Palaniappan & Awang, 2008); and in marketing, to create tools such as market-basket analysis (Agrawal, Mannila, Srikant, Toivonen & Verkamo, 1996). Data mining, which is mistakenly used as a synonym to Knowledge Discovery in Database (KDD), is just one of the steps in the knowledge discovery process (Fayyad, 1996). In general, data mining methods can be classified as two categories: supervised and unsupervised learning methods (Han, Kamber, & Pei, 2011). In supervised learning, the algorithm uses a training data set to learn model parameters. Classification algorithms, such as Decision Trees (Breiman, Friedman, Stone, & Olshen, 1984), Support Vector Machine (SVM) (Vapnik, 1995), and Nearest Neighbor (Cover & Hart, 1967) are all members of this group. Unsupervised learning, on the other hand, uses the data itself to build the model. Clustering algorithms are the best known of this group. A third type, semi-supervised, has also been introduced as a hybrid option. Matching the data set being studied with the appropriate data mining algorithm is one of the key factors for a successful outcome.

As more information is collected from different sources, the likelihood of needing to work with high dimensional data sources increases. High dimensional tables or those containing more than 10^2 to 10^3 attributes are starting to be the norm (Fayyad, 1996). Some disciplines such as genomic technology have sources that contain thousands if not tens of thousands of attributes (Dougherty, Hua, & Sima, 2009).

Dealing with high dimensional databases has been a key research area in statistics, pattern recognition, and machine learning (Blum & Langley, 1997). Researchers are just now applying the same interest to commercial data sets.

Statement of Problem and Goal

While disciplines such as bioinformatics and pattern recognition have been using data mining for years, more research needs to be done on high dimensional business data. The primary goal of this study is to use a real-world example, records from auto dealerships service departments, for this research. The main objective is to identify potential buyers of new vehicles based on the service history of their current vehicles. While most of the data in other domains come from a single source, the data used in this research came from many different systems. The data to be used in this study was collected from service records of approximately 200 automobile dealerships. This kind of data was combined with customer specific data retrieved from the Customer Relationship Management (CRM) systems of these same dealerships. The end result is a highly dimensional data set that contains thousands of records. There are several problems when data mining in any of high dimensional data sets.

1. As the number of features (dimensions) increases, the computational cost of running the induction task grows exponentially (Kuo & Sloan, 2005). This *curse of dimensionality*, as reported by Powell (2007) and Guyon & Elisseeff (2003), affects supervised as well as unsupervised learning algorithms.
2. The attributes within the data set may also be irrelevant to the task being studied, thus affecting the reliability of the outcomes.

3. There may be correlation between attributes in the data set that may affect the performance of the classification.

Feature selection or attribute selection is a technique used to reduce the number of attributes in a high dimensional data set. By reducing the number of variables in the data set the data mining algorithm's accuracy, efficiency, and scalability can be improved (Guyon & Elisseeff, 2003). The two main approaches to feature selection are the filtering and wrapper methods. In the filtering method the attributes are selected independently to the data mining algorithms used. Attributes deemed irrelevantly will be filtered out (John, Kohavi, & Pfleger, 1994). The wrapper method selects attributes by using the data mining algorithm selected as a function in the evaluation process (John, Kohavi, & Pfleger, 1994).

One of the successful factors in data mining projects depends on selecting the right algorithm for the question on hand. One of the more popular data mining functions is classification (Wu, et al., 2008). For this study we have opted to use several classification algorithms, as our goal is to classify our data into two labels, referred to as binary classification. There are different types of classification algorithms available. For the purposes of this study we have chosen C4.5, a Decision Tree algorithm, K-Nearest Neighbor (K-NN), and Support Vector Machine (SVM) algorithms.

The goal in this research is made up of five related sub-goals as follows:

- 1) Compare and contrast different feature selection methods against the mentioned high dimensional data set and a reference data set. Both filter and wrapper methods were applied to these data sets, and their results are compared and

analyzed. The classification accuracy achieved by each method is compared against the better feature selection method found.

- 2) Repeat the above procedure by using different classification methods, including C4.5, a Decision Tree algorithm, K-Nearest Neighbor (K-NN), and Support Vector Machine (SVM) algorithms.
- 3) Compare the accuracy of the classification algorithms by using the best attributes selected for each algorithm. All methods were tested and validated on a binary classification task.
- 4) Use different thresholds in the classification systems and compare the effects on accuracy. K values in K-NN, number of nodes in Decision Trees, and Cost of Error (C) and Gamma settings (γ) in the SVM algorithm.
- 5) Determine which classification algorithm and feature selection combination produces the better results in order to determine new potential car buyers.

The classification algorithms, Decision Tree, K-Nearest Neighbor, and Support Vector Machine, are selected from the top 10 most commonly used algorithms (Wu, et al., 2008).

Relevance and Significance

The application of data mining for decision making is relatively new in some real world environments. The purpose of this research was to run a comparative analysis on a real world data set not only on the feature selection methods but on different classification algorithms as well. In addition, different performance measures were

compared to illustrate the difference in using the previously mentioned methods and algorithms.

Barriers and Issues

As in many other data mining exercises we were confronted with several obstacles. The automotive data set consists of thousands of attributes and hundreds of thousands of records. Preliminary queries ran against our data tables showed that approximately 20% of the records contained null values on critical features. The quality of the data in our data set must be improved by cleaning the noise and dealing with null values. The original data set is also composed of disparate sources. This heterogeneity was dealt in the preprocessing stage of our study. Another challenge presented in this study is the highly unbalanced dataset. This imbalance, 90% in one class vs. 10% in the other, is a result of having the majority of records in a class other than the one of interest. The data set size restriction imposed on us by the software used in this research is limited by the amount of memory available in the system. We have tried to lift this restriction by populating the test computer with 32GB of random access memory. In addition, the data set was reduced initially by random selection due to its size (thousands of records). Finally, over-fitting the data to a particular model is another obstacle that needed to be addressed. This was accomplished by implementing feature selection methods, reasonable values for k when using the K-NN algorithm, and post pruning our decision trees.

Due to time constraints and the number of different permutations possible in our study, we restricted ourselves to comparing 6 feature selection methods (3 filter, 2 wrapper, and 1 hybrid) on 3 of the most popular classifier algorithms, Decision Trees

(C4.5), Support Vector Machines (SVM), and K-Nearest Neighbor (K-NN). The following are the definition of terms used in this dissertation:

Definition of Terms

AUC – Area under the receiver-operating-characteristic curve

C45 - Decision Tree Algorithm

CRISP-DM - Cross-Industry Standard Process-Data Mining

CRM – Customer Relationship Management

F-Measure – Metric used for classification accuracy

KDD – Knowledge Discovery in Database

K-NN - K Nearest Neighbor

ROC – Receiver Operating Characteristic

SVM - Support Vector Machine

WEKA – Waikato Environment for Knowledge Analysis

Curse of Dimensionality – A term used to describe the difficulty and increase in cost of computing as the number of features increases.

Organization of the Remainder of the Dissertation Report

Chapter 2 includes a review of literature related to the use of feature selection to increase the effectiveness of data mining models. Different methods were compared and contrasted as to their strengths and weaknesses. In addition, the data mining algorithms, classification in particular, are reviewed, discussed and compared. Current, and past research were also evaluated. It concludes with an analysis of the selected feature selection methods and classification algorithms selected for this research.

Chapter 3 proposes the methodology to be used in this research. The CRISP-DM process for knowledge discovery is discussed. Steps in the process such as preparation and cleaning of the source data will be described in detail. The pre-processing stage and transformation stage which includes feature selection are also detailed. The feature selection methods proposed are discussed and metrics used for comparison are explained. The classification algorithms selected for this study are detailed along with the tests used to analyze their effectiveness.

Chapter 4 presents and describes the results of the study. It begins with the results of applying the feature selection methods in our data sets. Once the feature sets have been reduced, a new data set is saved for each method to be used in the classification phase. Our selected classification algorithms are applied to each data set and the results are compared using different performance measures.

Chapter 5 reviews our research questions, discusses our conclusions of the research based on the results, and provides suggestions for future research.

The data schema, data dictionaries, parameters used in our workbench software, and all the classification results are presented in the Appendices.

Chapter 2

Review of Literature

In today's competitive market, companies must make critical decisions that will affect their future. These decisions are based on current and historical data the enterprise has collected using Customer Relationship Management (CRM), Enterprise Resource Management (ERP), websites, and legacy applications. As the dimensionality and size of the data warehouses grows exponentially, domain experts must use tools to help them analyze and make decisions in a timely manner.

Knowledge Discovery in Data (KDD)

The field of Knowledge Discovery in Databases (KDD) has grown in the past several decades as more industries find a need to find valuable information in their databases. The KDD process (Fayyad, Piatetsky-Shapiro, & Smyth, 1996) is broken down into five phases (Figure 1);

1. Selection – The first stage consists of collecting data from existing sources to be used in the discovery process. The data may come from single or multiple sources. This may be the most important stage since the data mining algorithms will learn and discover from this data.
2. Preprocessing - The main goal of this stage is to make the data more reliable. Methods used to account for missing data are analyzed and implemented. Dealing with noisy data or outliers is also part of this stage.

3. Transformation – Now that we have reliable data we can make it more efficient. The uses of feature selection methods to reduce dimensionality and feature extraction to combine features into new ones are implemented at this point. Discretization of numerical attributes and sampling of data are also common tasks performed in this stage.
4. Data Mining – Before the data is mined, an appropriate data mining task such as classification, clustering, or regression needs to be chosen. Next, one or several algorithms specific to the task, such as decision trees for classification, must be properly configured and used in the discovery of knowledge. This process is repeated until satisfying results are obtained.
5. Evaluation – The last step is the interpretation of results in respect to pre-defined goals. A determination is made if the appropriate data mining model was chosen. All steps of the process are reviewed and analyzed in terms of the final results.

This study concentrates in two critical areas of the KDD process; transformation by reducing the feature set and the data mining process.

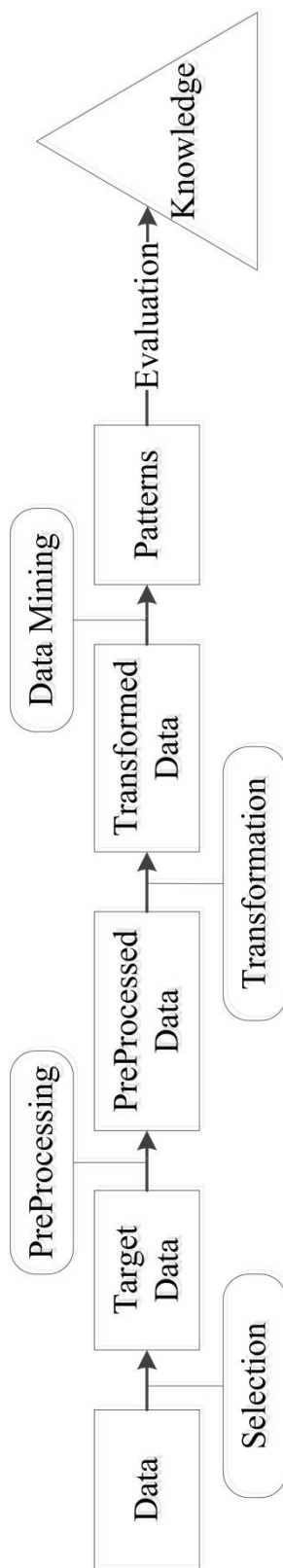


Figure 1. KDD Process (Fayyad et al., 1996)

Feature Selection

As the dimensionality of data increases so does the likelihood of having attributes which are irrelevant, redundant, and noisy (Chang, Verhaegen, & Duflou, 2014). A common method of reducing the dimensionality of the data to be analyzed is to reduce the number of features or variables to a more manageable number while not reducing the effectiveness of the study.

Feature selection or variable selection consists of reducing the available features to a set that is optimal or sub-optimal and capable of producing results which are equal or better to that of the original set. Reducing the feature set scales down the dimensionality of the data which in turn reduces the training time of the induction algorithm selected and computational cost, improves the accuracy of the final result, and makes the data mining results easier to understand and more applicable (Guyon & Elisseeff, 2003; Kohavi & John, 1997). While reducing the feature set may improve the performance of most classification algorithms, especially for K-NN algorithm, it may also lower the accuracy of decision trees (Li, Zhang, & Ogihara, 2004). Since decision trees have the capability of reducing the original feature set in the tree building process, beginning the process with fewer features may affect final performance.

Dash and Liu (1997) broke down the feature selection process into four steps; generation, evaluation, stopping criterion, and validation (Figure 2).

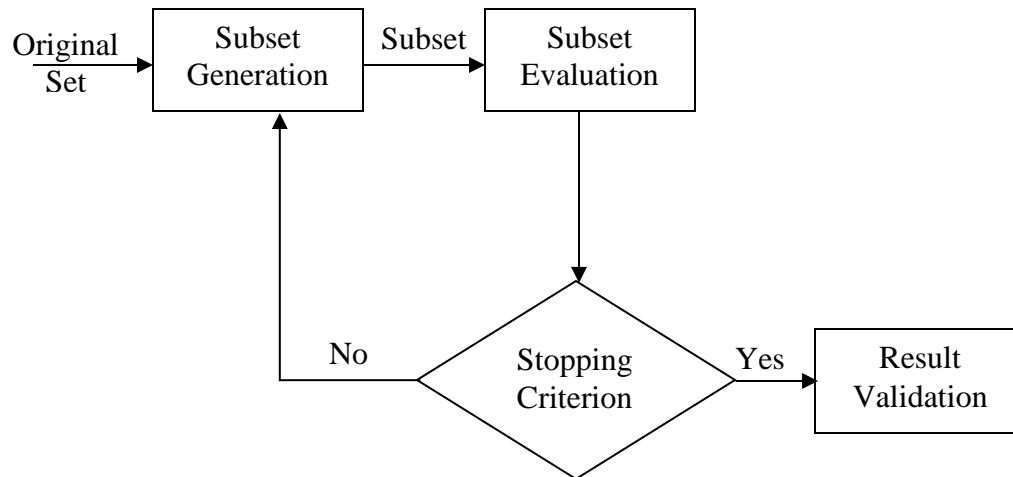


Figure 2. Feature Selection Process (Liu & Yu, 2005)

1. The first step, generation, involves searching the space of features for the subset that is most likely to predict the class best. Since the total number of possible subsets is 2^n , where n is the number of features, using all attributes becomes costly as the dimensionality of the data increases. In order to minimize cost search, algorithms have been developed that scan through the attributes in search of an optimal subset. Two common methods of traversing the space are Sequential Forward Selection and Backward Elimination. The Sequential Forward Selection begins with an empty set and adds attributes one at a time. Backward Elimination, on the other hand, begins with the entire set of attributes and starts eliminating until a stopping criterion has been met. Other variations such as a random method may be used which adds or deletes variables in its search for an optimal set (Devijer & Kittler, 1982). Other algorithms, such as; the Beam Search (BS) and

Smart Beam Search (SBS) algorithms select the best k features (beam-width), then proceed to add and test additional features to each of the selected k features until a stopping criterion is met (Ladha & Deepa, 2011). In their studies, Hall and Smith (1998) determined that backward and forward elimination search methods, although elementary, were proved to be as effective as more sophisticated ones such as Best First and Beam search algorithms (Rich & Knight, 1991).

2. The second step in the process uses a predetermined evaluation function that measures the goodness of the subset (Liu & Yu, 2005). This measurement is then used to determine the ranking of the evaluated sets, which in turn are used in the selection process. Among these functions are Information Gain, Correlation Analysis, Gini Index, and in the case of wrapper methods the induction algorithm itself.
3. The third step in the process is the stopping criterion. There are many ways in which the feature search may stop. The process may be stopped if the new feature set does not improve the classification accuracy. Other options are running a predetermined number of iterations, reaching a previously defined number of features, or selecting the top n features with the highest ranking.
4. The final step is the validation of the results against the induction algorithm selected. While not exactly being a part of the actual selection process, the authors include it as it will always follow the selection process (Liu & Yu, 2005).

Feature selection methods fall into three groups, Filter, Wrapper, and Hybrid. We will discuss each in the following sections.

Filter

The filter method of feature selection reduces the number of features using properties of the data itself independently to what learning algorithm is eventually used (John, Kohavi, and Pfleger, 1994). One advantage of applying a filter algorithm to a feature set is that the number of features used in the final induction algorithm will be reduced. Therefore not only the performance of classification algorithms will be improved, but also amount of the computer processing time will be reduced. Unlike wrapper methods, filter methods do not incorporate the final learning algorithm in their process. This independency has been reported as another benefit of using filter methods (Ladha & Deepa, 2011). Another benefit is that the same features may be used in different learning algorithms for comparative analysis. Hall and Smith (1998) reported that some filter algorithms such as Correlation-based Feature Selection (CFS) might produce results similar to or better than wrapper models on several domains. Yu and Liu (2003) also proposed a new correlation based feature selection method. In their study they showed the efficiency and effectiveness of such methods when dealing with highly dimensional data sets. However, Saeys et al. (2007) noted that filter based selection methods have the disadvantage of not interacting with the classifier algorithm eventually used. Another disadvantage reported was that most filter methods are univariate in nature, meaning that they don't take into consideration the values of other attributes. Their study was conducted on a highly dimensional bioinformatics data set (Saeys et al., 2007).

Hall and Holmes (2003) benchmarked the filtered based feature selection and one wrapper based method against 15 test data sets in their experiments. Their conclusion was

that filter based methods varied depending on the data set, but generally they were faster and improved the effectiveness of the classifying algorithms.

This study evaluated three different filter algorithms: two multivariate algorithms Relief-F and Correlation Based Feature Selection (CFS), and, *information gain* a univariate algorithm. Each method is described in the following paragraphs.

The principal behind the Relief-F algorithm (Kononenko, 1994) is to select features at random and then, based on nearest neighbors, give more weight to features that discriminate more between classes. These features are in turn ranked based on their relevance. In their empirical study, Wang and Makedon (2004) concluded that the Relief-F algorithm produced similar results to that of other filter algorithms, such as Information Gain and Gain Ratio, when the Relief-F algorithm is used in their particular domain, gene expression data.

Correlation-based Feature Selection (CFS) algorithms looks for features that are highly correlated with the class which has no or minimal correlation with each other (Hall, 2000).

Our last feature selection algorithm is *information gain* (IG). IG is a method that ranks features based on a relevancy score which is based on each individual attribute. The fact that the correlation between attributes is ignored makes it a univariate method.

Comparative studies between CFS and Gain Ratio methods have been performed in the past on different data domains. Karegowda, Manjunath, and Jayaram (2010) found that using the CFS method produced better results than Gain Ratio but at a substantial cost on computer time.

Wrapper

Unlike the filter method, wrapper algorithms use a preselected induction algorithm as part of the feature selection process. As features are added or subtracted the final results are ranked as to effectiveness of the selection. Since the induction algorithm itself is used in the evaluation phase of the selection process wrapper methods tend to score better results than filter methods. Kohavi and John (1997) compared the wrappers for feature subset selection against filter methods. They concluded that relevancy of attributes contribute greatly to the performance of the learning algorithms when the algorithm is taken into consideration. However, there are some limitations to these methods. The computational cost of running the evaluation is far greater than that of filter methods and increases as the number of attributes increases. Another disadvantage of the wrapper method is the likelihood of over-fitting the data.

There are also other wrapper methods. Instead of using single method wrapper such as sequential forward selection, Gheyas and Smith (2010) proposed a new method, *simulated annealing generic algorithm* (SAGA), which incorporates existing wrapper methods into a single solution. The research showed that combining methods reduced the weaknesses that were inherent to each individually.

Maldonado and Weber (2009) proposed a wrapper method based on the Support Vector Machine (SVM) classification. Their study concluded that using such method would avoid over fitting the data due to its capability of splitting the data. It also allowed the use of different Kernel functions to provide better results. One drawback noted was that their proposed algorithm used the backward elimination feature which was computationally expensive when working with highly dimensional data sets.

Hybrid/Two Stage Design

A hybrid method that incorporates the above methods has also been proposed (Kudo & Sklansky, 1998; Bermejo, de la Ossa, Gamez, & Puerta, 2012). This method uses a filter method in the first pass to remove irrelative features and then a classifier specific wrapper method to further reduce the feature set. By reducing the feature set from n features to a lower number k , the computation space in terms of the number of features is reduced from 2^n to 2^k . This hybrid filter-wrapper method would retain the benefits of the wrapper model while decreasing the computational costs that would be required by using a wrapper method alone.

Data Mining

The data mining phase of the KDD process is where the discovery of patterns in the data occurs. This discovery is performed by machine learning algorithms. This study will concentrate on the classification family of learning algorithms.

Classification Algorithms

One leg of this research is in classification algorithms. The goal of classification algorithms is to learn how to assign class labels to the unseen data based on models built from training data. When only two class labels exist, the classification is said to be binary. When more than two class labels exist, the problem becomes a multiclass classification. This study was focused on binary classification problems and in the comparison of different feature selection methods and their impact on three commonly studied classifier algorithms, K Nearest Neighbor, Decision Tree, and Support Vector Machine (SVM).

K-Nearest Neighbor K-NN

K-Nearest Neighbor (K-NN) classification is one of the simplest methods available for classifying objects (Cover & Hart, 1967). The algorithm assigns a class to the object based on its surrounding neighbor's class using a pre-determined distance function (Figure 3).

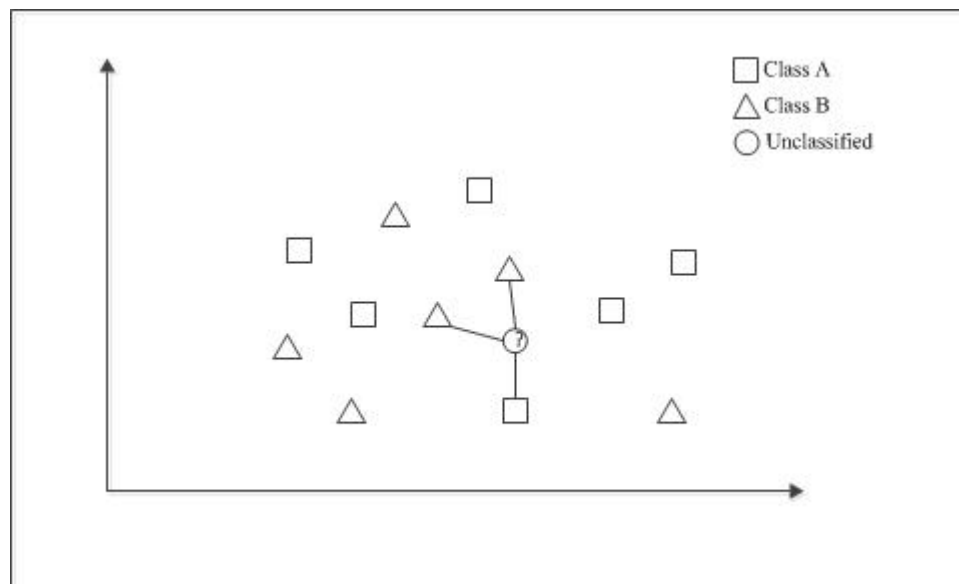


Figure 3. K-Nearest Neighbor with $k = 3$

The number of neighbors selected, k , is a positive odd integer, usually small, to avoid potential ties. If the value of k is 1, then the object is classified in the same class as its closest neighbor. One of the advantages of this method is that no complex training is required, an approach known as “lazy learner” or instance based learner. Kordos, Blachnick, & Strzempa (2010) showed that a properly configured K-NN may be as highly effective, if not more, than other classification algorithms.

The results of the K-NN algorithm depend on what values are used in its computation. The value, k , is the number of neighbors that will decide the class of the element in question. To avoid potential over-fitting of the data, researches have commonly used small numbers. In their studies, Cover and Hart (1967), proved that good predictions could be attained by using a value of $k = 1$. However, Kordos et al. (2010) reported that researchers should not confine themselves to small k values but test values in the ranges of 10 to 20 as well. Their research showed that while using a value in a higher range may take more computation time, but it may produce better results. Hamerly and Speegle (2010) proposed an algorithm that would cycle through different k values in order to minimize computational time while finding an optimum k value for the data set.

The second factor that may affect the outcome is how the distance between the elements is calculated. By default most researchers' use the Euclidean distance, but other calculations such as Chebyshev and Manhattan distance have also been implemented (Cunningham & Delany, 2007). Finally, in order to improve the results even more, weighting the distance calculation based on feature ranking has been studied as well (Hassan, Hossain, Bailey, & Ramamohanarao, 2008; Han, Karypis, & Kumar et al., 2001).

The advantage of using K-NN over other classification algorithms is that it is intuitive and easy to setup. However, there are several disadvantages when using the K-NN algorithm.

1. The distance function must be carefully selected and fine-tuned to achieve better accuracy. Since the distance equation is computed for all features selected,

features with higher scale values would dominate. In order to account for this, normalization of the attributes is performed before the distance is measured.

2. Data with irrelevant or correlated features must be cleaned beforehand as to not skew the results of the process (Bhatia & Vendana, 2010).
3. Computation cost is greater than other algorithms, since the process is computed in memory the amount of memory required is high. As high speed computer and memory become more affordable, this final disadvantage is becoming less concerned.

Decision Trees

Decision Trees is one of the most commonly used methods in classification (Ngai, Xiu, & Chau, 2009). Like all classification algorithms, the methods objective is to classify a target variable based on existing attribute values. In the case of decision trees, the process is broken down into individual tests (if-then) which begin at the root node and traverse the tree, depending on the result of the test in that particular node. The tree begins at the root node. From the root node the tree branches or forks out to internal nodes. The decision to split is made by impurity measures (Quinlan, 1986). Two commonly used measures in tree construction are Information Gain and Gini Index (Chen, Wang, & Zhang, 2011). These nodes in turn will continue to split until a final node or leaf node is grown. The leaf node determines the final classification of the variable being tested. Since each node tests a specific attribute in the data set, the model is very easy to understand. Tests at each node can be done on discrete as well as continuous data types. By default the tree will try to cover all possible outcomes in its

structure. The disadvantages of this method are that the tree will over-fit the data into its solution. The complexity of the tree will make the domain expert hard to follow the flow of decision making in the tree.

There are several ways to prevent over-fitting:

1. The processing of nodes can be stopped when all records belong to the same class.
2. Stop processing nodes when a predetermined threshold has been met or when all records have similar attribute values.
3. If expanding current node does not improve the information gain, then a leaf node can be introduced.
4. Other methods, such as post pruning (Witten et al., 2011), may be employed. In this case the tree is fully grown and then pruned for unnecessary branches. In their studies, Tsang et al. (2011), reported that pruning the decision trees improved the final results of the classification significantly.

There are several benefits to use decision trees. The algorithms are fast at classifying records, and easy to understand. They can handle both continuous and discrete attributes. The important attributes are easily identified by the decision maker. However, there are some disadvantages as well. Variations in data may produce different looking trees (Rokach & Maimon, 2005; Otero, Freitas, & Johnson, 2012), which are not good at predicting continuous attributes, because irrelevant attributes and noisy data may affect the tree structure (Anyanwu & Shiva, 2009). In addition, if the data set has missing attribute values, then the results of which the impurity measures computed will be affected. To circumvent this problem different methods have been introduced, such as

mean substitution and case substitution (Brown & Kros, 2003) which deal with missing values in data sets. Using this method, missing values are replaced with the mean of the given attribute and the substitutions are treated as valid observations.

When the ID3 (Iterative Dichotomiser 3) decision tree inducer was first introduced by Quinlan (1986), it did not support continuous attributes. Only categorical values were supported. Later, Quinlan(1993) introduced C4.5 which handled continuous attributes. The obstacle was overcome by discretizing the continuous data in order to perform testing at each node. Other inducers such as CART (Classification and regression trees) (Breiman et al., 1984) and SLIQ (Supervised Learning in Ques) (Metha et al. , 1996) have been introduced as well.

Decision tree classification has been studied in the medical sciences (Anunciacao et al. 2010; Ge and Wong 2008; Chen et al. 2011), text classification (Irani et al. 2010), and spam detection (Bechetti et al. 2009).

SVM

Support Vector Machines (SVM) (Vapnik, 1995) has shown great promise in binary classification (Yang & Liu, 1999). The goal of the SVM algorithm is to map the training data into a multi-dimensional feature space and then find a hyper-plane in said space that maximizes the distances between the two categories (Figure 4a-b).

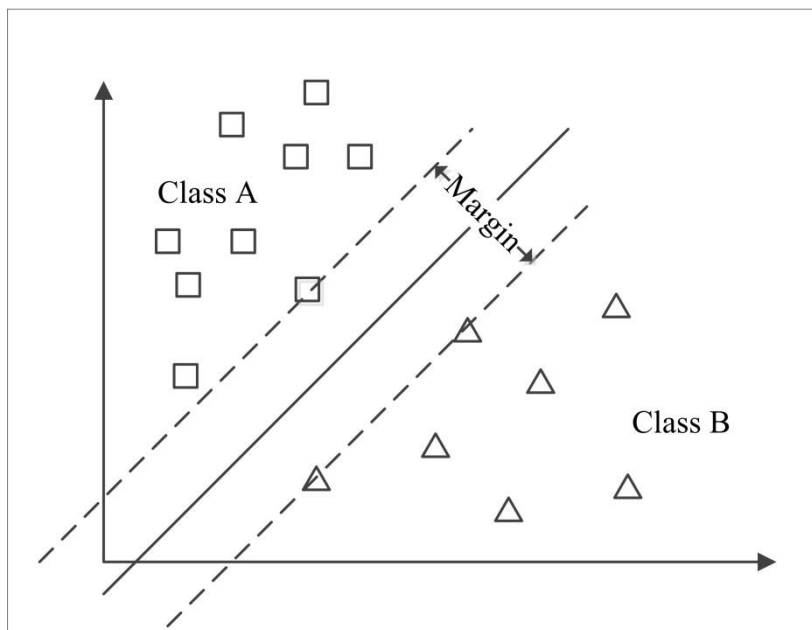


Figure 4a. Linear Support Vector Machine

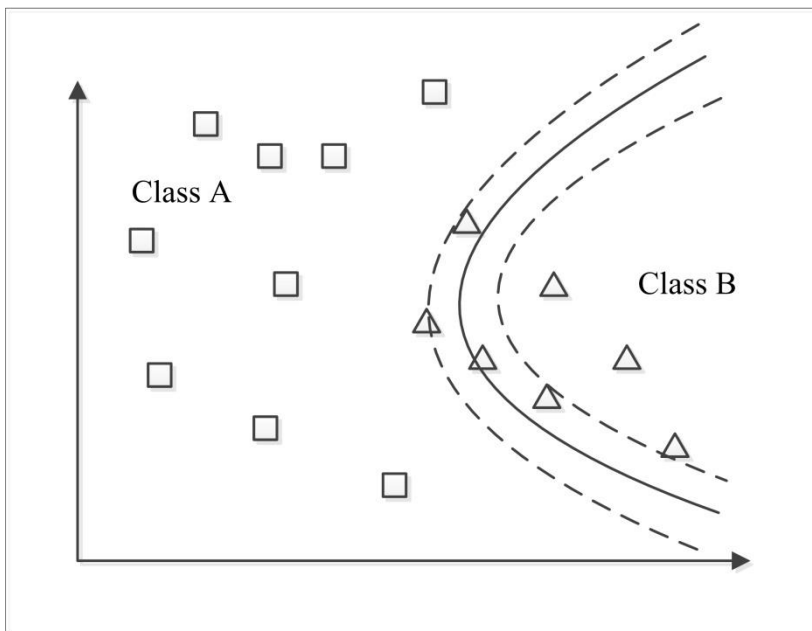


Figure 4b. Radial Support Vector Machine

Since the classifications may not be clearly separable by a linear plane non-linear kernels functions have been used (Figure 4b). Boser et al. (1992) reported that using non-linear functions proved to achieve higher performance and use less computing resources.

In addition, since features with different scale may affect the results of the SVM algorithm, normalization of the numeric data is performed. In addition, normalizing the data brings the numerical data within the same scale as categorical data, that is, to a (0, 1) scale.

Performance Measures

In order to determine the effectiveness of the classification algorithm used, a measurement is needed. Commonly used measurements include classification accuracy, F-Measure, precision, recall, Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) (Fawcett, 2006). These measurements can be calculated by the classification results commonly tabulated in a matrix format called a Confusion Matrix.

Confusion Matrix

In a classic binary classification problem, the classifier labels the items as either positive or negative. A confusion matrix summarizes the outcome of the algorithm in a matrix format (Chawla, 2005). In our binary example, the confusion matrix would have four outcomes:

True positives (TP) are positive items correctly classified as positive.

True negatives (TN) are negative items correctly identified as negatives.

False positives (FP) are negative items classified as positive.

False negatives (FN) are positives items classified as negative.

Table 1 illustrates a sample confusion matrix.

Table 1. Confusion Matrix

Confusion Matrix		Classified As:	
		Negative	Positive
Actual Class	Negative	TN	FP
	Positive	FN	TP

The following performance measures use the values of the confusion matrix in their calculation.

Classification Accuracy

The simplest performance measure is accuracy. The overall effectiveness of the algorithm is calculated by dividing the correct labeling against all classifications.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

The accuracy determined may not be an adequate performance measure when the number of negative cases is much greater than the number of positive cases (Kubat et al., 1998).

F-Measure

F-Measure (Lewis and Gale, 1994) is one of the popular metrics used as a performance measure. The measure itself is computed using two other performance measures, precision and recall.

$$precision = \frac{TP}{TP+FP}$$

$$recall = sensitivity = \frac{TP}{TP + FN}$$

Precision is the number of positive examples classified over all the examples classified.

Recall, also called the True Positive Rate (TPR), is the ratio of the number of positive

examples classified over all the positive examples. Based on these definitions F-measure is defined as follows:

$$f - measure = \frac{2 \times precision \times recall}{precision + recall}$$

In essence, the F-Measure is the harmonic mean of the recall and precision measures.

Using the confusion matrix and the performance measures mentioned above, Bramer (2007) noted four extreme cases a confusion matrix may detail:

- 1) A Perfect Classifier - A classifier that classifies all instances correctly. All positives are classified as positive and all negatives are classified as negative.
- 2) The Worst Classifier - A classifier that does not predict any positives or negatives correctly.
- 3) An Ultra-Liberal Classifier - A classifier that predicts all instances as positive.
- 4) An Ultra-Conservative Classifier - A classifier that predicts all instances as negative.

Tables 2a-2d show the confusion matrix for these cases along with the classification measures related to each matrix.

Table 2a. Confusion Matrix for a Perfect Classifier

Perfect Classifier				Total Instances
		Predicted		
		Positive	Negative	
Actual	Positive	P	0	P
	Negative	0	N	N
TP Rate (Recall) = $P / P = 1$ FP Rate = $0 / N = 0$ Precision = $P / P = 1$ F Measure = $2 \times 1 / (1 + 1) = 1$ Accuracy = $(P + N) / (P + N) = 1$				

Table 2b. Confusion Matrix for Worst Classifier

Worst Classifier				Total Instances
		Predicted		
		Positive	Negative	
Actual	Positive	0	P	P
	Negative	N	0	N
TP Rate (Recall) = $0 / P = 0$ FP Rate = $N / N = 1$ Precision = $0 / P = 0$ F Measure = Not Applicable (Precision + Recall = 0) Accuracy = $0 / (P + N) = 0$				

Table 2c. Confusion Matrix for an Ultra-Liberal Classifier

Ultra-Liberal Classifier				Total Instances
		Predicted		
		Positive	Negative	
Actual	Positive	P	0	P
	Negative	N	0	N
TP Rate (Recall) = $P / P = 1$ FP Rate = $N / N = 1$ Precision = $P / (P + N) = 1 / 2$ F Measure = $2 \times P / (2 \times P + N)$ Accuracy = $P / (P + N)$, the proportion of positive instances in the test set				

Table 2d. Confusion Matrix for an Ultra-Conservative Classifier

		Ultra-Conservative Classifier		Total Instances
		Predicted		
		Positive	Negative	
Actual	Positive	0	P	P
	Negative	0	N	N
TP Rate (Recall) = $0 / P = 1$ FP Rate = $0 / N = 0$ Precision = Not Applicable ($TP + FP = 0$) F Measure = Not Applicable Accuracy = $N / (P + N)$, the proportion of negative instances in the test set				

Sensitivity and Specificity

The performance of a binary classifier may sometimes be quantified by its accuracy as described above, i.e. the portion of misclassified classes in the entire set. However, there may be times when the types of misclassifications may be crucial in the classification assignment (Powers, 2011). In these cases, the values for sensitivity and specificity are used in determining the performance of the classifier. Sensitivity or Recall or True Positive Rate (TPR) is the ratio of true positive predictions over the number of positive instances in the entire data set.

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

The specificity or True Negative Rate (TNR) is the ratio of true negative predictions over the number of negative instances in the entire data set.

$$specificity = \frac{TN}{TN + FP}$$

These values can be further analyzed using a Receiver Operating Characteristic Curve (ROC) where the sensitivity is plotted against 1- specificity (Fawcett, 2006). ROC is described further in the next section.

Receiver Operating Characteristic (ROC)

Receiver Operating Characteristic (ROC) analysis has received increasing attention in the recent data mining and machine learning literatures (Fawcett, 2006; Chawla, 2005).

The graph is a plot of the false positive rate (FPR) in the X-axis and the true positive rate (TPR) in the Y-axis.

$$TPR = \frac{TP}{TP+FN} \quad FPR = \frac{FP}{FP+TN}$$

The plotted curve shows the effectiveness of the classifier being tested in ranking positive instances relative to negative instances. The point (0, 1) denotes the perfect classifier, in which the true positive rate is 1, and the false positive rate is 0. Likewise, point (1, 1) represents a classifier that predicts all cases as positive and point (0, 0) represents a classifier which predicts all cases to be negative. Figure 5 shows an example of an ROC curve for a non-parametric classifier. This classifier produces a single ROC point.

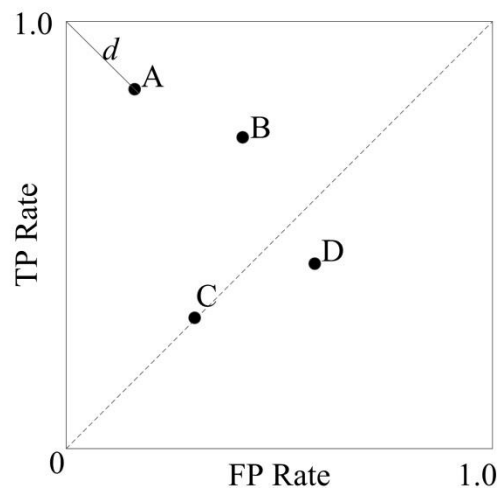


Figure 5. Receiver Operating Characteristic Curve Points

One way of comparing the performance of these classifiers is to measure the Euclidian distance d between the ROC point and the ideal $(0, 1)$. The closer the distance is, the better the classifier performance is. We define d as:

$$d = \sqrt{(1 - TP)^2 + FP^2}$$

There are some types of classifiers, or implementations of non-parametric classifiers, that allow the user to adjust a parameter that increases the TP rate or decreases the FP rate. Under these conditions, the classifier produces a unique (FP, TP) pair for each parameter setting, which can then be plotted as a scatter plot with a fitted curve as shown in Figure 6.

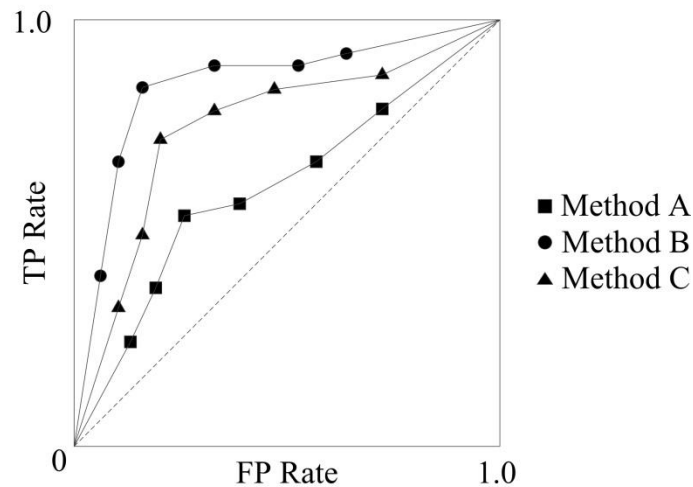


Figure 6. Receiver Operating Characteristic Curves

The main advantage of the ROC graph is that changes in class distribution will not affect the final result. The reason for this is that ROC is based on the TP rate and the FP rate, which is a columnar ratio of the confusion matrix (Bramer, 2007; Fawcett, 2006).

Area Under Curve (AUC)

While the ROC curve is a good visual aid in recognizing the performance of a given algorithm, a numeric value is sometimes needed for comparative purposes. The simplest way of calculating a value for the ROC is to measure the Area Under the ROC Curve (AUC) (Bradley, 1997; Zweig & Campbell, 1993). Since the ROC is plotted inside a unit square, the AUC's value will always be between 0 and 1 (Figure 7).

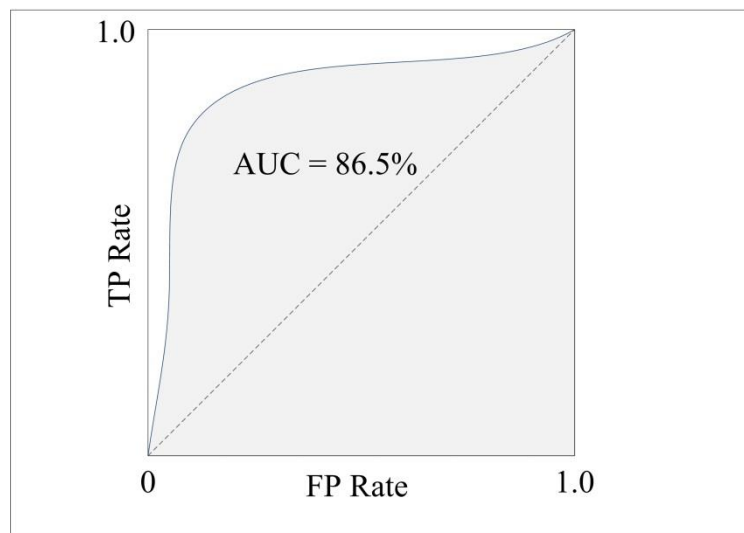


Figure 7. Area Under Receiver Operating Characteristic Curve (AUC)

Graphing an ROC of random guesses will produce a straight line from 0, 0 to 1, 1 and an AUC of 0.5. Based on this, any good classifier should always have an AUC value greater than 0.5.

Based on their empirical studies, which compared the binary classification results of Decision Trees, Naive Bayes, and SVM across 13 distinct data sets, Huang and Ling (2005) concluded that researchers should use AUC evaluation as a performance measure

instead of accuracy when comparing learning algorithms applied to real-world data sets. This recommendation was based on their studies showing that AUC is a statistically consistent and more discriminating performance measure than *accuracy*. They also showed that by using the AUC evaluation to measure profits, a real-world concern, could be easier optimized.

Chapter 3

Methodology

Introduction

This study is a comparative analysis of feature selection methods on classification systems in a real domain setting. As with any data mining exercise, before the data are mined, several key steps need to be performed (Fayyad et al., 1996). These steps, referred to as the preprocessing stage, will account for dealing with missing values, balancing data, discretizing or normalizing attributes depending on which algorithm is used, and finally minimizing the dimensionality of the data set by reducing the number of features with different feature selection methods.

Data Mining Process

The data mining framework followed in this study was the Cross-Industry Standard Process for Data Mining (CRISP-DM), a non-proprietary hierarchical process model designed by practitioners from different domains (Shearer, 2000). The CRISP-DM framework breaks down the data mining process into six phases:

- 1) Understanding the business process and determining the ultimate data mining goals
- 2) Identifying, collecting, and understanding key data sources
- 3) Preparing data for data mining
- 4) Selecting which modeling technique to use
- 5) Evaluating and comparing results of different models against the initial goals
- 6) Deploying Model

One distinctive feature of this framework is that it is more an iterative process than a straight flow design. Practitioners are encouraged to improve results by iterating through the data preparation process and model selection and use.

This research used this framework and provided a structured way to conduct the experiments used in this comparative study. Therefore, it improved the validity and reliability of the final results.

Figure 8 shows the flow used in this research.

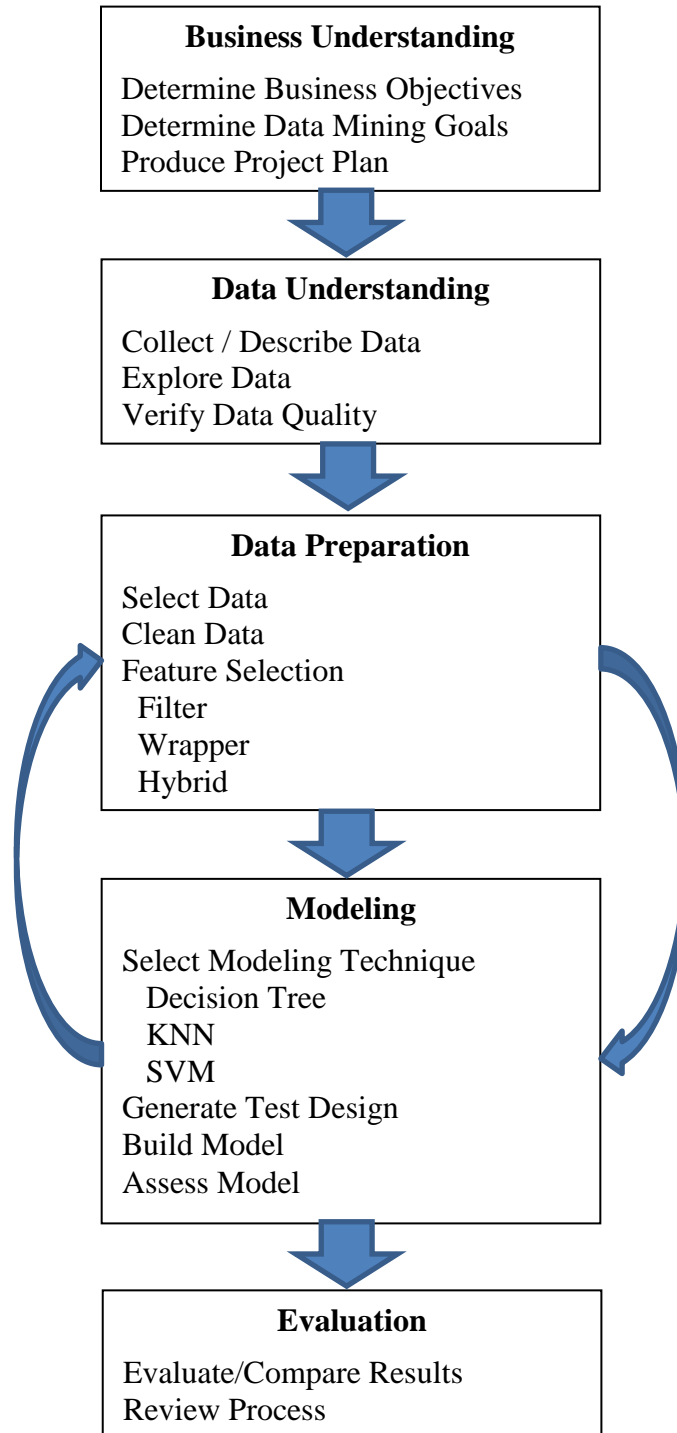


Figure 8. Framework Used in this Research

Data

We used two data sets for this analysis. The first data set to be analyzed was a vehicle service and sales data set that contains information about vehicle services performed and vehicle sales at over 200 auto dealerships. This data set contained thousands of records and thousands of attributes. The goal of this study was to determine the best performing feature selection method and classification algorithm combination that would help automotive dealerships determine if a particular vehicle owner would purchase a new vehicle based on service histories. The second data set was selected from the University of California, Irvine (UCI) Machine Learning Repository (Lichman, 2013) to compare results of our testing against other domains.

Data Acquisition

The data in the vehicle service and sales data set comes from the dealerships' Dealer Management System (DMS) (Appendix A). Data was captured from both the service and sales departments. During a service visit, the vehicle's owner information, vehicle identification number (VIN), and service detail are recorded in the system. Similarly, on the sales side, customer's information and vehicle information are saved into the system after every sale. At the end of each day all transactional data is transferred to a data warehouse server running Postgres SQL. The data is then extracted, transformed, and loaded into SQL Server using a SQL Server Integration Services (SSIS) ETL process (Figure 9).

The data set used in this study was extracted from the following 4 tables:

1. Customer
2. VehicleSales
3. ServiceSalesClosed
4. ServiceSalesDetail

A class label field was added to denote the purchase of a vehicle, new or used, after service was performed. The extraction process will join the data in these relational tables to produce a flat file in a format that the WEKA (Waikato Environment for Knowledge Analysis) (Witten et al., 2011) workbench recognizes. Refer to Appendixes A and B for complete list of attributes and data types.

Data Acquisition Flow

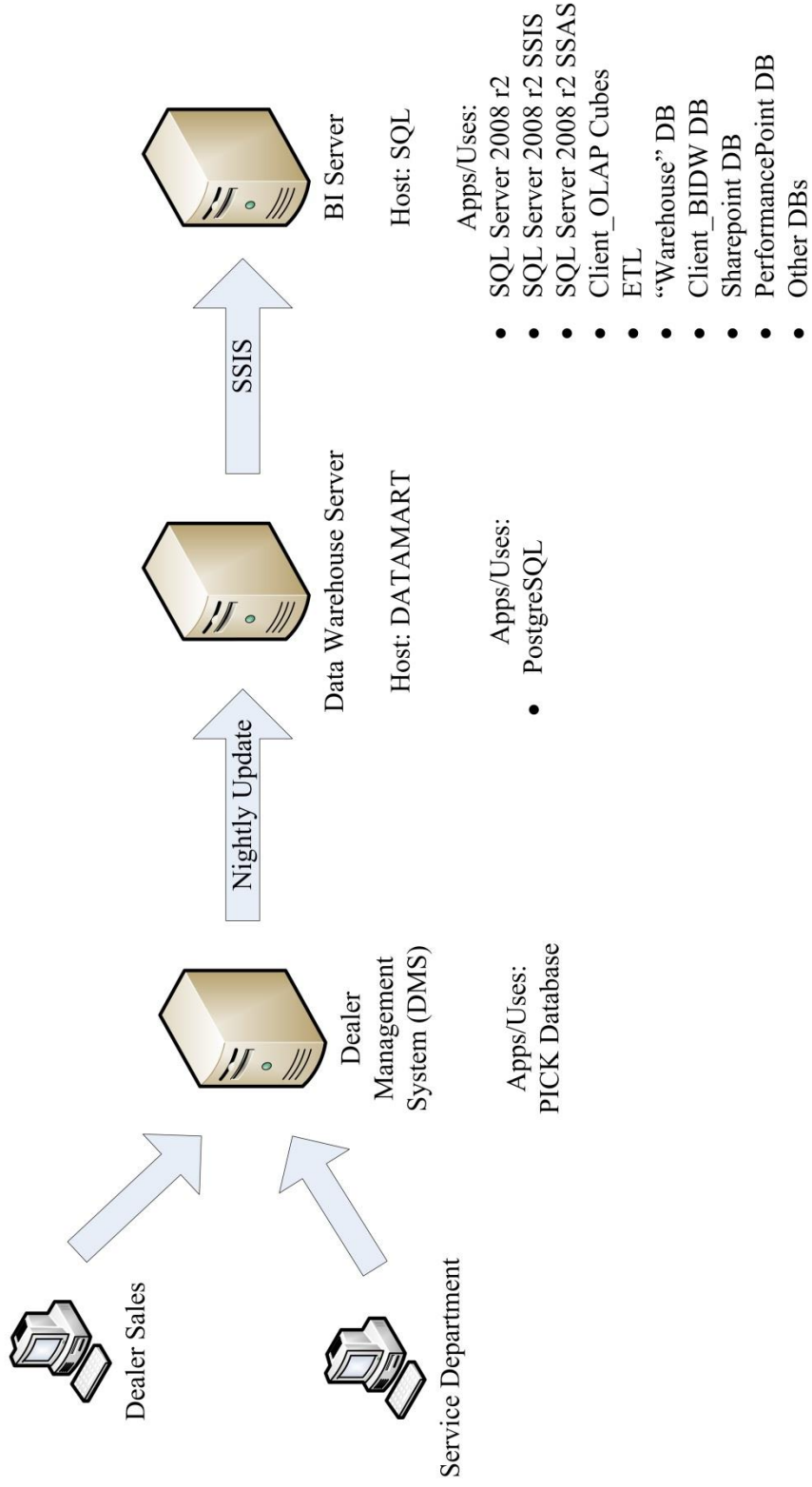


Figure 9. Data Acquisition Flow

Data Pre-Processing

Before running any classification algorithms on the data, the data must first be cleaned and transformed in what is called a pre-processing stage. During this pre-processing stage, several processes take place, including evaluating missing values, eliminating noisy data such as outliers, normalizing, and balancing unbalanced data.

Missing Values

Real world data generally contains missing values. One way of dealing with missing values is to omit the entire record which contains the missing value, a method called Case Deletion. However, Shmueli, Patel, and Bruce (2011) noted that if a data set with 30 variables misses 5% of the values (spread randomly throughout attributes and records), one would have to omit approximately 80% of the records from the data set. Instead of removing the records with missing values, different data imputation algorithms have been studied and compared. Among these methods are Median Imputation, K-NN Imputation, and Mean Imputation (Acuna & Rodriguez, 2004). Median Imputation, as its name implies, replaces the missing values in the record with the median value of that attribute taken across the data set. The K-NN method uses the K-NN model to insert values into the data set. Records with missing values are grouped with other records with similar characteristics which in turn provide a value for the missing attribute. Finally, the Mean Imputation method replaces the missing value with the mean or mode, depending on the attribute type, based on the other values in the data set. Farhangar, Kurgan, and Dy (2008) argued that *mean imputation* was less effective than newer methods, such as those based on Naives-Bayes methods, only when the missing data percentage in the data set surpassed 40%. They also concluded, like others (Acuna & Rodriguez, 2004), that any

imputation method was better than none. In addition, they reported that different imputation methods affected the accuracy classification algorithms differently.

In this study, we used the *mean imputation* method to populate our missing values. This decision was based on the percentage of missing values in our data set (< 20%) and its overall effectiveness in improving the accuracy of classification algorithms. The pseudo code for replacing the missing values is shown in Algorithm 1:

Algorithm 1 Mean Imputation Method

Let $D = \{A_1, A_2, A_3, \dots, A_n\}$

where D is the data set with missing values, A_i is the i^{th} attribute column of D with missing value(s), and n is the number of attributes

For each missing attribute in A_i {

If numeric, impute the mean value of the attribute in class

If nominal (i.e. good, fair, bad), impute the mode value of the attribute in class

}

Imbalanced Data

The problem of imbalanced data classification is seen when the number of elements in one class is much smaller than the number of elements in the other class (Gu, Cai, Zhu & Huang, 2008). If they are left untouched, most machine learning algorithms would predict the most common class in these problems (Drummond & Holte, 2005). Simple queries on our data set had shown us that the data set was imbalanced in respect to the

class label which we were working on. The majority of our records used in this research, 90%, fall into the “Did not buy vehicle” class as opposed to the “Bought a vehicle” class. Processing the data without changes may result in over fitting or under performance of our classifying algorithms. If the data set is small, we could rely on algorithms to synthetically create records to better balance the data. These algorithms, such as Synthetic Minority Oversampling Technique (*SMOTE*) filter (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), do just that. Since our main data set consisted of thousands of records we implemented a *random undersampling* (RUS) to balance our data. RUS removes records randomly until a specified balance (50:50 ratio in our case) is achieved. For instance, if a data set consists of 100,000 records in which 10% belong to the positive class that would leave 90,000 records belonging to the negative class. Undersampling this data set to achieve a 50:50 class ratio would remove 80,000 records and leave us 10,000 records in the positive class and 10,000 records in the negative class. While this method has been argued to remove important data from the classification analysis in small data sets (Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2010) it is effective in larger ones (Lopez, Fernandez, Garcia, Palade, & Herrera, 2013). The pseudo code for RUS is shown in Algorithm 2:

Algorithm 2 Random Undersampling Method

- 1: Determine minimum/majority class ratio desired (i.e. 50:50 ratio)
 - 2: Calculate number of tuples N in majority class that need to be removed
 - 3: Select random tuples in majority class using a structured query language statement such as: `SELECT TOP N FROM tblDealerData ORDER BY NEWID()`
 - 4: Save new data set
-

This sampling occurred before applying the classifier algorithms in WEKA.

Data Normalization

Some algorithms, such as Support Vector Machines and K-NN, may require that the data be normalized to increase the efficacy as well as efficiency of the algorithm. The normalization will prevent any variation in distance measures where the data may not been normalized. A prime example is that data values from different attributes are on a completely different scale, i.e. age and income. Normalizing the attribute will place all attribute within a similar range, usually $[0, 1]$.

In this study we use a *feature scaling normalization method* to transform the values, using the following formula:

$$\delta = \frac{d - d_{min}}{d_{max} - d_{min}}$$

where δ is our normalized value, d is our original value, d_{\max} is maximum value in range, and d_{\min} is minimum value in range.

Data Discretization

Discretization is the process of converting continuous variables into nominal ones. Studies have shown that discretization makes learning algorithms more accurate and faster (Dougherty, Kohavi, & Sahami, 1995). The process can be done manually or by predefining thresholds on which to divide the data. Some learning algorithms may require data to be discretized. An example is the C4.5 decision tree. This tree algorithm does not support multi-way splits on numeric attributes. One way to simulate this is to discretize the attribute into buckets which can in turn be used by the tree.

Feature Selection

Part of this study was to compare the performance of classifiers based on the features selected. By omitting attributes that do not contribute to the efficacy as well as efficiency of the algorithm, we reduced the dimensionality of our data set and improved the processing performance. Tests were conducted on the following feature selection categories:

Filters: Attributes were ranked and chosen independently to classifier algorithm to be used.

Wrappers: Attributes were selected taking the classification algorithm into account.

Hybrid: Attributes were first selected using a filter method then a wrapper method.

Filters

The three filter methods we used in our study were:

1. Information Gain
2. Correlation-based Feature Selection (CFS)
3. Relief-F

These feature selection methods were chosen based on their differing approach in identifying key features.

Information Gain

The *information gain* filter (Quinlan, 1987) measures the attribute's information gain with respect to the class. We began calculating our information gain by calculating the entropy for our class. Entropy was defined as follows (Shannon, 1948):

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

Where D is our data sample, p_i is the proportion of D in respect to class C_i and can be estimated as $\frac{|C_i \cap D|}{|D|}$, and m is the number of possible outcomes. The extreme entropy values for $Info(D)_{max}$ are 1 (totally random) and the minimum is 0 (perfectly classified). $Info(D)$ is the information needed to classify a tuple in D , also known as the entropy of D .

The next step in calculating the information gain is to calculate the expected information required to classify a tuple from D based on the partitioning of attribute A .

The expression is described as follows:

$$Info_A(D) = \sum_{j=1}^v (|D_j|/|D|) \times Info(D_j)$$

where D_j is the subset of D containing distinct value of A , and v is the number of distinct values in A .

The *information gain* measurement can now be calculated as the difference between the prior entropy of classes and posterior entropy (Kononenko,1994):

$$Gain(A) = Info(D) - Info_A(D)$$

Example

Using the data set in Table 3, let's determine the *information gain* of the Windy attribute. First we determine the entropy of the set S . Our response variable (Play) has 9 responses in the Yes class and 5 responses in the No class. We insert these values into our Entropy formula:

$$Entropy(S) = - (9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) = 0.940$$

Next, we calculate the entropy of the different values in the Windy attribute (Yes and No). By analyzing our data, we see that we have 8 entries where Windy = No and 6 entries where Windy = Yes. There are 8 entries where Windy = No, 6 of the entries fall in the Play = Yes class and 2 in the Play=No class. Where Windy=No, we have 3 entries in the Play=Yes class and 3 entries in the Play=No class. Using this information we calculate the entropy of these values:

$$\text{Entropy}(S_{\text{not_windy}}) = - (6/8) \log_2 (6/8) - (2/8) \log_2 (2/8) = .811$$

$$\text{Entropy}(S_{\text{windy}}) = - (3/6) \log_2 (3/6) - (3/6) \log_2 (3/6) = 1.0$$

Finally, we calculate the Information Gain:

$$\text{Gain}(S, \text{Windy}) = \text{Entropy}(S) - (8/14) \times \text{Entropy}(S_{\text{not_windy}}) - (6/14) \times \text{Entropy}(S_{\text{windy}})$$

$$\text{Gain}(S, \text{Windy}) = 0.940 - (8/14) \times 0.811 - (6/14) \times 1.0 = 0.048$$

Once the information gain has been calculated for all attributes and sorted, the attributes which obtain an *information gain* over a predetermined threshold will be added to the feature selection subset.

Table 3. Quinlan (1986) Golf Data Set

Day	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	85	85	No	No
2	Sunny	80	90	Yes	No
3	Overcast	83	78	No	Yes
4	Rain	70	96	No	Yes
5	Rain	68	80	No	Yes
6	Rain	65	70	Yes	No
7	Overcast	64	65	Yes	Yes
8	Sunny	72	95	No	No
9	Sunny	69	70	No	Yes
10	Rain	75	80	No	Yes
11	Sunny	75	70	Yes	Yes
12	Overcast	72	90	Yes	Yes
13	Overcast	81	75	No	Yes
14	Rain	71	80	Yes	No

Correlation-based Feature Selection (CFS)

The main drawback of using the *information gain* filter described above is that it tests each feature individually thus any correlation between features may be ignored. CFS, in turn, looks for features that are highly correlated with the specific classes yet have minimum inter-correlation between the features themselves. We can define CFS as follows:

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}}$$

where r_{zc} is the correlation between the summed features, the class variable, k is the number of features, $\overline{r_{zi}}$ is the average of the correlations between the features and the class variable, and $\overline{r_{ii}}$ is the average inter-correlation between features (Hall, 2000). The inter-correlation here is defined as the ability of a feature to predict another feature. Thus redundant features would be highly correlated.

Relief-F

The last filter method used was the Relief-F method (Kira & Rendell, 1992). This method evaluates the worth of the attribute being tested by randomly sampling instances and detecting the nearest class. The feature's weight is updated by how well it differentiates between classes. Features which have a weight that exceed the predefined threshold will be selected. The formula for updating the weight is as follows:

$$W_x = W_x - \frac{\text{diff}(X, R, H)^2}{m} + \frac{\text{diff}(X, R, M)^2}{m}$$

where W_X is the weight for attribute X , R is a randomly sampled instance, H is the nearest hit, M is the nearest miss, and m is the number of randomly sampled instances.

Based on these parameters, we calculate the difference between two instances for a given feature using the diff function. Running the ReliefF algorithm against a dataset would produce an output of attributes ranked by weight as shown in Table 4. A weight threshold may be used to cut off the number of attributes returned.

Table 4. Top 10 Attributes Ranked by Relief-F Using the UCI Bank Data Set

Rank	Weight	Description
1	0.05200416	Outcome = Success
2	0.05059858	Duration
3	0.04711666	Outcome = Unknown
4	0.02138873	Day of week
5	0.0204481	Housing
6	0.01680847	Month = Aug
7	0.01274343	Outcome = Failure
8	0.01219512	Month = May
9	0.01158064	Month = Apr
10	0.01020042	Month = Nov

Wrappers

Wrapper methods use the classifying algorithm as part of the selection process. The method uses cross validation to estimate the accuracy of the classifying algorithm for a given set of attributes. For our comparative analysis, we ran the wrapper method using Classification Accuracy (ACC) and Area Under Curve (AUC) as performance evaluation measures. Since the wrapper method employs the end classification on its decision, performance is expected to be better. However, since the classification algorithm must be executed for each feature subsets, the cost of computation is high (Gheyas & Smith, 2010). WEKA's "Wrapper" subset evaluator is an implementation of Kohavi's (1997)

evaluator. This implementation performs a 5-fold cross validation on the training data in evaluating the given subset with respect to the classification algorithm selected. In order to minimize bias, this cross validation is run on the internal loop of each training fold in the outer cross-validation. Once the feature set is selected it is run on the outer loop of the cross-validation.

Hybrid

For our hybrid test, we used the features selected by our best performing filter method, and ran them through our wrapper method. We analyzed the performance as well as computational costs.

The performance of each extracted feature set; classification accuracy, AUC, F-Measure, TP rate, and FP rate was compared in a matrix as shown in Table 5.

In addition, a confusion matrix displaying the results of each classification algorithm was presented for each of the feature selection methods that produced the highest accuracy results.

Classification Algorithms

The features selected by our different techniques (filter, wrapper, and hybrid) were tested on three different classification algorithms, K-NN, Decision Tree, and SVM. The classification algorithms were chosen based on their accuracy and different approaches in the learning processes.

k-Nearest Neighbor Classifier (K-NN)

The first classification algorithm we ran our data through is the k-Nearest Neighbor classifier (K-NN). K-NN is one of the easiest and most well-known supervised learning algorithms (Li & Lu, 2009). The algorithm classifies an unknown instance and predicts its class as same as the majority of its k nearest neighbors (Figure 10). The basic algorithm for K-NN is shown in Algorithm 3.

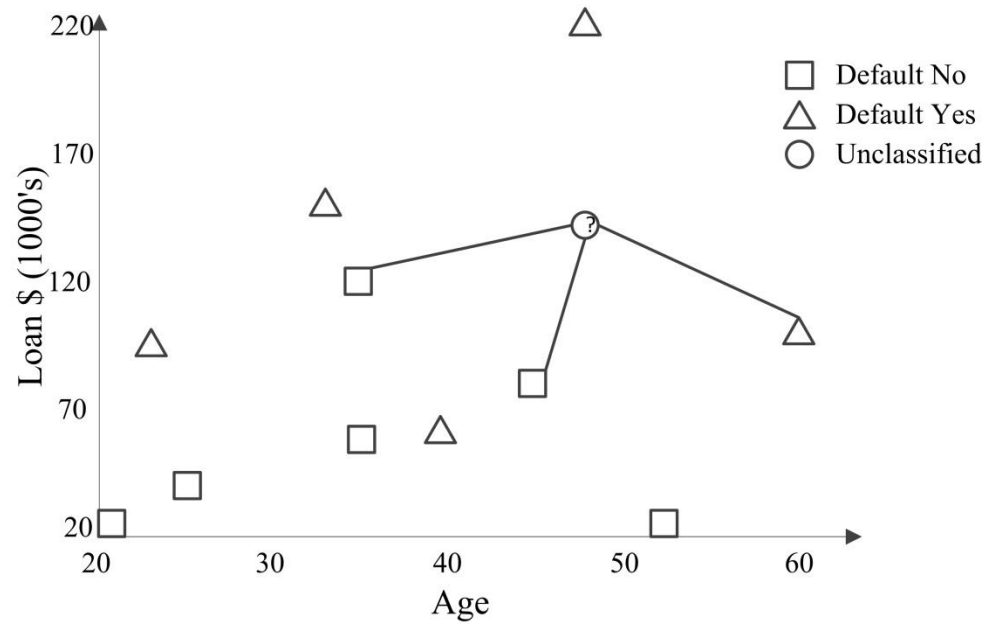


Figure 10. K-NN Visualization with $k = 3$

Algorithm 3 K-NN Classification

input: $D = \{(x_1, c_1), \dots, (x_n, c_n)\}$

1: **begin**

2: $y = (y_1, \dots, y_p)$ new instance to be classified

3: compute $d(x_i, y)$ for each (x_i, c_i)

4: sort $d(x_i, y)$ from lowest to highest, $i = (1, \dots, n)$

5: select the k points nearest to y : D_x^k

6: assign to y the most frequent class in D_x^k

7: **end**

where D is our data set, k is number of neighbors, p number of features,

$d(x_i, y)$ is the Euclidean distance, and n is the number of values

Our implementation of K-NN is a variant of the original K-NN. The K-NN algorithm we used added a weight value to the distance measured to the neighbors. This algorithm, proposed by Dudani (1976), showed higher accuracy results in comparison to the existing K-NN approach. Our proposed procedure to our K-NN implementation were as follows:

1. Selecting a k value
2. Determining a distance measure to use
3. Normalizing data
4. Assigning a weight formula

The k value selected will affect the classification's performance greatly (Wu et al, 2008). During training, we proposed to use a 10-fold cross validation methodology with a range of k values from 1 to 20. This methodology ran the K-NN algorithm for each value in the range. Once processed, we selected the k value with the best accuracy for our testing phase.

Our next step was to implement a distance formula to be used when measuring the distance between our unknown instance and those of its neighbors. We have decided to use the Euclidean formula for this purpose. The formula is defined as follows:

$$d(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

where testing vector $x = x_1, x_2, \dots, x_n$ and training vector $y = y_1, y_2, \dots, y_n$ in \mathbb{R}^2 vector space.

Example

A website has collected data from its customer base to determine which type of membership an individual would most likely buy. For the sake of simplicity, we have limited the number of attributes in this example to two: Customers Age (Age), and Annual Salary (Salary). A real world implementation would most likely have tens if not hundreds of variables. The data used in this example is shown in Table 6.

Table 6. Web Data Set

Age	Salary (\$)	Membership
25	40,000	Standard
35	60,000	Standard
45	80,000	Standard
20	20,000	Standard
35	120,000	Standard
52	18,000	Standard
23	95,000	Premium
40	62,000	Premium
60	100,000	Premium
48	220,000	Premium
33	150,000	Premium

For example, a new customer, age 48, and a salary of \$148,000, applies for membership.

A decision on what membership status to grant will be made based on a K-NN classification algorithm with $k = 3$. The first step is to calculate the distances from existing observations to the unclassified one. Once all distances are calculated, we select 3 closest observations ($k = 3$) and classify our unknown observation based on these (Table 7).

Table 7. Web Data Set with Distance Calculated

Age	Salary (\$)	Membership	Distance
25	40,000	Standard	102000
35	60,000	Standard	82000
45	80,000	Standard	62000
20	20,000	Standard	122000
35	120,000	Standard	22000
52	18,000	Standard	124000
23	95,000	Premium	47000
40	62,000	Premium	80000
60	100,000	Premium	42000
48	220,000	Premium	78000
33	150,000	Premium	8000

The process generates 3 closest neighbors ($k = 3$) denoted in bold.

Of the 3 closest neighbors, we have 2 observations with Membership = Premium and 1 observation with Membership = Standard. Based on majority votes we would classify our unknown observation as a Membership = Premium and offer it accordingly.

If large attributes are left untouched, they will affect the distance calculation more than those in smaller scales. In the example above, we see that salary amounts would have a greater impact on the distance calculation than that of client's age. In order to prevent this, all attributes need to be normalized before implementing the classifier (Table 8).

Table 8. Data after Normalization (bold denotes closest distances)

Age	Salary	Membership	Distance
.0125	.11	Standard	0.765
.375	.21	Standard	0.520
.625	.31	Standard	0.316
0	.01	Standard	0.925
0.375	0.50	Standard	0.343
0.8	0.00	Standard	0.622
0.075	0.38	Premium	0.667
0.5	0.22	Premium	0.444
1	0.41	Premium	0.365
0.7	1.00	Premium	0.386
0.325	0.65	Premium	0.377

Using this new information we see that the client would be offered a Standard membership instead of the Premium offered before this change.

When the data set is imbalanced, the majority voting may produce invalid results. The probabilities of having members of the majority class closer to an unknown instance are greater, thus they dominate the prediction of the new value. In order to prevent this, we can apply a weight formula to the equation. Dudani (1976) showed that applying a weight to the K-NN algorithm significantly improved the results. We used an inverse weight formula. That is, the neighbors were weighted by the inverse of their distance when voting. The formula is defined as follows:

$$w_j = \frac{1}{d_j}, \quad d_j \neq 0.$$

where w_j is weight assigned to j^{th} nearest neighbor and d_j denotes the distance from neighbor to unclassified sample.

As the distance d_j approximates 0 the weight value w_j increases. This will give the neighbors that are closer to our unknown instance a stronger weight when computing distance (Table 9).

Table 9. Data Set with Weighted Distance

Age	Salary	Membership	Distance	Weighted Distance
.0125	.11	Standard	0.850	0.972
.375	.21	Standard	0.515	0.715
.625	.31	Standard	0.309	0.550
0	.01	Standard	0.922	0.959
0.375	0.50	Standard	0.343	0.586
0.8	0.00	Standard	0.618	0.784
0.075	0.38	Premium	0.665	0.815
0.5	0.22	Premium	0.438	0.658
1	0.41	Premium	0.361	0.597
0.7	1.00	Premium	0.390	0.628
0.325	0.65	Premium	0.377	0.614

Table 9 shows that after using a weighted distance, 2 out of the 3 closest observations belong to the Standard membership class. Therefore, the Standard membership would be offered to the new client.

As part of our comparative analysis we ran our data set against the K-NN algorithm, before and after the pre-processes. We compare the original data with that after K-NN processes with the weighted distance.

Decision Tree

Decision trees have become a popular choice in classification due to the features of understanding and visualization. Users with no technical background can look at a decision tree's output and easily follow the flow of decisions. The most commonly used decision trees today are the Iterative Dichotomiser 3 (ID3), C4.5, and C5.0 (Quinlan,

1993); Classification and Regression Trees (CART) (Breiman, Friedman, Olshen, & Stone, 1984), and Chi-Square Automatic Interaction Detector (CHAID) decision tree (Kass, 1980).

For the purposes of this study we used WEKA's J48 implementation of the C4.5 (release 8) algorithm. The C4.5 algorithm makes several key improvements on the ID3 algorithm.

Among these are:

1. Ability to handle missing values
2. Accept discrete and continuous data. Continuous data is discretized prior to use.
3. Post pruning

These will be discussed in more detail in the following.

The general algorithm for building decision trees is (Xiaoliang, Jian, Hongcan, & Shangzhuo, 2009):

1. Check for base cases;
2. For each attribute A , find the normalized information gain from splitting on A ;
3. Let a_{best} be the attribute with the highest normalized information gain;
4. Create a decision *node* that splits on a_{best} ;
5. Recur on the sub lists obtained by splitting on a_{best} , and add those nodes as children of the node.

Our proposed procedure for building and using our decision tree are as follows:

1. Preprocess data
2. Select split criteria
3. Determine minimum number of splits
4. Prune the tree

Preprocess data

In order to improve our classification accuracy we must first analyze the raw data. Factors, such as missing data and numerical attributes, must be addressed. The C4.5 algorithm handles attributes with missing values by not incorporating them into the information gain calculation. In our study, missing values were handled in the main pre-processing stage, as described earlier. In addition to missing values, numerical data must be discretized for better results. If numerical data is not discretized, the tree will perform a binary split on the attribute.

For example, we could discrete the Temperature and Humidity attributes as follows:

If temperature < 70 degrees then
Temperature is cold
If temperature is between 70 and 80 degrees then
Temperature is mild
If temperature > 80 degrees then
Temperature is hot

Likewise, for Humidity

If humidity < 80 then
Humidity is normal
If humidity ≥ 80 then
Humidity is high

Table 10 shows the data after discretization.

Table 10. Data after Discretization

Day	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	No	No
2	Sunny	Hot	High	Yes	No
3	Overcast	Hot	High	No	Yes
4	Rain	Mild	High	No	Yes
5	Rain	Cold	Normal	No	Yes
6	Rain	Cold	Normal	Yes	No
7	Overcast	Cold	Normal	Yes	Yes
8	Sunny	Mild	High	No	No
9	Sunny	Cold	Normal	No	Yes
10	Rain	Mild	Normal	No	Yes
11	Sunny	Mild	Normal	Yes	Yes
12	Overcast	Mild	High	Yes	Yes
13	Overcast	Hot	Normal	No	Yes
14	Rain	Mild	High	Yes	No

Split Criteria

Like other inductive decision tree algorithms, in order to build a classification tree model, the C4.5 tree begins at the root node. At this point, the algorithm chooses the attribute that best splits the data into different classes. The split is determined by the attribute which has the highest normalized information gain.

For example, to begin building our decision tree, we must first determine its root node. In order to do that, we must first calculate the information gain of all attributes. We do this by first finding the entropy of the attribute then calculating the information gain, as we explained earlier.

After processing, we determined the following:

$$\text{Gain}(S, \text{Windy}) = 0.048$$

$$\text{Gain}(S, \text{Temperature}) = 0.029$$

$$\text{Gain}(S, \text{Outlook}) = 0.246$$

$$\text{Gain}(S, \text{Humidity}) = 0.151$$

Checking the results, we decided to make the Outlook as the root node because the attribute has the largest information gain. The next step is to branch out from our root node. Since Outlook has three possible outcomes (overcast, sunny, and rain), we will first create three branches off the Outlook node. Next, we determine which attribute is tested at each of the branches. Once again, we calculate the information gain for the remaining attributes, and continue growing the trees until we run out of attributes, or the data is classified perfectly. Figure 11 shows the final decision tree.

Splits

The minimum number of instances a split may have is a user defined parameter in C4.5. For our study, the number of minimum instances per node was set to 2.

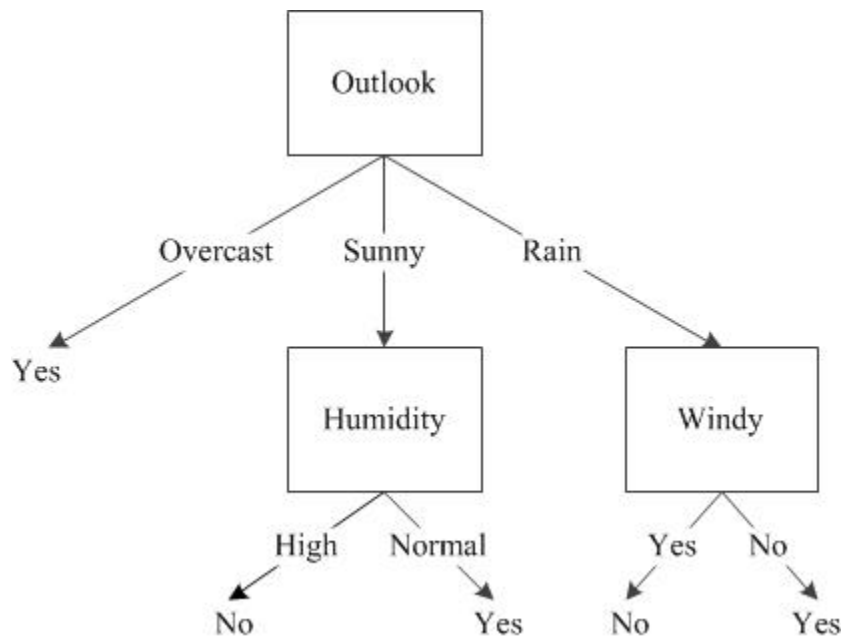


Figure 11. Final Decision Tree

Pruning

Fully grown decision trees might contain many unnecessary nodes and cause overfitting. The process of cleaning up a tree and making it more efficient is called pruning. Pruning may be done while the decision tree is being built or after it has been fully grown. Pruning a tree while it's being built is called pre-pruning. The logic here is that only those attributes that make the most effective decisions at the time are included in the tree. The main drawback in this method is that no correlation among features is considered. The C4.5 algorithm uses a post-pruning method called subtree raising. The idea here is to replace a parent node with the child node if the error rate of validation does not decrease (Figure 12). We do this by comparing the estimated error rate of the subtree with that of its replacement.

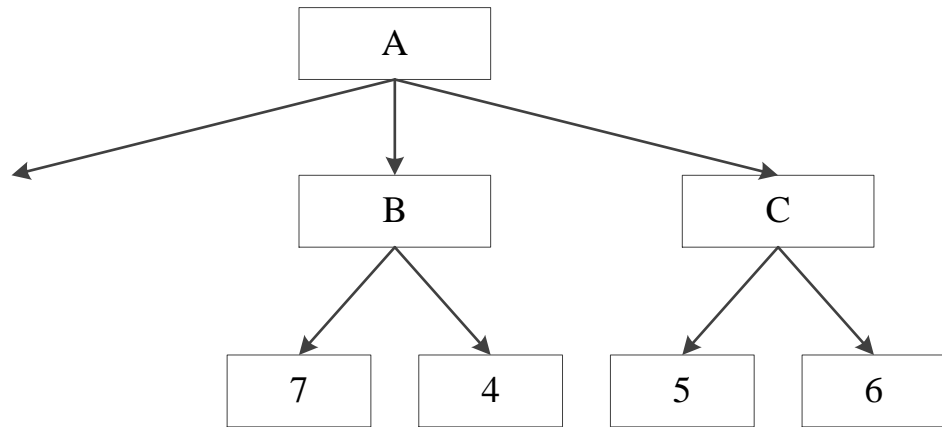
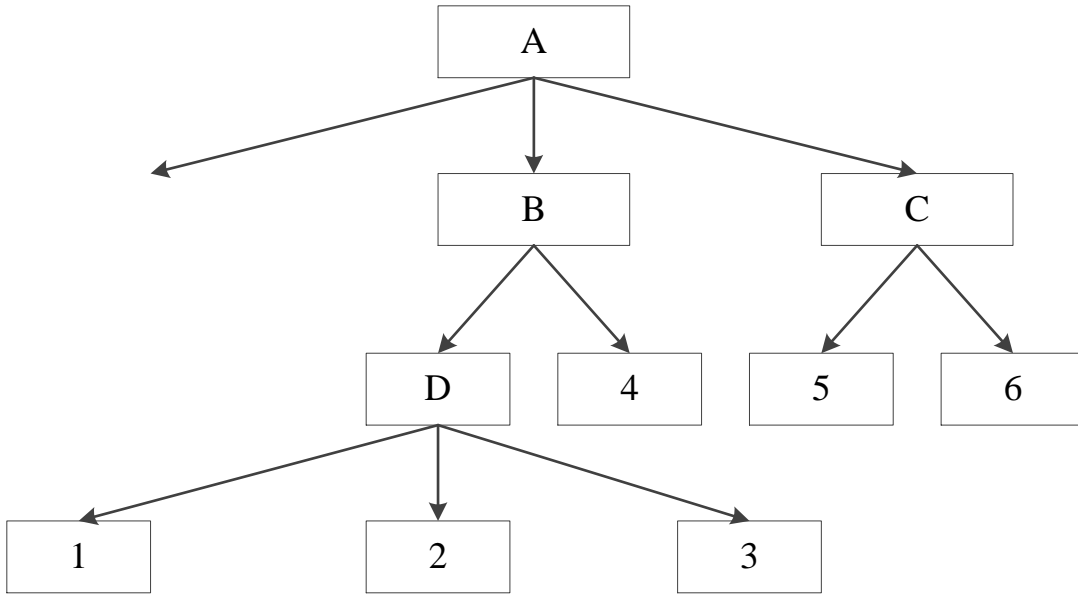


Figure 12. Subtree Raising - Subtree D is Replaced by a Single Node

We estimated the error rate using the following formula (Frank, 2000):

$$e = \frac{\left(f + \frac{z^2}{2N} + z \times \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right)}{\left(1 + \frac{z^2}{N} \right)}$$

where:

f is the error on the training data

N is the number of instances covered by the leaf node

z is the z-score based on the confidence interval desired

Example

Let's assume the D node in our un-pruned tree in Figure 11 has three children (1, 2, and 3). The class breakdown for each child is:

Child 1 – 2 Play, 4 Don't Play

Child 2 – 1 Play, 1 Don't Play

Child 3 – 2 Play, 4 Don't Play

Using a confidence level of 75% ($z = 0.69$) we can calculate the error rates at D and each of the child nodes as follows:

Node D: $f = 5/14$, error rate = 0.46 (5 plays over 14 instances)

Child 1: $f = 2/6$, error rate = 0.47 (2 plays over 6 instances)

Child 2: $f = 1/2$, error rate = 0.72 (1 play over 2 instances)

Child 3: $f = 2/6$, error rate = 0.47 (2 plays over 6 instances)

Combining the error rates of the children (using a ratio of 6:2:6) gives us

$$(6/14 \times 0.472 / 14 \times 0.72 \times 6/14 \times 0.47) = 0.51$$

Since the error rate of the parent D is less than the children's rate we do not gain by having the children and we prune back to D.

This study will use a confidence factor ranging from 0.1 to 1.0 incremented by 0.2. A lower confidence factor will equate to a larger error estimate at each node, thus it increases the chances that the node will be pruned.

Support Vector Machines (SVM)

Support Vector Machines (SVM) or Support Vector Classification (SVC) that, is sometimes referred to, is one of the most popular and successful classification algorithms (Carrizosa, Martin-Barragan, & Morales, 2010). Given a training set of instance labeled pairs (x_i, y_i) where $x_i \in R^n$ (the data space) and $y_i \in \{-1, 1\}$, where the y_i is either 1 or -1, indicating the class (positive or negative) that the point x_i belongs to. SVMs (Boser, Guyon, & Vapnik, 1992; Vapnik, 1995) work by finding a hyper-plane that maximizes the distance between the classes (y_i) being investigated (Figure 13).

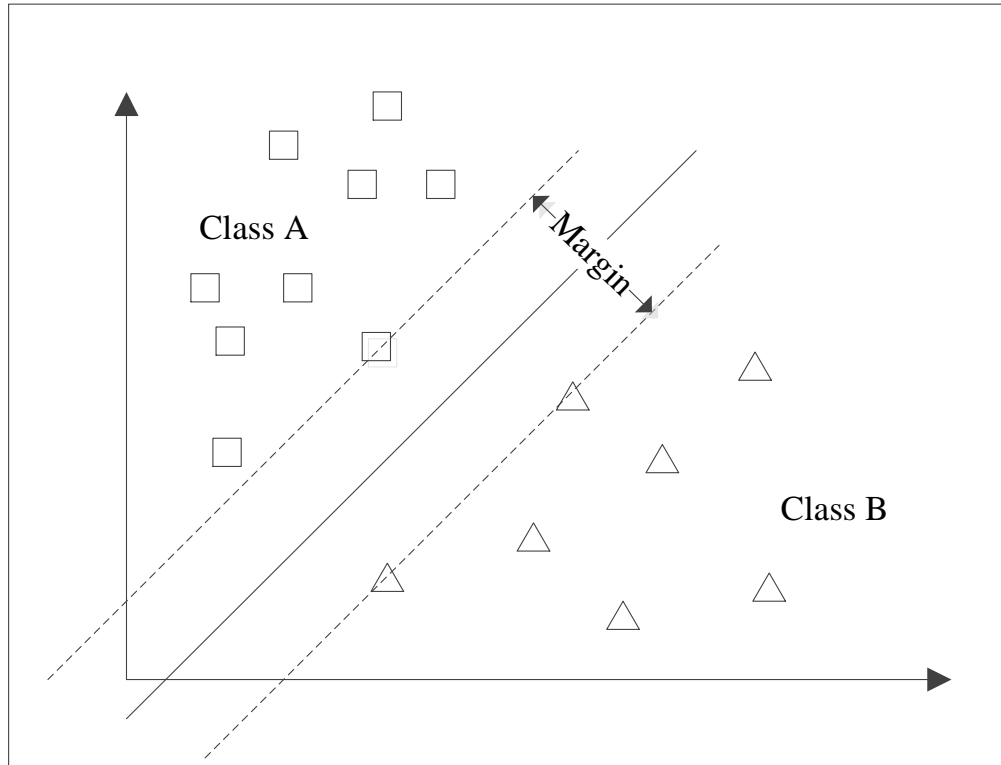


Figure 13. Support Vector Machine for a Binary Class

The most commonly used SVM models incorporate a kernel function into the equation in order to account for training data that cannot be linearly separated.

Our proposed procedures for applying the SVM classification are as follows:

1. Preprocess the Data
2. Select SVM model
3. Select a kernel function
4. Tune the parameters
5. Test the model for deployment

Preprocess the Data

Before we import our data into our SVM classifier, we must first pre-process the data. SVMs like K-NN classification algorithms cannot handle text attributes. The first step would convert all text and nominal attributes to contain a value, either 0 or 1. For example, if we have an attribute Vehicle_Color with values red, blue, or green, then conversion process would create a new attribute called Veh_ColorRed, Veh_ColorBlue, and Veh_ColorGreen with values of 1 if it is true or 0 if it is false. In addition, all numerical values must be normalized to a range of [0, 1] to prevent attributes with larger numbers from dominating the process over those with smaller values.

Select SVM Model

Based on previous research (Bennett & Campbell, 2000; Brekke & Solberg, 2008) we have decided to use the C-Support Vector Classification (C-SVC) algorithm (Cortes & Vapnik, 1995) in this study.

The main objective of the C-SVC algorithm is to solve the following optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2} \omega^T \omega + C \sum \xi, \quad \text{where } C > 0 \\ \text{subject to} \quad & y_i(\omega^T \Phi(x_i) + h) \geq 1 - \xi_i, \quad \text{where } \xi > 0 \end{aligned}$$

where x_i is our training vector, $\Phi(x_i)$ maps x_i into a higher dimensional space, C is the cost parameter, ω the vector variable, and $\xi > 0$ the slack variable.

Select a kernel function

Several different kernel functions have been proposed and studied in the past (Hsu, Chang, & Lin, 2011; Herbrich, 2001; Lin & Lin, 2003). Among the most popular functions are:

1. Linear $K(x_i, x_j) = x_i^T x_j$
2. Polynomial $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$, for $\gamma > 0$
3. Sigmoid $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$
4. Radial Basis function. $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, for $\gamma > 0$

where γ is the gamma parameter, d is degree, and r is the kernel projection

The kernel function selected to be used in this study is the Radial Basis function (RBF), or Gaussian kernel which, is sometimes referred to. This decision is based on previous studies by Keerthi and Lin (2003) which showed that given a certain cost of error (C) and gamma (γ) values RBF could replicate the results of a linear function. Similarly, Lin and Lin (2003) showed that RBF behaved the same as sigmoid function when (C, γ) were in a certain range.

Tune the parameters

One of the advantages of the SVM class type algorithms is that there are only a few parameters that the algorithm needs to optimize. The SVM model we have chosen, C-SVC, has two parameters which we can work with; the cost of error (C) and the gamma (γ) value.

The cost of error determines how many observations we will use in determining our margin. A larger value of C uses those observations closest to the separating line (Figure

14a), while a small value of C uses many observations in calculating the margin (Figure 14b). The gamma (γ) value determines the curvature of our separation line and possible isolation of distinct groups in our SVM model as applied to our data set.

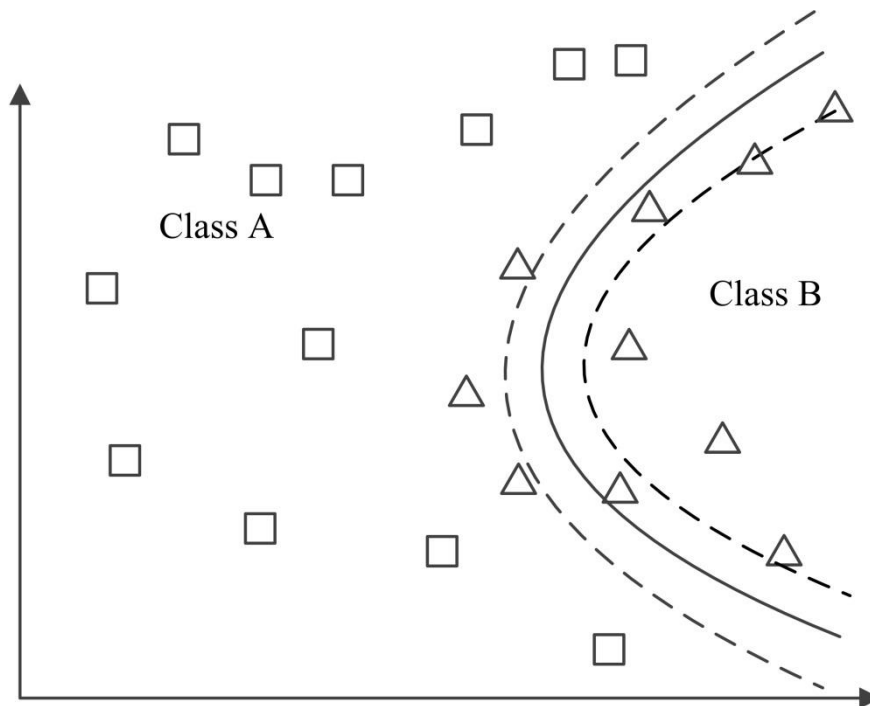


Figure 14a. SVM Optimized by Grid Search of Parameters

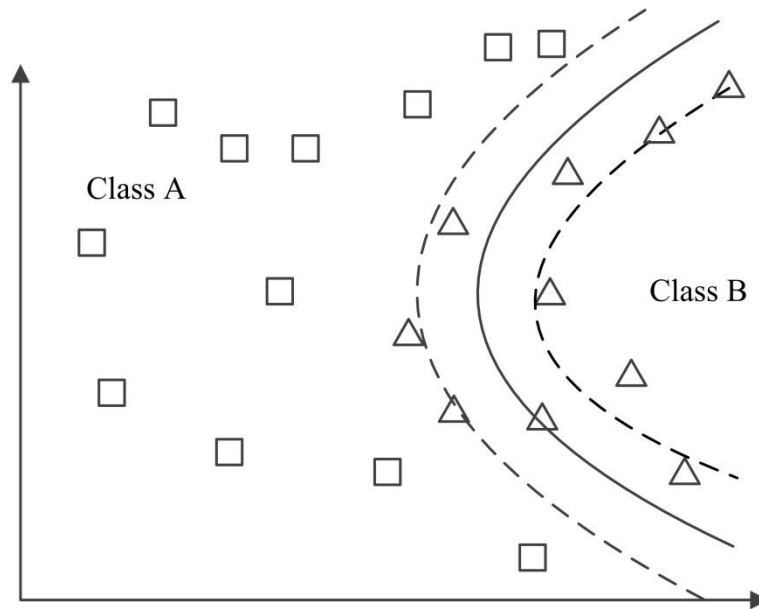


Figure 14b. Grid Search with a Decrease in C Value

Instead of optimizing each parameter individually through cross validation, a grid search was used. The grid search method allows us to set a range of values for cost of error (C) and gamma (γ) and find the best value combination through a 10-fold cross validation. To further improve the results, we refined the grid using values in a range of our first results. For example, if our initial range for C was from 1 to 30 and the optimized result was 9, we could rerun our grid search with C values from 8.5 to 9.5 in increments of 0.1. Our optimization can be based on the results of Accuracy or Mean Absolute Error (Chapelle, Vapnik, Bousquet, & Mukherjee, 2002).

Test

The final step in our process is to run our SVM algorithm by using the optimized parameters against our test data set. Different set of features were used in our analysis and compared to the other classification algorithms being studied in this research.

Cross Validation

All our classification methods were tested by using an n -fold cross validation. This test splits the data set into n equal subsamples. One subsample is kept for validating the data, while the remaining $n - 1$ subsamples are used for training. This process is repeated until all subsamples have been used as validation. For example, applying a 5-fold cross validation on a data set with 100 entries the data set would be split into 5 equal folds. In the first round, the first fold of data (20 entries) is kept for testing and the other 4 (80 entries) are used for training. In the next round, the second fold is reserved for testing and the remaining 80 entries are used for training. This process continues until all n folds are used. The final results are averaged across to produce a single result. Figure 15 illustrates a 5-fold cross validation. In our experiments we performed the cross validation with $n = 10$.

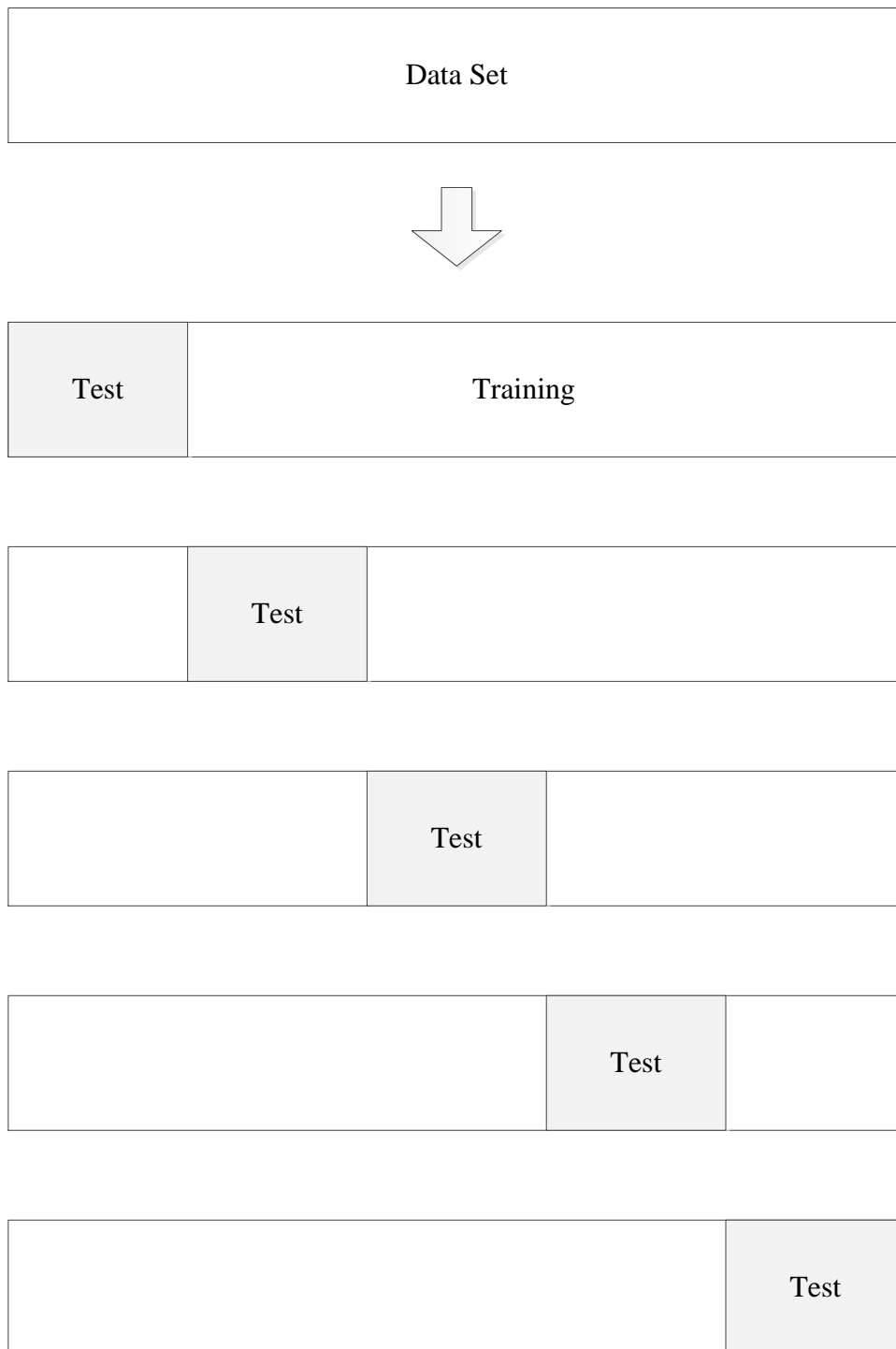


Figure 15. 5-Fold Cross-Validation

Performance Evaluation

In our analyses, we evaluated effects of feature selection on classification performance. Classifiers accuracy (ACC), F-Measure, Receiver Operating Characteristic (ROC) graphs, and Area Under Curve (AUC), were used as our performance measures. Performance evaluators were compared against each other in terms of classification results.

Confusion Matrix

A confusion matrix (Chawla, 2005) is a table that contains information about actual and predicted classifications for any given classification algorithm. For example, a confusion matrix for a classification model used on a data set of 100 entries is shown in Table 11. We can easily see that the algorithm classified 59 positive entries correctly, and 12 negative entries correctly. However, it misclassified 2 positive entries as negative, and 27 negative entries as positive.

Table 11. Confusion Matrix for a Binary Classification Model

		Predicted	
		Positive	Negative
Actual	Positive	59	2
	Negative	27	12

The performance of the algorithm is calculated based on these numbers, as described in the following sections.

Accuracy

Accuracy represents the percentage of correctly classified results. It can be easily calculated as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Example

Using the confusion matrix data in table 8 we can calculate accuracy as follows:

$$Accuracy = (59+12) / (59+27+12+2)$$

$$Accuracy = .71$$

The higher the accuracy rate is, the better our classification model is performing.

Sensitivity and Specificity

In addition to the accuracy, we calculated the sensitivity (True Positive Rate) and specificity (True Negative Rate) of each classifier using data from the confusion matrix.

We calculated as follows:

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

$$specificity = \frac{TN}{TN + FP}$$

F-Measures

The F-Measure is the harmonic mean of precision and recall. Precision is the number of positive examples classified over all the examples classified. Recall is the number of positive examples classified over all the positive examples.

$$precision = \frac{TP}{TP+FP}$$

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

Based on these definitions F-measure is defined as follows:

$$f - measure = \frac{2 \times precision \times recall}{precision + recall}$$

Example

Again, using the confusion matrix in Table 8, we have the results as follows:

$$Precision = 59 / (59 + 27) = .686 \text{ and } Recall = 59 / (59 + 2) = .967$$

$$F-Measure = 2 \times .686 \times .967 / (.686 + .967) = .803$$

The confusion matrices for the classifiers being tested in this report were set up and computed using WEKA's KnowledgeFlow environment. The following steps were followed for each data set in the experiment:

1. *Arffloader* - Loads the data set
2. *ClassAssigner* - Select class attribute
3. *CrossValidationFoldMaker* - Run data through cross-validation
4. Send training data and test data to our classifiers
5. *ClassifierPerformanceEvaluator* - Evaluate classifier performance
6. *TextViewer* - Display results

Figure 16 shows the flow of data in WEKA's KnowledgeFlow.

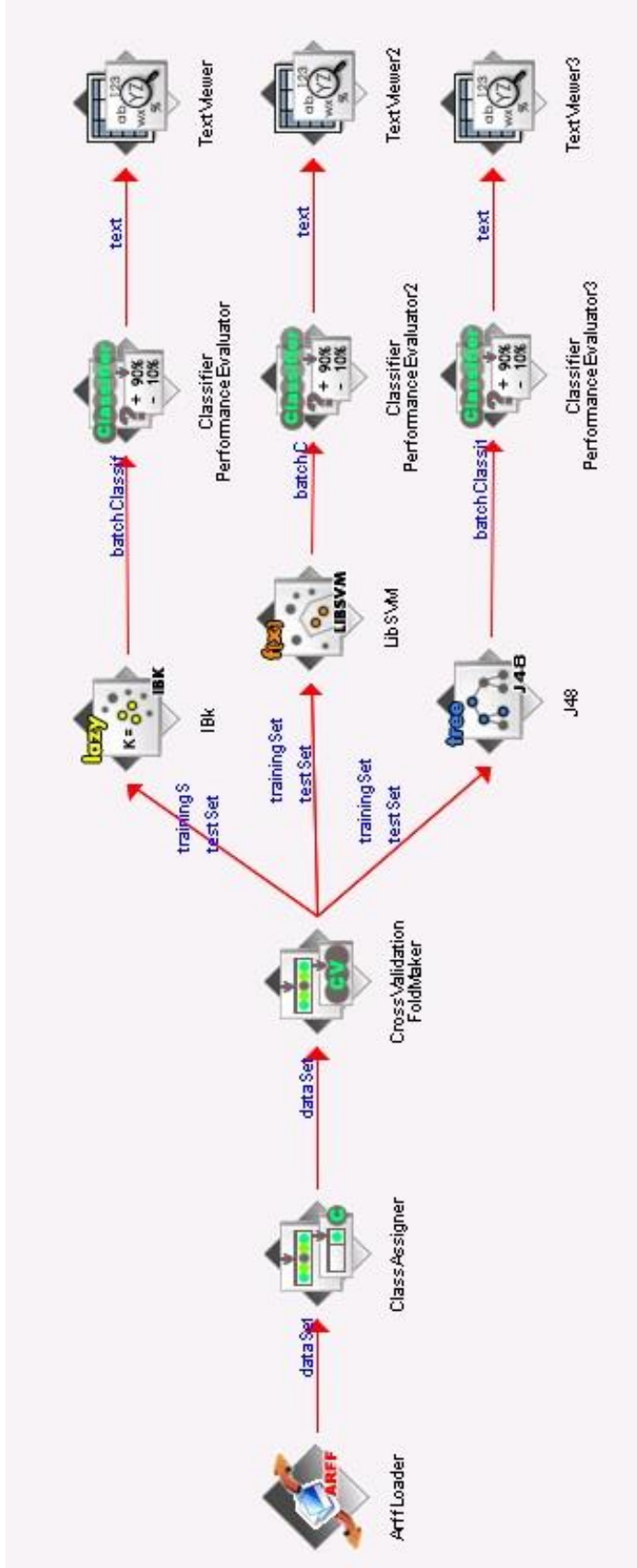


Figure 16. Classification Flow with WEKA's KnowledgeFlow

Receiver Operating Characteristic (ROC)

ROC curves encapsulate all the information provided by the confusion matrix in a visual format. The plot represents the classifiers ability for correctly identified positive labels and incorrectly identified negative labels. The major advantage of using the ROC curve over the previously mentioned measures is that the ROC curves provides performance values over all possible thresholds. The results of our classifier performance are plotted against different feature selection methods and across different classification algorithms. To better compare the ROC curves produced by our algorithms we charted them simultaneously using WEKA's workflow manager (Figure 17). We began to process our dataset by using the following steps:

1. *ArffLoader* – Loads the data set
2. *ClassAssigner* - Select class attribute
3. *ClassValuePicker* - Select which class label (Positive or Negative) to plot
4. *CrossValidationFoldMaker* – Split training set and test set into folds using cross-validation
5. Select classifiers (IBk – KNN, libSVM – SVM, J48 – C 4.5 Decision Tree)
6. Send training data and test data from cross validation to our classifiers
7. *ClassifierPerformanceEvaluator* - Evaluate classifier performance
8. *ModelPerformanceChart* - Plot ROC curves

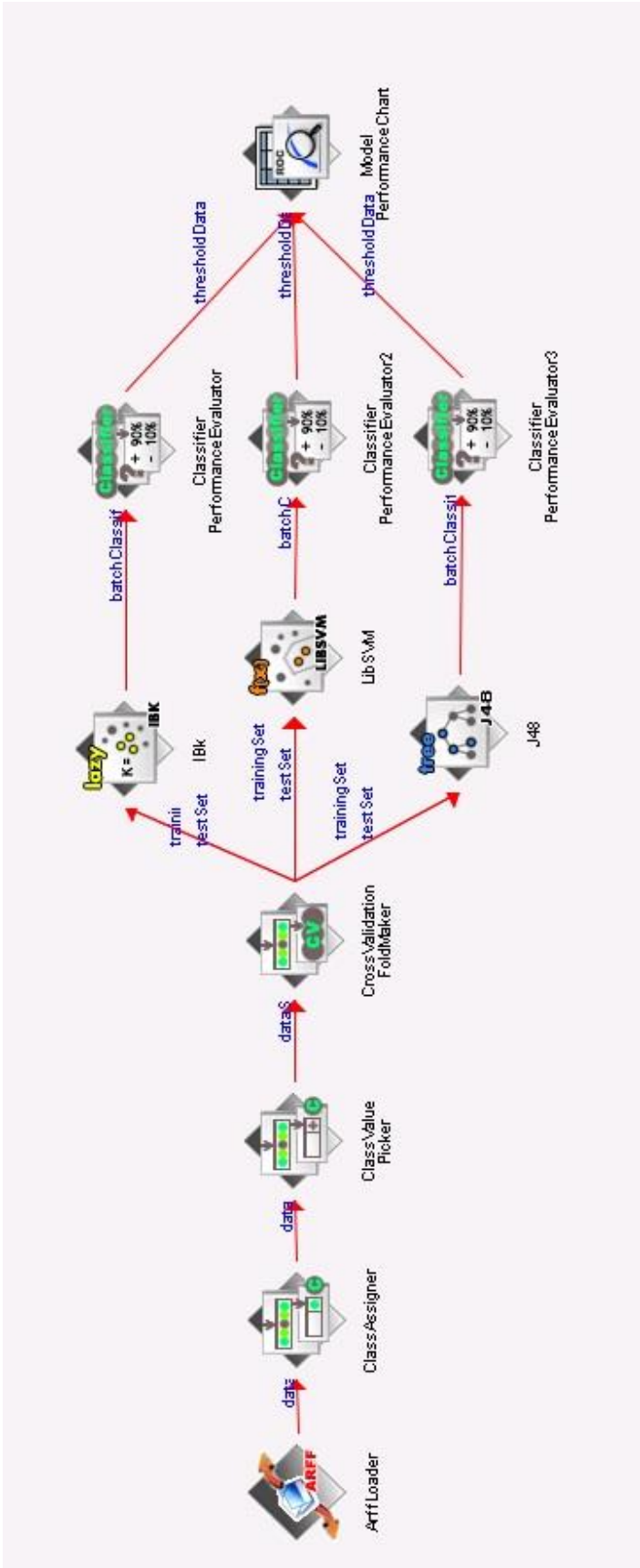


Figure 17. WEKA Multiple ROC Workflow

To produce multiple ROC points using the SVM classifier, the *parameter probability estimates* was set to true. The resulting graph is shown in Figure 18.

An alternative to use WEKA's plotting utility (the last step) is to export the ROC points from WEKA, import the points into Excel, and plot them. This method was used due to Excel's better graphing capabilities.

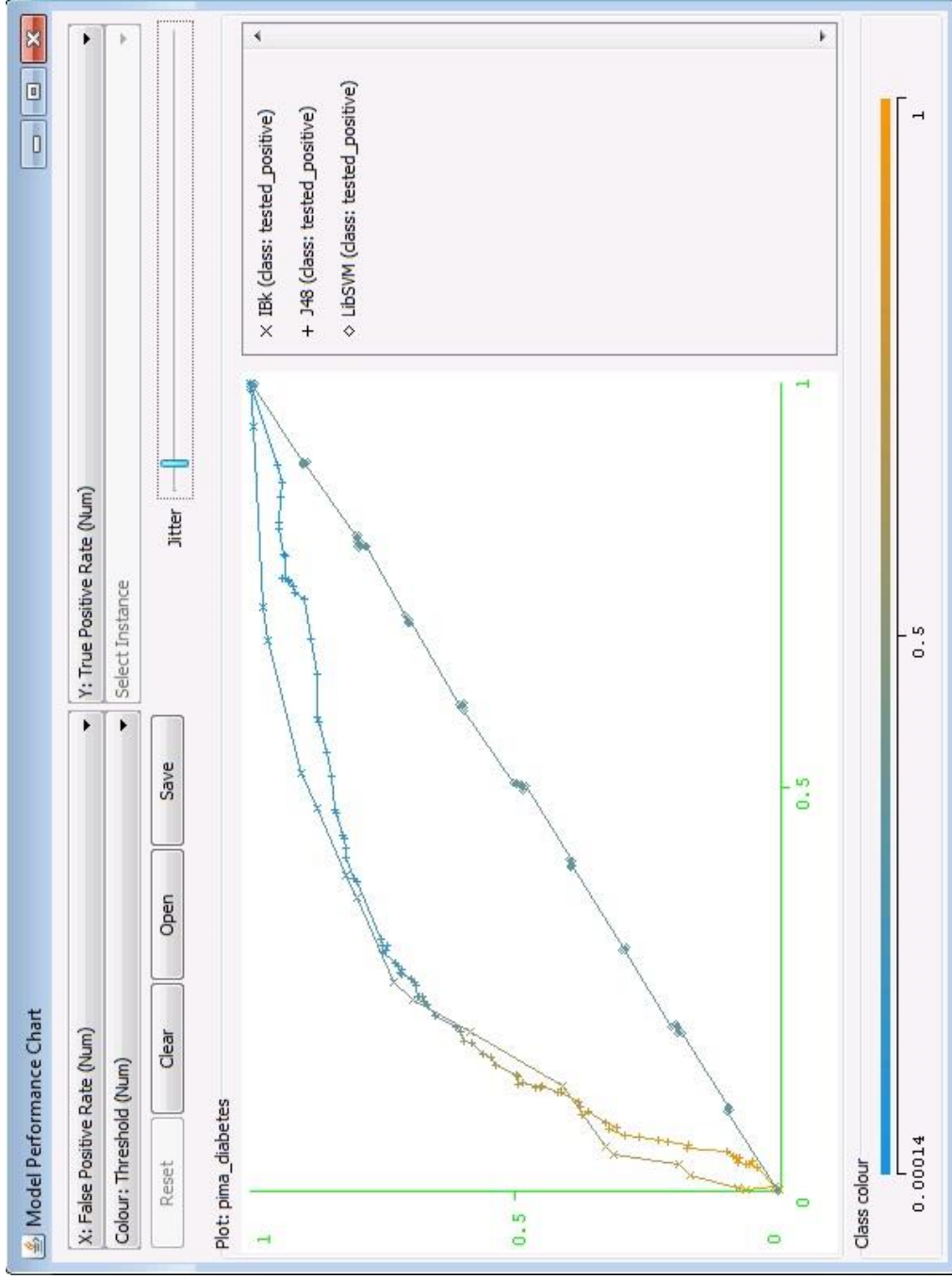


Figure 18. Multiple Receiver Operating Characteristic Curves for IBK – KNN, J48 – C4.5 Decision Tree, and LibSVM – SVM Classifiers

Area Under Curve (AUC)

Another way of calculating performance is by measuring the area under the ROC curve (AUC). This method allows us to easily compare different ROCs in our analysis (Figure 19). We used the Mann Whitney U statistic (Mendenhall, Beaver, & Beaver, 1996) to calculate the area:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2}$$

$$AUC_1 = \frac{U_1}{n_1 n_2}$$

where n_1 is the sample size for sample 1, n_2 is the sample size for sample 2, and R_1 is the sum of the ranks in the sample

Once again, performances of all classifiers were tabulated for ease of comparison.

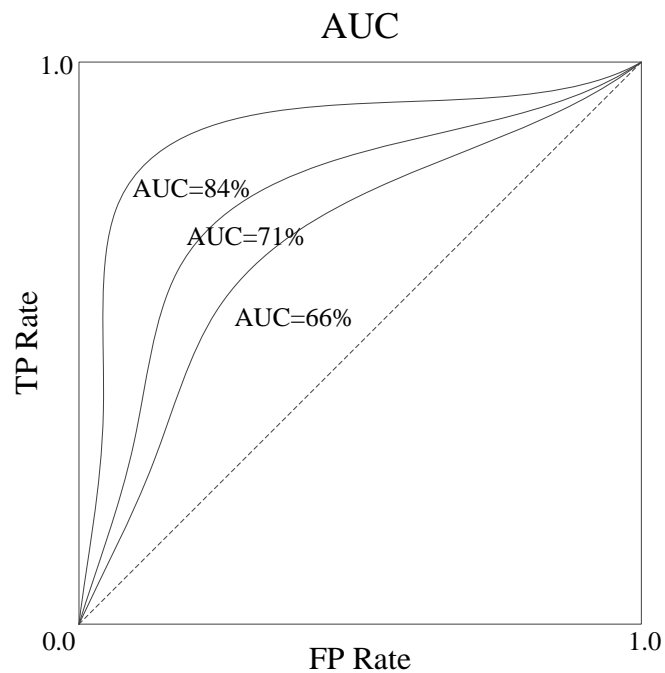


Figure 19. Area Under Curve

Apparatus

All testing was done on a Personal Computer (PC) with a dual Intel Xeon processor and 32 GB of memory. The software used for the evaluation was WEKA (Waikato Environment for Knowledge Analysis) (Witten et al., 2011), an open source machine learning workbench. WEKA has an extensive collection of pre-processing methods and machine learning algorithms implemented in Java as classes with an optional graphical user interface. WEKA Version 3.7.7 was used in this study. Microsoft's SQL Server 2008 R2 and SQL Server Integration Services was also used in the data transformation process.

Data Sets

Bank Data

The first data set used in our experiments was obtained from the UCI repository (Frank & Assuncion, 2010). The data set consists of 10578 records in which none of them is blank. The data set was captured from a direct marketing campaign conducted by a Portuguese banking institution (Moro, Cortez, & Rita, 2011). The main purpose of the campaign was to identify if clients would or would not subscribe to a new bank term deposit. The data variables (Table 12) fall into three different groups:

1. Demographic Data (age, job, marital, education)
2. Financial Information (default, balance, housing, loan)
3. Previous Contact Information (contact, day, month, duration, campaign, pdays, previous, poutcome)

The classification goal using this particular data set is to predict if the client will subscribe to a term deposit (variable y) based on the provided information.

Table 12. Bank Data Set Attributes

Variable Name	Type
age	Numeric
job	Categorical
marital	Categorical
education	Categorical
default	Binary
balance	Numeric
housing	Binary
loan	Binary
contact	Categorical
day	Numeric
month	Categorical
duration	Numeric
campaign	Numeric
pdays	Numeric
previous	Numeric
poutcome	Categorical
y – Class Label	Binary

Service Data

The second data set consists of 15417 records on which the vehicle service performed at an automotive dealership. The 15417 records consist of vehicle information (age, mileage, etc.) as well as what service was performed. The provider of the data has asked us to obfuscate the variable names in order to maintain the confidentiality of the customers and types of services at dealership. There is no clear description of what the variable measured except mileage and age. The main goal of the classification was to identify service customers who purchased vehicles within a year after service was performed based on service history.

This particular data set presented us two problems, high dimensionality and data imbalance. The first step taken to reduce the dimensionality of this data set was to select

the top 200 services performed out of thousands available. The second step was to remove records where attributes contained less than 5% data. Further reduction was attained by implementing the feature selection methods in this study. The data was then balanced using a sub-sample method.

Feature Selection

For our comparison analysis, we reduced our pre-processed data sets for each domain by applying 3 filter feature selection methods; Information Gain, CFS, and Relief-F, and a wrapper method in each classification method. In the case of the Service data set we used the Relief-F method to reduce the original data set to 40 attributes, and applied the wrapper methods to the resulting data set. This method is known as the hybrid method.

After the end of the feature selection process, we obtained 7 data sets to be compared and tested for each classification algorithm in each domain:

- Domain_ALL – Data set with all attributes
- DomainName_IG – Data set chosen using the Information Gain method
- Domain_Name_RLF – Data set containing attributes selected by the Relief-F method
- DomainName_CFS – Data set containing attributes selected by the CFS method
- DomainName_J48_WRP – Data set composed of attributes selected by the wrapper method using the J48 classification algorithm
- DomainName_K-NN_WRP - Data set composed of attributes selected by the wrapper method using the K-NN classification algorithm
- DomainName_SVM_WRP - Data set composed of attributes selected by the wrapper method using the SVM classification algorithm

Summary

The main objective of this study was to find out how different feature selection methodologies (e.g. Filters, Wrappers, and Hybrid) affect the performance of different

classification algorithms on a vehicle service data set in the real world. The classification algorithms to be compared were Decision Tree, k Nearest Neighbor (K-NN), and Support Vector Machines (SVM).

All tests were conducted by using the WEKA workbench and the parameter settings described in Appendix C.

Chapter 4

Results

Introduction

This chapter details the results of our experiment across two domains, in which the data sets have been selected by using different feature selection methods. The first section will describe and compare in details the results of the different feature selection methods tested on each data set. The second section will cover the different classification algorithms being compared in this study. The different options available within each method will be described as well as the performance measures utilized.

Bank Data Set

Information Gain

The first feature selection applied to our data was *information gain* which calculates the entropy of features in each class. The result of this analysis is a listing of features ranked by their importance. Table 13 shows the features and their *information gain* scores ranked in descending order of importance.

Table 13. Top 15 Attributes Ranked by Information Gain by Using the UCI Bank Data Set

Rank	Information Gain	Attr. No.	Description
1	0.238109	41	duration
2	0.077362	43	pdays
3	0.074453	48	poutcome=success
4	0.053002	25	contact=unknown
5	0.045529	44	previous
6	0.045483	45	poutcome=unknown
7	0.037748	26	contact=cellular
8	0.03614	1	age
9	0.033403	23	housing
10	0.023489	28	day
11	0.021891	29	month=may
12	0.019476	42	campaign
13	0.018605	22	balance
14	0.015177	33	month=oct
15	0.01471	40	month=sep

Any *information gain* value above zero shows some type of significance. However, in our experiments, we have limited our results to the top 15 ranked features. The results indicate that attribute “duration” has an information gain of 0.238, almost 3 times greater than the next attributes ranked, “pdays” and “poutcome=success”.

Relief-F

Table 14 shows the results of running the Relief-F feature selection method on the Bank’s data set. Once again the features are ranked in descending order based on the metric used. Using the Relief-F method we see that “poutcome=success” ranks the highest with a value of 0.052 while “duration”, which once ranked first using *information gain*, drops to the second. The significance of the fourth attribute “day” drops more than 50% from that of the top three attributes.

Table 14. Top 15 Attributes Ranked by Relief-F by Using the UCI Bank Data Set

Rank		Attr. No.	Description
1	0.05200416	48	poutcome=success
2	0.05059858	41	duration
3	0.04711666	45	poutcome=unknown
4	0.02138873	28	day
5	0.0204481	23	housing
6	0.01680847	32	month=aug
7	0.01274343	46	poutcome=failure
8	0.01219512	29	month=may
9	0.01158064	39	month=apr
10	0.01020042	34	month=nov
11	0.00986954	30	month=jun
12	0.00925506	24	loan
13	0.00899981	33	month=oct
14	0.00897145	38	month=mar
15	0.00890528	40	month=sep

Correlation-based Feature Selection (CFS)

The last filter type feature selection technique used on our Bank data set was Correlation Feature Selection (CFS). This method searches through all combination of the features in the data set and concludes with a subset that includes features which have good predicting capabilities, and yet take redundancy and correlation between the features into account. In our experiment, the number of variables was reduced to 9. The search method used in our testing was “Greedy Stepwise (forwards)” which starts the search with no attributes as it searches forward. The merit of our final subset was 0.161 from a possible value range from 0 to 1.0 with values closer to 0 being better. Table 15 lists the attributes selected by this method.

Table 15. Attributes Selected by CFS by Using the UCI Bank Data Set

Attribute
duration
balance
loan
contact=unknown
day
month=oct
month=mar
age
poutcome=success

By observing the table we can see that attributes “duration”, “day”, “month=oct”, and “poutcome=success” also had high rankings in both feature selection methods, Information Gain and Relief-F. The rest of the attributes selected by this method were also ranked by the previous two methods.

Wrapper

The final feature selection method we applied to the data set was the wrapper method. In this method, we applied feature reduction to the data set by using the classifier as part of the selection process. Table 16 shows the results which were generated by applying this feature selection method to each of our three classification methods in this experiment.

Table 16. Results after Applying the Wrapper Selection by Using the UCI Bank Data Set

1-K-NN	Decision Tree	SVM
duration	duration	duration
contact=unknown	contact=unknown	contact=unknown
poutcome=success	poutcome=success	poutcome=success
age	Age	housing
day	Day	month=jan
pdays	Pdays	month=mar
month=jun	Balance	campaign
month=jul	Housing	
month=aug	job=unknown	
month=nov	marital=divorced	
month=dec	month=may	
month=jan	month=jun	
month=feb	month=jul	
month=apr	month=aug	
previous	month=oct	
month=may	month=feb	
	month=mar	
	month=apr	
	month=sep	
	education=primary	
	Campaign	

We can see that attributes “duration”, “poutcome=success”, “contact=unknown”, and “age” continue to show significance in the classification process.

In the following section, we will run our three classification algorithms on the data sets built by our different feature selection methods. We will then compare their performance using different evaluation metrics.

Results

Decision Tree

We tested the decision tree classifier by using WEKA's J48 algorithm with all the features in our original data set as well as those selected by the Information Gain, Relief-F, CFS, and wrapper methods. The first run was done by using the default settings in WEKA, which include a minimum of two instances per leaf and a confidence factor of 0.25 (Table 17).

Table 17. Performance of the Decision Tree Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics

Data Set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
J48ALL	0.847	0.862	0.847	0.821	0.873
J48IG	0.833	0.866	0.833	0.803	0.863
J48RLF	0.858	0.901	0.858	0.834	0.882
J48CFS	0.821	0.882	0.821	0.802	0.840
J48WRP	0.862	0.899	0.862	0.839	0.886

J48ALL – Using all features

J48IG – Features selected by using the information gain method

J48RLF – Features selected by using the Relief-F method

J48CFS – Features selected by using correlation based feature method

J48WRP – Features selected by using the wrapper method

We can see that reducing the feature set on our data set improved the accuracy and F-Measure scores over that of using all the attributes in 2 out of 4 data sets. However, the AUC rate increased in all cases where the data set was reduced. By implementing the wrapper method we were able to increase the accuracy and F-Measure from 84.7% to 86.2% and the AUC from 86.2% to 89.9%. With a 10-fold cross validation accuracy of 86.2% the wrapper method produced the highest accuracy, F-Measure, sensitivity and specificity scores amongst our tests. The data set that produced the highest AUC was that

one selected using the Relief-F method which produced an AUC of 90.1%. The performance measures were graphed for easier visualization (Figure 20).

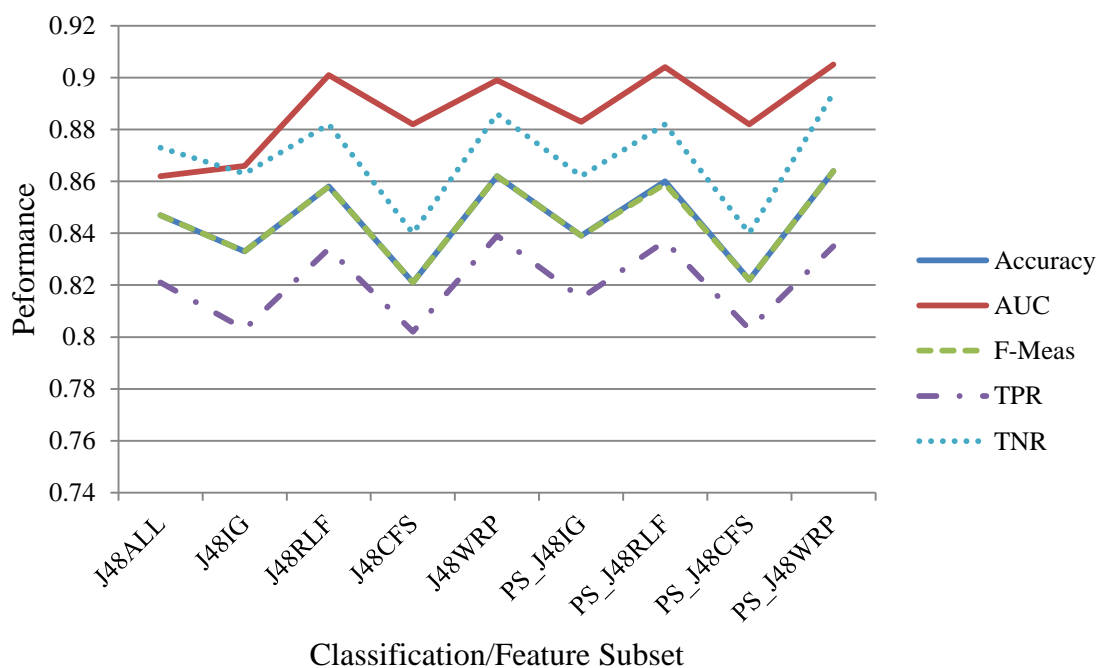


Figure 20. Performance of J48 Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

Our experiments showed that by using the wrapper feature selection method, the confusion matrix for the J48 classification produced the highest accuracy rate. The result is shown in Table 18. The performance measures are shown in Table 19.

Table 18. Confusion Matrix for Decision Tree Algorithm Using J48 Wrapper Data Set

No	Yes	<- Classified As
4437(TP)	852(FN)	no
605(FP)	4684(TN)	yes

Table 19. Performance Measures for Decision Tree Algorithm Using J48 Wrapper Data Set

Accuracy	0.862
Precision*	0.863
Recall*	0.862
F-Measure*	0.862
TP Rate	0.886
TN Rate	0.839

* Weighted average

Using the values in the confusion matrix we can calculate our accuracy by summing the correct predictions (4437+4684) and dividing it by our total number of observations (4437+852+605+4684) which gives us an accuracy of .862. The precision rate for our 1st class can be calculated by dividing our true positives (4437) by the sum of all observations predicted as positive (4437+605) 5042, which results in a precision rate of 88.0%. Our recall can be calculated by dividing our true positives 4437 by the sum of true positives and false negatives (4437+852) 5289, which yields a recall of 83.9% for our first class. Once we have the recall and precision, we can calculate the F-measure by using the following formula:

$$f\text{-measure} = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}}$$

By substituting the values in the above formula we get an F-measure of 0.866.

From here on we will use the values calculated by the WEKA framework for our measurements.

Next, the J48 classifier was run using the wrapper data set while the confidence interval parameter varies from 0.1 to 0.5 in increments of 0.1. Lowering the confidence factor decreases the amount of post-pruning performed by the algorithm. The results are reflected in Table 20.

Table 20. Performance of the J48 Classifier across the UCI Bank Data Set in Terms of Accuracy AUC, TP Rate, and FP Rate Evaluation Statistics for Different Confidence Factors

Confidence Factor	Accuracy	AUC	No. of Leaves	TP Rate	TN Rate
0.1	0.864	0.905	120	0.895	0.834
0.2	0.863	0.901	162	0.877	0.840
0.3	0.862	0.899	229	0.883	0.841
0.4	0.860	0.895	265	0.877	0.842
0.5	0.857	0.891	352	0.869	0.846

The effects of our post pruning process on the decision tree are shown on Figure 21.

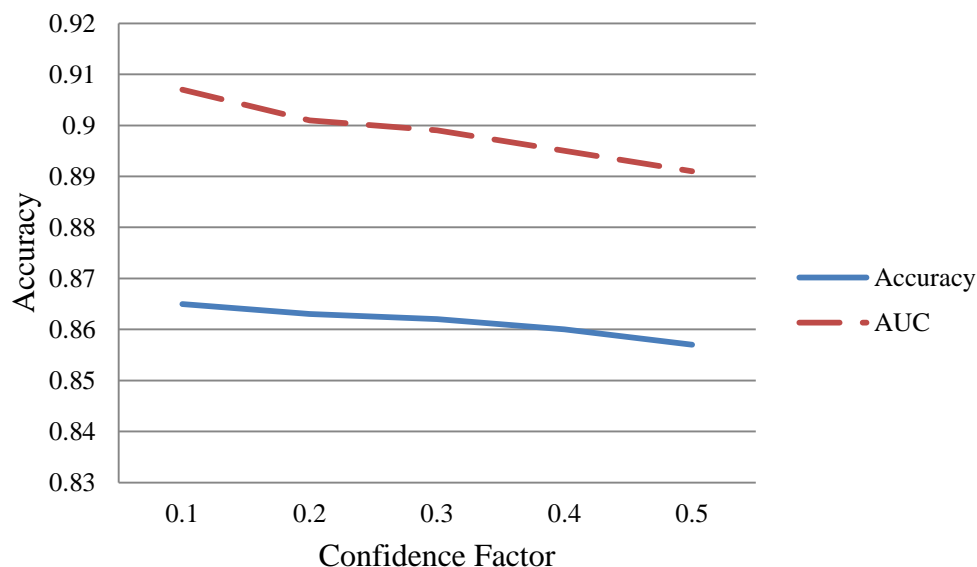


Figure 21. Decision Tree Pruning. Lower Confidence Factor Indicates Higher Pruning

The best result was found by using the features selected by the wrapper method along with a confidence factor of 0.1. This combination produced an accuracy of 86.4%.

However, an AUC rate of 90.5% was attained when using the wrapper attributes and a confidence interval of 0.1 as shown in Table 21. The highest True Positive rate (83.7 %) was achieved using attributes selected by the Relief method.

Table 21. Performance of the Decision Tree Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics Using a Parameter Search

Data Set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
PS_J48_IG	0.839	0.883	0.839	0.815	0.862
PS_J48RLF	0.86	0.904	0.859	0.837	0.882
PS_J48CFS	0.822	0.882	0.822	0.803	0.840
PS_J48WRP	0.864	0.905	0.864	0.835	0.894

The confusion matrix for the decision tree algorithm by using the wrapper data set is shown in Table 22 and the performance measures achieved in Table 23.

Table 22. Confusion Matrix for Decision Tree with Optimized Parameter Wrapper Data Set

No	Yes	<- Classified As
4414(TP)	875(FN)	no
560(FP)	4729(TN)	yes

Table 23. Performance Measures for Decision Tree with Optimized Parameter Wrapper Data Set

Accuracy	0.864
Precision	0.866
Recall	0.864
F-Measure	0.864
TP Rate	0.894
TN Rate	0.835

K-Nearest Neighbor Classifier (K-NN)

For the K-NN algorithm we performed our tests using a k value of 1, 5, and 10, as well as with a parameter search with k values ranging from 1 to 10 by increments of 1. Results are shown in Table 24.

Table 24. Performance of the K-NN Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics

Data Set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
1NNALL	0.735	0.801	0.735	0.787	0.684
1NNIG	0.776	0.775	0.776	0.792	0.759
1NNRLF	0.807	0.809	0.807	0.810	0.804
1NNCFS	0.761	0.758	0.761	0.767	0.755
1NNWRP	0.796	0.794	0.796	0.806	0.787
5NNALL	0.735	0.801	0.735	0.787	0.684
5NNIG	0.809	0.87	0.809	0.835	0.782
5NNRLF	0.842	0.898	0.842	0.839	0.845
5NNCFS	0.809	0.867	0.809	0.808	0.810
5NNWRP	0.849	0.905	0.849	0.832	0.866
10NNALL	0.734	0.812	0.730	0.852	0.615
10NNIG	0.798	0.879	0.797	0.867	0.729
10NNRLF	0.842	0.898	0.842	0.839	0.845
10NNCFS	0.811	0.884	0.811	0.840	0.782
10NNWRP	0.830	0.900	0.830	0.859	0.801
PS_NNALL	0.74	0.810	0.730	0.851	0.610
PS_NNIG	0.806	0.875	0.806	0.836	0.77
PS_NNRLF	0.841	0.901	0.841	0.838	0.845
PS_NNCFS	0.815	0.882	0.815	0.815	0.815
PS_NNWRP	0.835	0.889	0.835	0.830	0.841

After running the tests we saw that using the feature selection wrapper method and a k value of 5 resulted in an accuracy and F-Measure of 84.9% and an AUC of 90.5%. This set also produced the highest F-Measure and True Negative Rate (TNR). The worst performer was using all the attributes with 1 nearest neighbors. The results of the tests are shown in Figure 22.

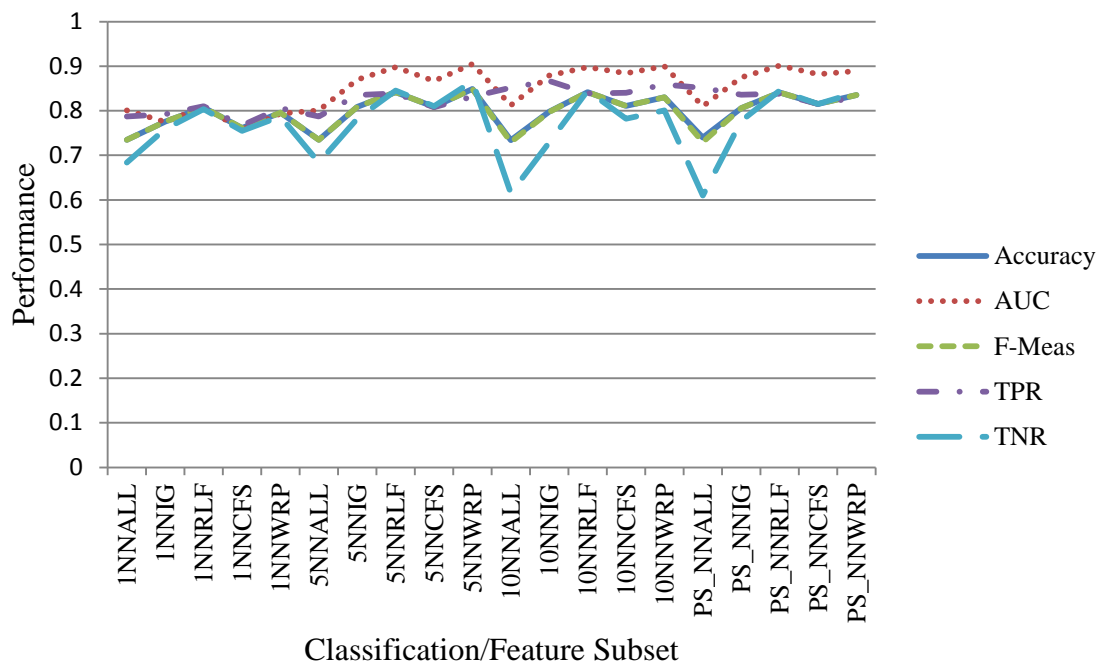


Figure 22. Performance of K-NN Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

The confusion matrix for the best performer is shown in Table 25 and performance measures on Table 26.

Table 25. Confusion Matrix for Nearest Neighbor Using $k = 5$

No	Yes	<- Classified As
4399 (TP)	890(FN)	no
711 (FP)	4578 (TN)	yes

Table 26. Performance Measures for Nearest Neighbor Using $k = 5$

Accuracy	0.849
Precision	0.849
Recall	0.849
F-Measure	0.849
TP Rate	0.866
TN Rate	0.832

We produced an accuracy rate of 84.9% by using the parameter search with k values ranging from 1 to 10, and AUC rate of 90.5% by using attributes selected by Relief-F method of a k value of 5.

Support Vector Machine (SVM) classification results

The last classification algorithm to be tested was the Support Vector Machine (SVM) using the Lib-SVM algorithm (Chang & Lin, 2011). Once again, we ran the algorithm on each data set. First, we used the default parameters in WEKA. Secondly, we used a grid search to optimize the cost of error (C) and gamma parameters. Results of the tests are shown in Table 27.

Table 27. Performance of the Lib-SVM Classifier across the UCI Bank Data Set in Terms of Accuracy, AUC, F-Measure, True Positive Rate, and True Negative Rate Evaluation Statistics

Data Set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
LibSVMALL	0.785	0.785	0.784	0.860	0.71
LibSVMIG	0.814	0.814	0.813	0.872	0.756
LibSVMRLF	0.819	0.819	0.819	0.863	0.775
LibSVMCFS	0.785	0.785	0.783	0.890	0.680
LibSVMWRP	0.818	0.818	0.818	0.854	0.782
GS_LibSVMIG	0.814	0.814	0.813	0.872	0.756
GS_LibSVMRLF	0.844	0.844	0.844	0.815	0.872
GS_LibSVMCFS	0.841	0.841	0.841	0.836	0.847
GS_LibSVMWRP	0.838	0.838	0.838	0.845	0.831

The best accuracy of 84.4% was obtained when the grid search parameter, cost of error (C), was set to 4 and gamma (γ) set to 1. Efficiency of the classifier was determined by comparing the predicted and expected class labels of the data set using 10 fold cross validation. True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) values were 4311, 4613, 676, and 978 respectively. They produced a sensitivity rate of 0.815 and specificity rate of 0.872, and an Accuracy rate of 0.844. Interestingly, the AUC and F-Measure measures on this particular run were identical to the accuracy of 0.844. Figure 23 illustrates the final results.

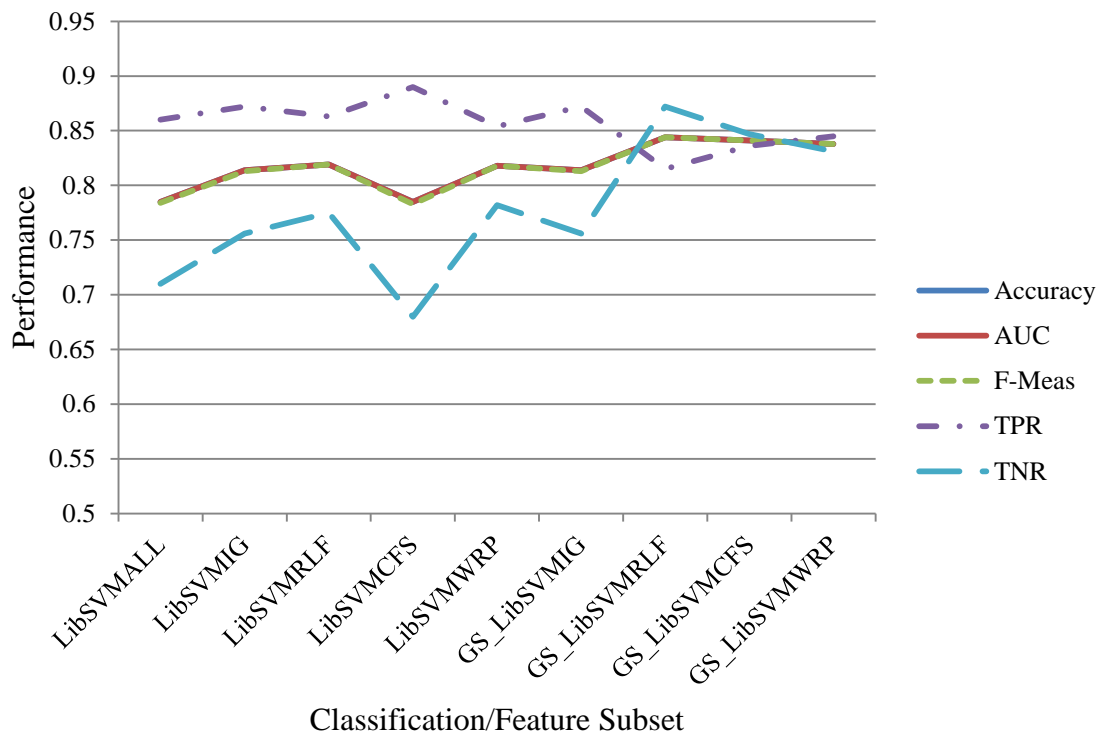


Figure 23. Performance of SVM Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

Analyzing the best accuracy for each classifier we concluded in Table 28 that the J48 decision tree model produced a 10-fold cross validation accuracy of 89.3%, followed by the Support Vector Machine with a 10-fold cross validation accuracy of 85.0%, and K-NN with a 10-fold cross validation accuracy of 84.9%. In the next section we will use WEKA's experimenter to determine if the decision tree classifier is significantly better than the K-NN and support vector machine classifiers in this domain.

Table 28. Average Overall Classification Accuracy on Bank Data Set Based on Individual Runs

	K-NN	Decision Tree	SVM
IG	0.810	0.840	0.818
CFS	0.815	0.822	0.815
RLF	0.842	0.860	0.850
WRP	0.849	0.893	0.838

Experimenter to compare

Table 29 shows the results of the WEKA experimenter with the three classifiers selected. The experiment was run by using multiple data sets across all classifiers. By using the same seeding we can evaluate the performance of each classifier on common grounds. The percentage of good results for each of the 3 classifiers is shown in the dataset rows. For example, by using the CFS data set, the result of J48/Decision Tree is 82.03%, K-NN is 80.91%, and Lib-SVM is 80.86%. Each algorithm was compared against the base algorithm, Decision Tree, in this case. If the performance of the classifier being tested was statistically higher than the base classifier, then it was tagged with a (v). If the annotation is a (*) the classifiers, the result was worse than the base classifier. No tags signified if the classifier performed neither worse or better than the base classifier. All our tests were performed by using a corrected two-tailed t-test (Nadeau and Bengio, 2000) with a significance level of 0.05

Our best classification performance with accuracy as a measure was achieved by using the Decision Tree algorithm and the J48 Wrapper data set with an 86.33% accuracy rate. The worse performing classifier was SVM with the CFS data set, which achieved an 80.86% accuracy rate. All our tests indicate that K-NN and SVM performed statistically worse across all data sets.

Table 29. Accuracy Results with Optimized Parameters by Using Experimenter

Dataset	Decision Tree	K-NN	SVM
CFS	82.03	80.91 *	80.86 *
IG	83.41	80.89 *	82.49 *
RLF	85.62	84.23 *	83.52 *
SVM Wrapper	83.21	81.14 *	82.15 *
J48 Wrapper	86.33	83.14 *	83.89 *
K-NN Wrapper	84.84	83.52 *	83.00 *
	(v/ /*)	(0/0/6)	(0/0/6)

The last row in Table 29 shows how many times the classifier was better, same, or worse (x, y, z) than the base classifier.

The final numbers are based on 1800 results. A 10 fold cross validation was run 10 times across all 3 classifiers using 6 data sets.

The same test was run again but the performance measure parameter was changed to AUC. The results of this run are shown in Table 30.

Table 30. AUC Results Using Optimized Parameters by Using Experimenter

Dataset	Decision Tree	K-NN	SVM
CFS	0.88	0.87 *	0.81 *
IG	0.89	0.87 *	0.82 *
RLF	0.91	0.90 *	0.84 *
SVM Wrapper	0.89	0.87 *	0.82 *
J48 Wrapper	0.91	0.89 *	0.84 *
K-NN Wrapper	0.90	0.89 *	0.83 *
	(v/ /*)	(0/0/6)	(0/0/6)

Once again we saw that the decision tree classifier produced the highest AUC score in the Relief-F and wrapper data sets. The resulting AUC curves generated are shown in Figure 24.

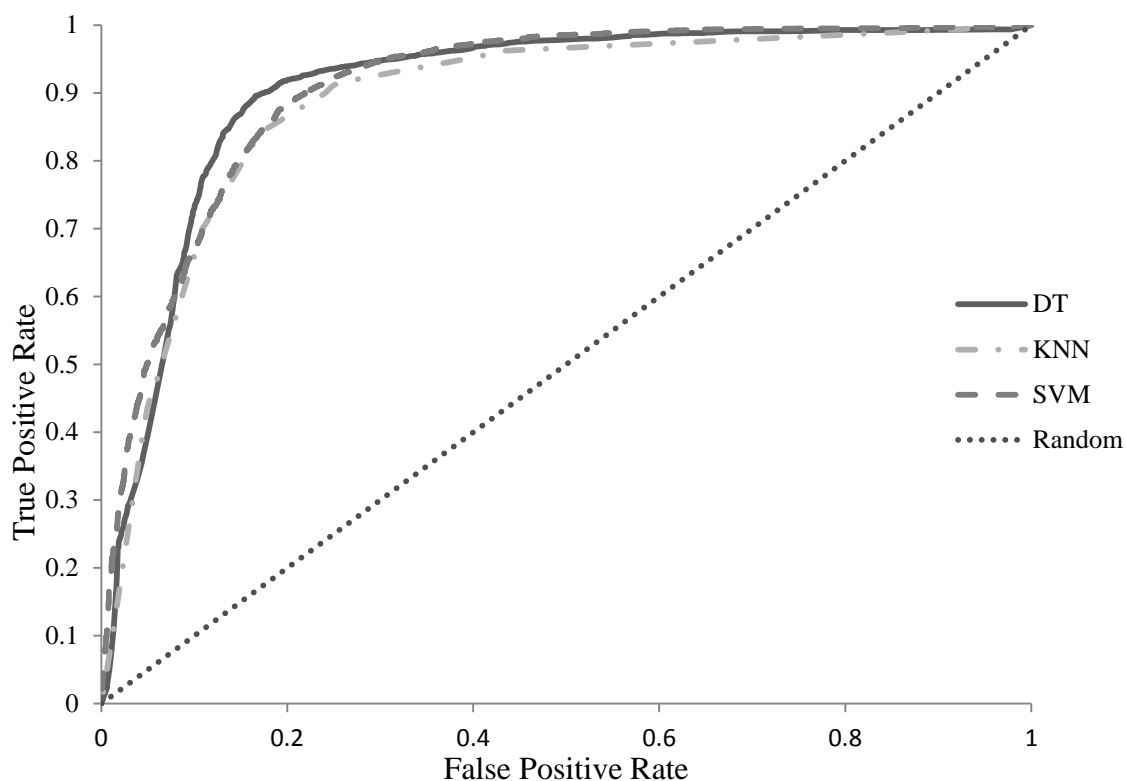


Figure 24. ROC Curve for Bank Data Set

The final measure to be tested was the F-Measure. The Decision Tree with attributes selected by the wrapper method produced the highest score of 0.84 as shown in Table 31 and graphed in Figure 25.

Table 31. F-Measures Results for Bank Data Set Using Optimized Parameters by Using Experimenter

Dataset	Decision Tree	K-NN	SVM
CFS	0.83	0.73 *	0.82 *
IG	0.83	0.72 *	0.82 *
RLF	0.82	0.82	0.82 *
SVM Wrapper	0.83	0.83	0.83 *
J48 Wrapper	0.84	0.84 *	0.84 *
K-NN Wrapper	0.83	0.83	0.83 *
	(v/ *)	(0/3/3)	(0/0/6)

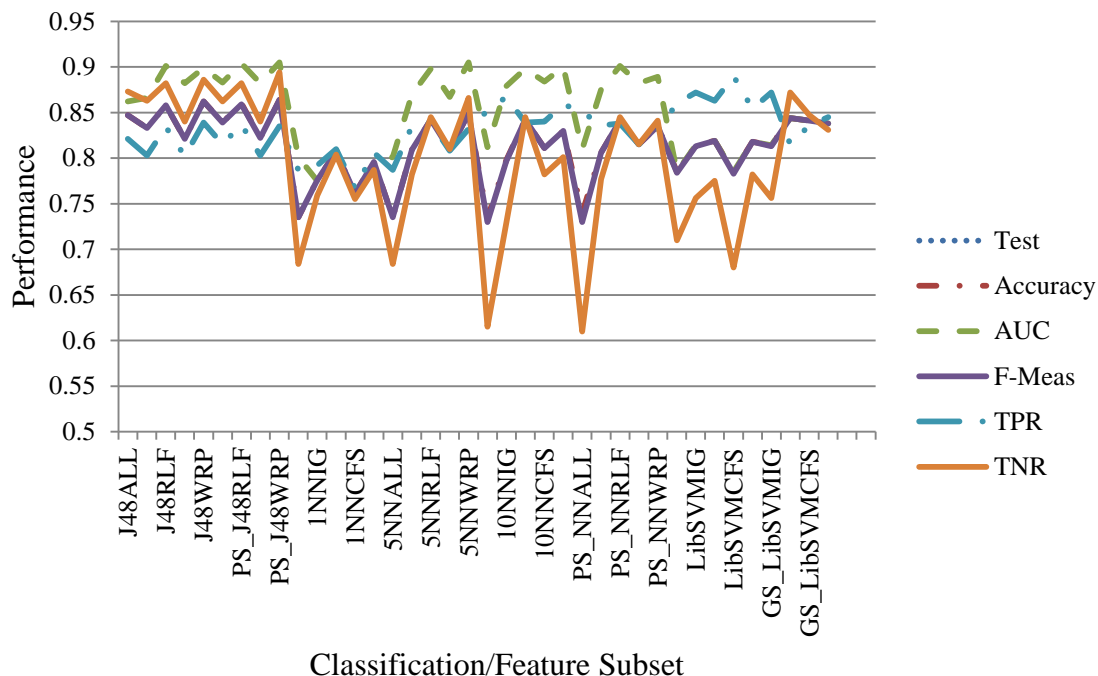


Figure 25. Classification Performance on Bank Data across Data Sets

Comparison and Analysis

The results indicated that the accuracy performance of the three classifiers was greatly improved after applying feature reduction to our data set. This was especially true in the case of the K-NN classifiers. In the case of the decision tree classifier, we saw that reducing the feature set did not always perform better than by using all the features. This is because the decision tree algorithm itself performs some type of data reduction in the building process. The high percentage in sensitivity and specificity on all results were also encouraging of good classification.

Service Data Set

Information Gain

The top 20 ranked variables with the Information Gain feature selection method are shown on Table 32. Please note that most of the variable names in the data set have been obfuscated in order to protect the privacy of the original data owner.

Table 32. Attributes Ranked Using Information Gain Method across the Service Data Set

Score	Attribute No.	Attribute
0.0759965874	20	39CTZ22
0.0712005663	21	76CTZ01
0.0592927439	22	76CTZ
0.0388523704	2	Mileage
0.0380071942	46	76CTZDLRTRANS
0.0302880702	57	76CTZETCH
0.0263680886	62	39CVZ21
0.0256478072	72	76CVZ02
0.0244553597	58	39CTZ21
0.0237713931	69	76CTZ02
0.0237075594	1	Age
0.0201856411	4	46CTZ
0.0178737538	67	38CTZ1
0.0175826467	65	39CVZ22
0.0169490617	66	76CVZ01
0.0160997116	71	76CVZ
0.01604993	88	01CTZ01
0.0157341061	28	25CTZ
0.0153268992	97	01CVZ
0.0146639142	99	76CVZETCH

By using the *information gain* we reduced the original 200 attributes down to 20. We saw that attributes “39TZ22”, “76CTZ01”, and “76CTZ” scored more than twice than the rest of the features.

Relief-F

The results of running the Relief-F feature reduction method on the Service data set are shown in Table 33. Once again we have kept the top 20 ranked attributes in this data set.

Table 33. Attributes Ranked by the Relief-F Method across the Service Data Set

Rank	Attribute No.	Attribute name
0.0265616	20	39CTZ22
0.0243757	46	76CTZDLRTRANS
0.0230849	21	76CTZ01
0.0192515	62	39CVZ21
0.0186158	72	76CVZ02
0.0182072	22	76CTZ
0.0169359	57	76CTZETCH
0.0158007	4	46CTZ
0.0146073	58	39CTZ21
0.0132646	69	76CTZ02
0.0110398	97	01CVZ
0.0105792	88	01CTZ01
0.0102614	19	25
0.0099565	65	39CVZ22
0.009269	7	39
0.0092236	66	76CVZ01
0.0083025	67	38CTZ1
0.0082701	71	76CVZ
0.007589	124	01CVZ01
0.0059934	87	39CVZ

We can see that some of the attributes ranked “39TZ22”, “76CTZDLRTRANS”, and “76CTZ01” were also highly ranked by the Information Gain method. However we see that attributes “Mileage” and “Age” were not.

Correlation-based Feature Selection (CFS)

The last filter type feature selection method applied to the Service data set was CFS (Table 34).

Table 34. Attributes Selected by CFS Method by Using the Service Data Set

Attribute Name
Mileage
39
39CTZ22
76CTZ01
76CTZ
76CTZDLRTRANS
76CTZETCH
39CTZ21
39CVZ21
39CVZ22
38CTZ1
76CTZ02
76CVZ02
76CVZETCH

Our data set was reduced from 200 attributes to 14 using this method. Once again, we saw that most attributes selected with the CFS method when using the Information Gain and Relief-F methods.

Wrapper

Table 35 shows the attributes selected when the wrapper type feature selection method was used on the Service data set. The wrapper method was run 3 times, one time for each classification algorithm used in this study, K-NN with $k = 1$, Decision Tree / J48, and Support Vector Machine classification algorithms.

Table 35. Features Selected by Wrapper Selection by Using Service Data Set

1-K-NN	Decision Tree	SVM
39CTZ22	39CTZ22	39CTZ22
Mileage	Mileage	Mileage
76CTZDLRTRANS	76CTZDLRTRANS	76CTZ01
76CTZ01	39CVZ21	39CVZ21
Age	76CVZ02	76CVZ02
39CVZ21	76CTZETCH	76CTZETCH
76CVZ02	39CTZ21	39CTZ21
76CTZETCH	76CTZ02	76CTZ02
39CTZ21	39	01CTZ01
76CTZ02	76CVZ01	39
39	25	76CVZ01
76CVZ01	39CVZ22	25
76CVZ	29	76CVZ
46CVZ	38CTZ1	10
39CVZ	46CVZ	46CVZ
39CTZ	76CVZETCH	76CVZETCH
PDI	76CTZ1	39CVZ
	39CVZ	39CTZ
	38CVZ1	PDI
	76CVZDLRTRANS	
	39CTZ	
	76CVZ1	
	PDI	

Results

Decision Tree

Table 36 shows the results of running the decision tree algorithm against our Service data sets.

Table 36. Performance of the Decision Tree Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics

Data set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
J48ALL	0.798	0.778	0.787	0.966	0.575
J48IG	0.778	0.750	0.765	.954	0.546
J48RLF	0.763	0.723	0.743	.977	0.480
J48CFS	0.779	0.749	0.765	0.964	0.534
J48WRP	0.797	0.778	0.797	0.964	0.577
PS_J48IG	0.782	0.759	0.769	0.963	0.542
PS_J48RLF	0.762	0.723	0.743	0.976	0.480
PS_J48CFS	0.779	0.749	0.765	0.963	0.534
PS_J48WRP	0.797	0.778	0.787	0.965	0.576

Based on the results in Table 36 we concluded that the decision tree algorithm had the worst performance of 76.2%, with the Relief-F feature reduction method. The best accuracy (79.8 %) and the best AUC (77.8%) were achieved by using no feature reduction at all. A parameter search determined that a confidence factor of 0.1 would achieve the highest accuracy of 79.7 % slightly less than by using no attribute selection. The classifier performed equally when information gain, CFS, and Relief-F feature selection methods were used. The most important observation in this group of tests is the below average values of the True Negative Rate. The highest, 57.6 percentage, was achieved using the wrapper method. Figure 26 shows the performance across the different data sets used in this experiment.

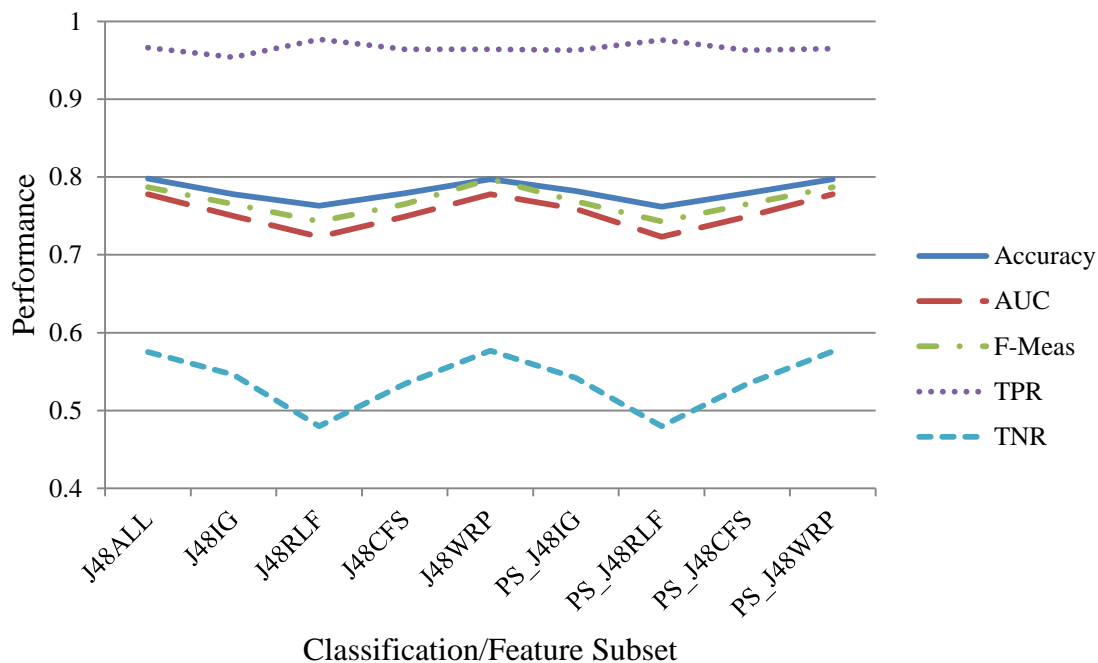


Fig. 26. Performance of J48 Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

The confusion matrix produced by the best classification is shown in Table 37 and performance measures in Table 38.

Table 37. Decision Tree Confusion Matrix

Did Not Buy	Bought	<- Classified As
8483 (TP)	298(FN)	Did Not Buy
2820(FP)	3816(TN)	Bought

Table 38. Decision Tree Performance Measures

Accuracy	0.798
Precision	0.827
Recall	0.966
F-Measure	0.787
TP Rate	0.966
TN Rate	0.576

K-NN

The results of running the K-NN classifier on our data sets are shown in Table 39. A 10-fold average accuracy of 79.5%, F-Measure of 78.3%, and an AUC of 79.6%, the highest in our experiment, was obtained when running the classifier with the wrapper data set and a parameter search which selected the optimal performance at $k = 9$. We also noted that using the CFS feature selection method performed worse than using no reduction only when k was equal to 1. Otherwise, we saw that all feature reduction sets performed better than the data set which was not reduced.

Table 39. Performance of the K-NN Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics

Data set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
1NNALL	0.702	0.696	0.701	0.759	0.628
1NNIG	0.683	0.674	0.683	0.722	0.631
1NNRLF	0.763	0.740	0.742	0.979	0.476
1NNCFS	0.693	0.689	0.692	0.741	0.629
1NNWRP	0.795	0.796	0.783	0.970	0.564
5NNALL	0.749	0.775	0.741	0.874	0.583
5NNIG	0.741	0.757	0.735	0.857	0.587
5NNRLF	0.761	0.740	0.741	0.978	0.475
5NNCFS	0.744	0.760	0.738	0.858	0.592
5NNWRP	0.791	0.795	0.778	0.970	0.553
10NNALL	0.762	0.786	0.749	0.933	0.535
10NNIG	0.771	0.764	0.759	0.940	0.548
10NNRLF	0.759	0.740	0.738	0.976	0.472
10NNCFS	0.772	0.768	0.761	0.936	0.556
10NNWRP	0.787	0.794	0.773	0.971	0.543
PS_NNALL	0.762	0.785	0.749	0.933	0.535
PS_NNIG	0.771	0.764	0.759	0.940	0.548
PS_NNRLF	0.762	0.740	0.742	0.979	0.476
PS_NNCFS	0.772	0.768	0.761	0.936	0.556
PS_NNWRP	0.795	0.796	0.783	0.970	0.564

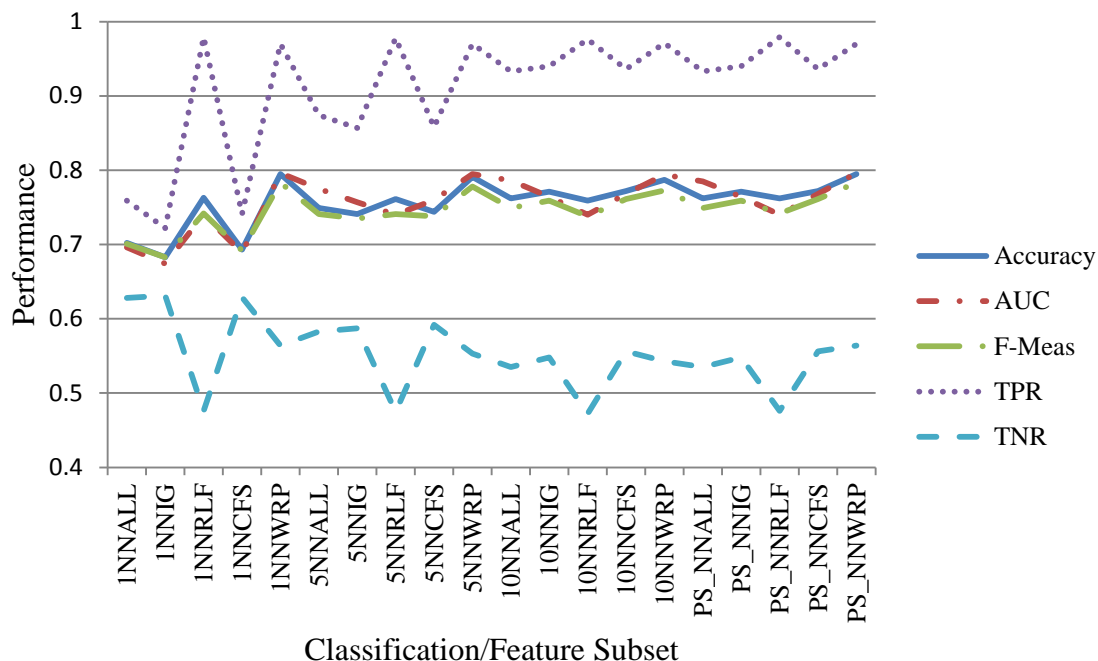


Figure 27. Performance of K-NN Classifier across Different Feature Sets by Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

The confusion matrix for the best performing K-NN classifier is shown in Table 40 and performance measures in Table 41.

Table 40. Confusion Matrix for Nearest Neighbor using $k = 1$ and Wrapper Data Set

Did Not Buy	Bought	<- Classified As
8515 (TP)	266(FN)	Did Not Buy
2893(FP)	3743 (TN)	Bought

Table 41. Performance Measures for Nearest Neighbor using $k = 1$ and Wrapper Data Set

Accuracy	0.795
Precision	0.827
Recall	0.970
F-Measure	0.783
TP Rate	0.970
FP Rate	0.564

A note of observation is that the classifier/subset combination with the highest accuracy did not have the highest specificity (TN Rate) of all tests. The highest specificity of 63.1 % was achieved when using attributes selected by the information gain method and $k = 1$.

Lib-SVM

Our last classifier to be tested was the Support Vector Machine. The algorithm was first run with the default cost of error value of 1.0 and default gamma value of 0.

Table 42. Performance of the LIB-SVM Classifier across the Service Data Set in Terms of Accuracy, AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics with Default Parameters

Data set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
LibSVMALL	0.766	0.733	0.748	0.973	0.493
LibSVMIG	0.764	0.730	0.745	0.972	0.489
LibSVMRLF	0.763	0.729	0.743	0.977	0.480
LibSVMCFS	0.761	0.727	0.742	0.974	0.480
LibSVMWRP	0.793	0.765	0.781	0.964	0.567

As we can see in Table 42, running the Support Vector Machine classifier on the data set selected by the wrapper method produced the best average accuracy performance

(79.3%), best AUC (76.5%), and best F-Measure (78.1%). Data sets built with the Relief-F, information gain, and CFS methods performed worse than on the original data set with all attributes across all measures.

The same tests were then repeated using a grid search to obtain optimum values for cost of error and gamma. These results are shown in Table 43.

Table 43. Performance of the Lib-SVM Classifier across the Service Data Set in Terms of Accuracy AUC, F-Measure, TP Rate, and TN Rate Evaluation Statistics Grid Search Parameters

Data set	Accuracy	AUC	F-Measure	TP Rate	TN Rate
GS_LibSVMALL	0.762	0.726	0.745	0.971	0.490
GS_LibSVMIG	0.764	0.730	0.743	0.972	0.489
GS_LibSVMRLF	0.763	0.729	0.743	0.977	0.480
GS_LibSVMCFS	0.759	0.724	0.739	0.976	0.473
GS_LibSVMWRP	0.798	0.771	0.787	0.965	0.576

The results showed, once again, that using the attributes selected by the wrapper method produced the best results. By using the grid search, which obtained a cost of error value of 1 and gamma of 1, we were able to increase the accuracy performance of the Support Vector Machine from 79.3% to 79.8%, the AUC from 76.5% to 77.1%, and the F-Measure from 78.1% to 78.7%. Using these parameters the information gain and Relief-F data sets performed better than with no reduction at all when using the accuracy and AUC measures. On the other hand, if measured using the F-Measure, information gain, Relief-F, and CFS produced lesser results than not using any feature reduction at all on the data set. The performance measure of each data set is graphed on Figure 28.

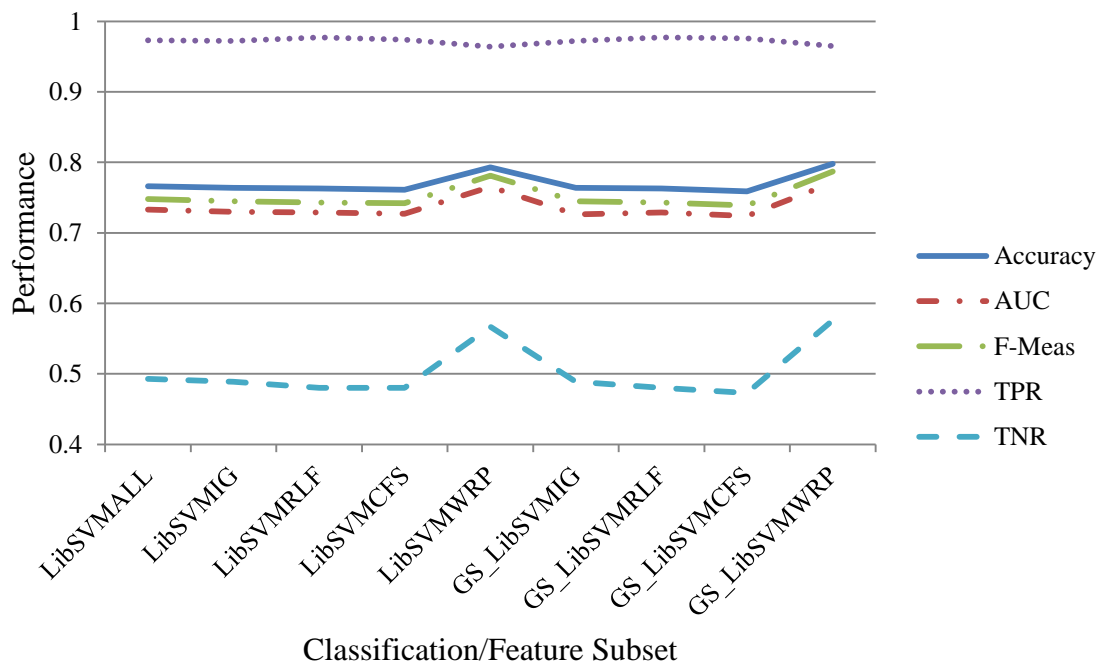


Figure 28. Performance of SVM Classifier across Different Feature Sets Using the Service Data Set in Terms of Accuracy, AUC, F-Measure, TPR, and TNR Evaluation Statistics

By comparing the results of our individual tests (Table 44), it showed that using the SVM algorithm with the wrapper data set produced the highest accuracy of 79.8%. The worse performer was K-NN using the information gain data set which resulted in an accuracy rate of 77.1%. In the next section we will run the classification algorithms on these data sets using WEKA's experimenter module.

Table 44. Average Overall Classification Accuracy on Service Data Set across All Classifiers and Feature Selection Methods through Individual Testing

	K-NN	Decision Tree	SVM
IG	0.771	0.782	0.764
CFS	0.772	0.779	0.761
RLF	0.762	0.763	0.763
WRP	0.795	0.797	0.798

Experimenter

Table 45. Average Overall Classification Accuracy by Using the Service Data Set across All Classifiers and Feature Selection Methods Using Optimized Parameters

Dataset	Decision Tree	K-NN	SVM
CFS	74.94(1.01)	71.43(1.08) *	72.74(0.95) *
IG	77.79(0.86)	74.14(1.04) *	76.42(0.82) *
RLF	76.22(0.83)	76.13(0.82)	76.32(0.84) v
SVM Wrapper	77.94(0.80)	77.67(0.84) *	77.95(0.79)
J48 Wrapper	79.77(0.82)	79.03(0.80) *	79.74(0.82)
K-NN Wrapper	77.50(0.82)	77.03(0.83) *	77.48(0.80)
	(v/ /*)	(0/1/5)	(1/3/2)

The schemes used in the experiment are shown in Table 45. The classification algorithms compared are shown in the columns and the data sets used are shown in the rows. The percentage correct for each of the 3 schemes is shown in each dataset row: 74.94% for Decision Tree, 71.43% for K-NN, and 72.74% for SVM using the CFS data set. Once again, the annotation “v” or “*” indicates that a specific result is statistically better (v) or worse (*) than the baseline scheme at the significance level of 0.05 (user defined). In the first result set, we saw that the K-NN and SVM algorithms performed statistically worse than the Decision Tree algorithm. The Decision Tree classification only performed worse than SVM when using the data set created by the RLF feature selection method. When running the SVM, J48, and K-NN wrapper data sets Decision Trees and SVM obtained statistically similar results. Our highest accuracy rate of 79.77% was obtained when running the Decision Tree scheme with the attributes selected by its own method. Using SVM on the same data set produced an accuracy of 79.74% which was not statistically different.

Table 46 shows the results of the experimenter using the same data sets but measuring performance using the AUC measure. Here, K-NN produced the best result with a performance of 80%.

Table 46. Average Overall AUC by Using Service Data Set across All Classifiers and Feature Selection Methods Using Optimized Parameters

Dataset	Decision Tree	K-NN	SVM
CFS	0.75(0.01)	0.76(0.01) v	0.73(0.01) *
IG	0.76(0.01)	0.76(0.01)	0.73(0.01) *
RLF	0.73(0.01)	0.75(0.01) v	0.73(0.01)
SVM Wrapper	0.76(0.01)	0.78(0.01) v	0.75(0.01) *
J48 Wrapper	0.78(0.01)	0.80 (0.01) v	0.77(0.01) *
K-NN Wrapper	0.75(0.01)	0.78(0.01) v	0.74(0.01) *
	(v/ /*)	(5/1/0)	(0/1/5)

The ROC curves for each classification are compared in the Figure 29.

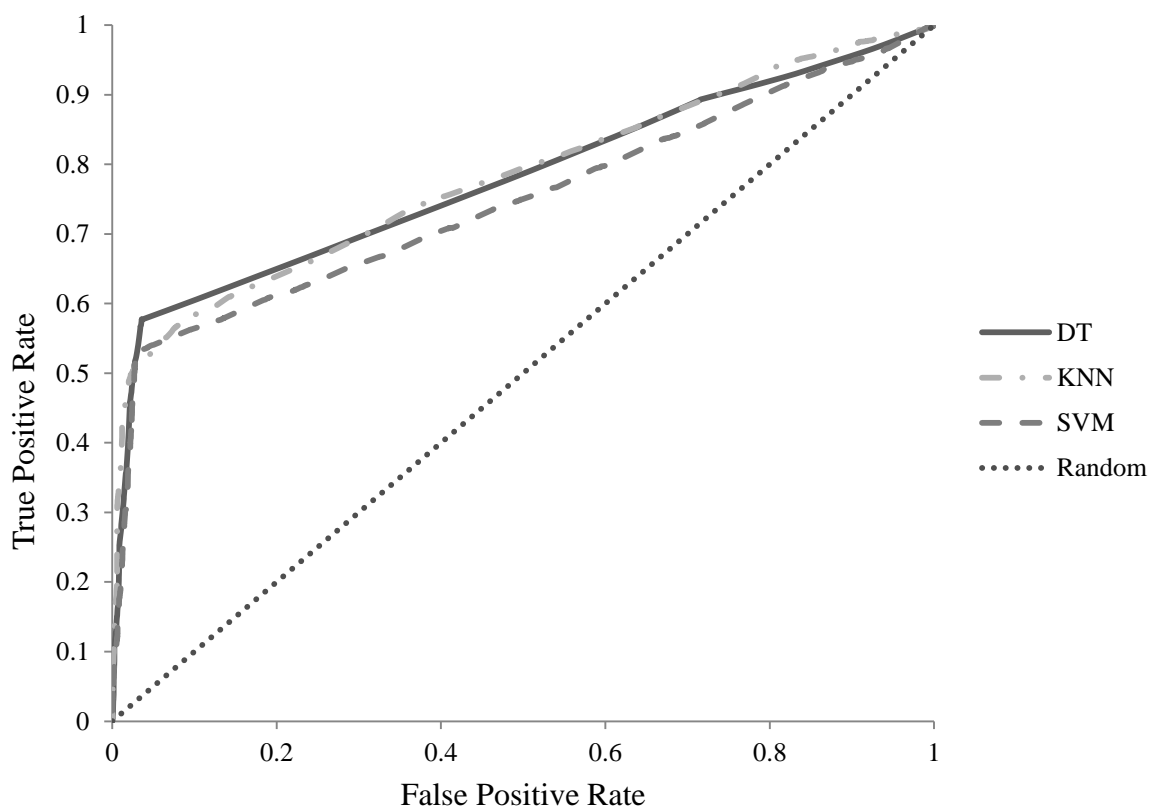


Figure 29. Service Data ROC Curves

Finally, the F-Measure was used to measure the performance of the classification algorithms across the data sets. The decision tree algorithm performed better than all others using the RLF and J48 wrapper data sets, with a performance of 91%, which was followed by K-NN with 90% and SVM with 84% (Table 47).

Table 47. F-Measures Results for Service Data Set with Optimized Parameters Using Experimenter

Dataset	Decision Tree	K-NN	SVM
CFS	0.88	0.87 *	0.81 *
IG	0.89	0.87 *	0.82 *
RLF	0.91	0.90 *	0.84 *
SVM Wrapper	0.89	0.87 *	0.82 *
J48 Wrapper	0.91	0.89 *	0.84 *
K-NN Wrapper	0.90	0.89 *	0.83 *
	(v/ /*)	(0/0/6)	(0/0/6)

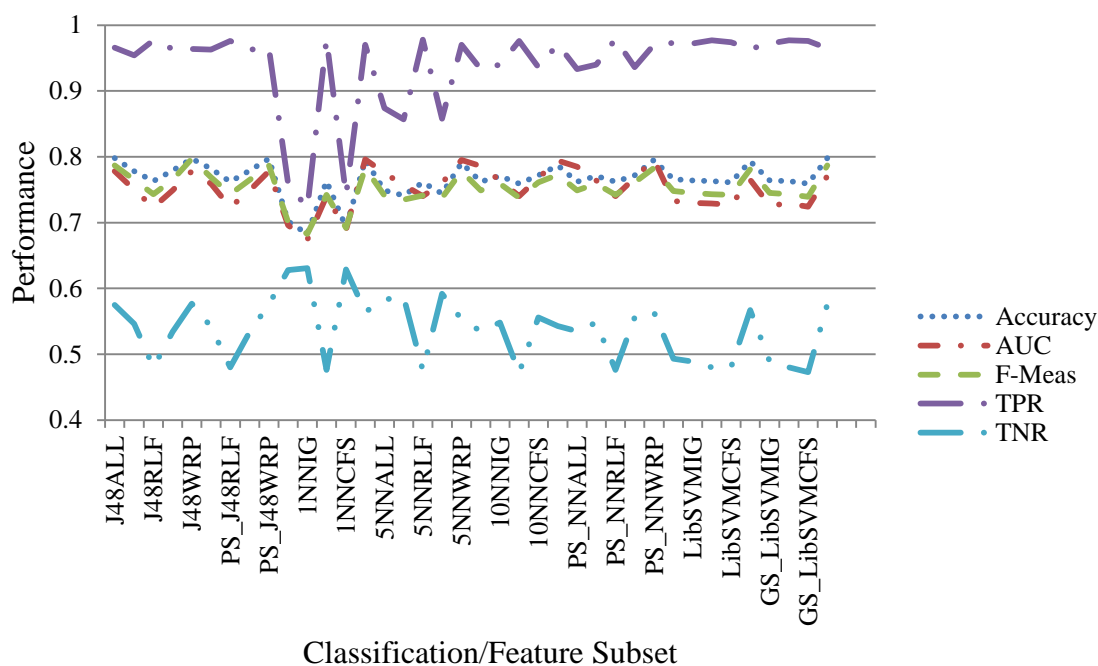


Figure 30. Service Data Set Classification Performance Measures across Data Sets

Running the classification tests using different feature subsets in our target domain showed that while the average performance may be acceptable the True Negative Rates are lower than expected (Figure 30).

Summary

This chapter focused on presenting and comparing different feature selection methods and the predictive accuracy as applied to the three different classification algorithms when we applied them on two distinct domains. In addition, different performance measures used to evaluate the classification algorithms were presented.

The first domain, marketing data from a bank, was obtained from the UCI repository. The main goal was to identify clients who would likely buy long term deposits based on previous contact and history.

The second domain consisted of automotive service history. Here our goal was to identify car owners who would be more likely to buy a new vehicle based on the service histories of their current vehicles.

Chapter 5

Conclusions, Recommendations, and Summary

Introduction

As competition grows among companies, the value of client retention becomes increasingly important. With computers' speed increasing and the cost of disk space decreasing, the amount of customer data being retained increases. One effective way of analyzing this data is through data mining. In this report, we concentrated on the automotive industry domain, more specifically information captured by the service departments. Our goal was to determine potential car buyers based on service history of their current vehicle by using 3 different types of classification algorithms.

The goal in this research was made up of five related sub-goals as follows:

- 1) Compared and contrasted different feature selection methods against the mentioned high dimensional data sets and a reference data set. Both filter and wrapper methods were applied to these data sets and their results were compared and analyzed. The classification accuracy achieved by each method was compared against the better feature selection method found.
- 2) Repeated the above procedure using different classification methods, including C4.5, a Decision Tree algorithm, K-Nearest Neighbor (K-NN), and Support Vector Machine (SVM) algorithms.

- 3) Compared the accuracy of the classification algorithms using the best attributes selected from each algorithm. All methods were tested and validated on a binary classification, bought or not bought, task.
- 4) Used different thresholds in the classification systems and compare the effects on accuracy. K values in K-NN, pruning in Decision Trees, and Cost of Error and Gamma settings in the SVM algorithm.
- 5) Determine which classification algorithm and feature selection combination produced the better results in order to determine new potential car buyers.

Conclusions

Applying feature selection methods to our data sets produced mixed results. In the case of the bank reference data set applying the *information gain*, Correlation Feature Selection (CFS) and Relief-F filter methods increased our accuracy results as opposed to using all attributes only when using the K-NN classification algorithm. The True Positive Rates and True Negative Rates were in line with the rest of the performance measures. On the other hand, our target domain only saw an increase in accuracy after using the attributes selected by the Relief-F method. The accuracy increased from 70.2 % to 76.3% by using 1NN classification. Information Gain and CFS methods showed degradation in performance. Our target domain showed a bias to the positive class as shown by the lower True Negative Rate values versus the higher True Positive Rates. This may be attributed to the imbalance of the data set.

Using the wrapper method of feature selection increased the classification accuracy of the Decision Tree and Nearest Neighbor, for example, when $k = 5$, in the reference domain. In all other tests, attributes selected by the Relief-F method performed better

than all other feature selection methods. In the target domain, using wrapper data sets increased the accuracy rate of the Nearest Neighbor and SVM algorithms but underperformed by 0.1% when compared to the Decision Tree using all attributes. The second best performing feature selection method was CFS, and then it was followed by Relief-F.

Fine tuning of the parameters of our classification systems improved our accuracy and AUC rates most of the time. An increase in accuracy from 83.9% to 86.4% was seen in our reference data when the confidence factor parameter of the decision tree algorithm was decreased. No increase was seen on our target domain. Optimizing the Cost and Gamma parameters of the SVM classification algorithms produced better results in both domains.

Results of our 38 runs using different feature sets and classification algorithms indicated that the SVM classification algorithm produced the highest accuracy of 79.8% in our target domain. This SVM accuracy was obtained when optimizing the cost of error and gamma parameters to 1.0 and 1.0 respectively in the wrapper selection method. Using the decision tree algorithm resulted in the same accuracy rate when using all attributes. However, when using AUC as a performance measure nearest neighbor with $k = 1$ performed better than the other algorithms with an AUC of 79.6% but produced a lower recall of 79.5% than that of our best performer using the accuracy measure.

Results on our target domain showed that there is no statistical difference between using the Decision Tree algorithm with all attributes and SVM with the wrapper subset.

Since the computing cost of using all attributes was fairly high using the SVM model with the wrapper data set is recommended.

Implications

From the conclusions just discussed, several implications were provided in the following observations and suggestions. Results of this study indicate and confirm that there is no specific classification algorithm that would work effectively across all domains. In addition, multiple factors, such as pre-processing and feature selection, may affect the results of any classification algorithm. It is hoped that this study advances the understanding that these factors may have in the selection of a data mining methodology. This study probably invites more research in the area as identified in the Recommendations section as well

Recommendations

Based on the data in this study and the conclusions drawn, the following topics for additional research are recommended:

1. On our target domain we noticed that the scoring of our attributes was low by using Information Gain and Relief feature selection methods. While the attributes did contribute to the classification, there was no significant attribute that stood out. To overcome this, we recommend using a feature extraction method such as Principal Component Analysis (PCA). PCA uses existing attributes to create new ones that may have a better association with the output class.

2. This study was concentrated on the three most popular classification algorithms; Decision Trees, Nearest Neighbor, and Support Vector Machines. Ensemble

methods which use multiple algorithms to obtain better predictive performance may be used. In addition, other types of classification algorithms, such as Neural Networks, which have been successfully used in other domains (Lam et al., 2014), may be also be applied to the data sets and compared against the base classification systems.

3. Finally, alternative feature selection methods to those tested may be implemented to improve the predictive accuracy of the tested algorithms. For example, K-means and hierarchal clustering (Guyon & Elisseeff, 2003) of features may also be studied.

Summary

This paper is focused on the comparison of 4 different feature selection methods across 3 classification algorithms on data sets pertaining to our 2 domains, Bank Marketing and Vehicle Service data. The main tasks were followed the three main phases of the CRISP-DM process:

- Data Preparation
- Modeling
- Evaluation

The first step was to prepare the data for use in our modeling stage. In the Bank domain, no missing values had to be accounted for, but attributes which were discretized had to be transformed since the K-NN and SVM classification algorithms do not handle discretized data. These attributes were transformed to separate attributes with values denoting their original state. One drawback to this process is that it increases the number of attributes in the working data set. In this case however, the increase was not

significant. Another step in the preprocessing stage was to normalize the values in the data. Again, the K-NN and SVM classification algorithms use distance measures in their calculations. It may affect the final accuracy rate with numeric values in different scales. This was rectified in both data sets by normalizing the numeric values to a value between 0 and 1.

The next step in the data preparation stage was to apply the three filter selection methods; Information Gain, Correlation Feature Selection (CFS), and Relief-F, to our data sets. The wrapper and hybrid methods were also applied. This process resulted in the following data sets for each domain:

- Domain_ALL – Data set with all attributes
- DomainName_IG – Data set chosen using the Information Gain method
- Domain_Name_RLF – Data set containing attributes selected by the Relief-F method
- DomainName_CFS – Data set containing attributes selected by the CFS method
- DomainName_J48_WRP – Data set composed of attributes selected by the wrapper method using the J48 classification algorithm
- DomainName_K-NN_WRP - Data set composed of attributes selected by the wrapper method using the K-NN classification algorithm
- DomainName_SVM_WRP - Data set composed of attributes selected by the wrapper method using the SVM classification algorithm

All feature selection methods indicated that the “duration” attribute was the most significant in the Bank data set. In addition, “poutcome=success”, “contact=unknown”, and “age” were also ranked as significant. Running the feature selection on the Service data set resulted in the following common attributes being selected “39CTZ22”, “76CTZ01”, “39CVZ21”, and “76CTZDLRTRANS”. Other attributes were selected as well depending on the feature selection method applied. It should be noted that our highest ranking attribute, “duration”, in the Bank data set had an Information Gain score

of .32 yet in the Service data set the highest ranking attribute, “39CTZ22”, only scored .075.

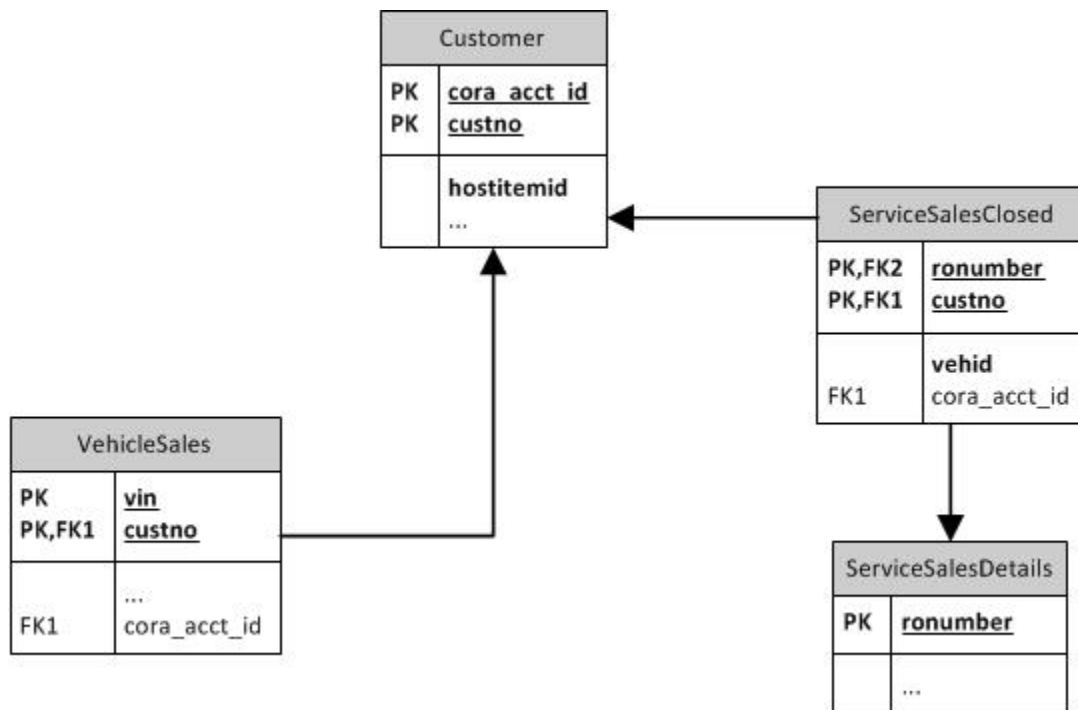
During the modeling stage, we ran our classification algorithms on each data set based on attributes selected by each feature selection method. Initial runs were performed with the default parameters for the classification function in WEKA (Appendix C). After the initial runs were performed, the classification models were executed again but with optimized parameters as selected by the Parameter Search and Grid Search modelers in WEKA.

The best accuracy performance in the Bank domain was achieved by the Decision Tree algorithm with features selected by the wrapper method and a confidence factor of 0.1. The precision accuracy obtained was 86.4%. This same classification/feature selection combination also produced the highest AUC of 90.4%. Running the K Nearest Neighbor with $k = 5$ and wrapper attributes also produced an AUC score of 90.4%, but accuracy was only 84.9%.

Using the SVM classification with the SVM wrapper data set produced the highest accuracy, 79.8%, in our Service domain. This same accuracy was also achieved by the Decision Tree algorithm using all attributes. The best performing classification using AUC as a measuring tool was Nearest Neighbor with $k = 1$ and the NN wrapper data set. An AUC of 79.6% was achieved in this case.

Appendix A

Data Schema



Appendix B

Data Dictionaries

Note. Attribute names have been changed to protect proprietary content.

Table Name: Customer

	Column_name	Type	Computed	Length	Prec	Scale	Nullable
1	CustAttribute1	int	No	4	10	0	no
2	CustAttribute2	varchar	No	40			no
3	CustAttribute3	varchar	No	10			yes
4	CustAttribute4	varchar	No	17			no
5	CustAttribute5	varchar	No	45			yes
6	CustAttribute6	varchar	No	1			yes
7	CustAttribute7	datetime2	No	8	27	7	yes
8	CustAttribute8	datetime2	No	8	27	7	yes
9	CustAttribute9	varchar	No	15			yes
10	CustAttribute10	varchar	No	7			yes
11	CustAttribute11	varchar	No	10			yes
12	CustAttribute12	varchar	No	35			yes
13	CustAttribute13	varchar	No	255			yes
14	CustAttribute14	datetime2	No	8	27	7	yes
15	CustAttribute15	varchar	No	10			yes
16	CustAttribute16	varchar	No	1			yes
17	CustAttribute17	varchar	No	1			yes
18	CustAttribute18	varchar	No	1			yes
19	CustAttribute19	varchar	No	1			yes
20	CustAttribute20	varchar	No	1			yes
21	CustAttribute21	varchar	No	1			yes
22	CustAttribute22	varchar	No	1			yes
23	CustAttribute23	varchar	No	1			yes
24	CustAttribute24	numeric	No	9	19	4	yes
25	CustAttribute25	numeric	No	9	19	4	yes
26	CustAttribute26	varchar	No	17			yes
27	CustAttribute27	varchar	No	2			yes
28	CustAttribute28	datetime2	No	8	27	7	yes
29	CustAttribute29	varchar	No	4			yes
30	CustAttribute30	datetime2	No	8	27	7	yes
31	CustAttribute31	varchar	No	20			yes
32	CustAttribute32	varchar	No	50			yes
33	CustAttribute33	varchar	No	50			yes

	Column_name	Type	Computed	Length	Prec	Scale	Nullable
34	CustAttribute34	varchar	No	50			yes
35	CustAttribute35	varchar	No	10			yes
36	CustAttribute36	varchar	No	10			yes
37	CustAttribute37	varchar	No	10			yes
38	CustAttribute38	varchar	No	32			yes
39	CustAttribute39	varchar	No	25			yes
40	CustAttribute40	varchar	No	10			yes
41	CustAttribute41	varchar	No	37			yes
42	CustAttribute42	varchar	No	10			yes
43	CustAttribute43	varchar	No	12			yes
44	CustAttribute44	varchar	No	1			yes
45	CustAttribute45	varchar	No	2			yes
46	CustAttribute46	varchar	No	40			yes
47	CustAttribute47	numeric	No	9	19	4	yes
48	CustAttribute48	datetime2	No	8	27	7	yes
49	CustAttribute49	numeric	No	9	19	4	yes
50	CustAttribute50	datetime2	No	8	27	7	yes
51	CustAttribute51	datetime2	No	8	27	7	yes
52	CustAttribute52	varchar	No	25			yes
53	CustAttribute53	varchar	No	45			yes
54	CustAttribute54	varchar	No	45			yes
55	CustAttribute55	varchar	No	1			yes
56	CustAttribute56	varchar	No	30			yes
57	CustAttribute57	numeric	No	9	19	4	yes
58	CustAttribute58	numeric	No	9	19	4	yes
59	CustAttribute59	numeric	No	9	19	4	yes
60	CustAttribute60	numeric	No	9	19	4	yes
61	CustAttribute61	varchar	No	10			yes
62	CustAttribute62	varchar	No	10			yes
63	CustAttribute63	varchar	No	2			yes
64	CustAttribute64	varchar	No	10			yes
65	CustAttribute65	varchar	No	1			yes
66	CustAttribute66	varchar	No	10			yes
67	CustAttribute67	varchar	No	255			yes
68	CustAttribute68	datetime2	No	8	27	7	yes
69	CustAttribute69	varchar	No	10			yes
70	CustAttribute70	varchar	No	4			yes
71	CustAttribute71	varchar	No	40			yes
72	CustAttribute72	varchar	No	40			yes
73	CustAttribute73	datetime2	No	8	27	7	yes

	Column_name	Type	Computed	Length	Prec	Scale	Nullable
74	CustAttribute74	varchar	No	20			yes
75	CustAttribute75	varchar	No	10			yes
76	CustAttribute76	varchar	No	2			yes
77	CustAttribute77	varchar	No	40			yes
78	CustAttribute78	varchar	No	11			yes
79	CustAttribute79	varchar	No	3			yes
80	CustAttribute80	varchar	No	4			yes
81	CustAttribute81	varchar	No	2			yes
82	CustAttribute82	varchar	No	10			yes
83	CustAttribute83	varchar	No	30			yes
84	CustAttribute84	varchar	No	30			yes
85	CustAttribute85	numeric	No	9	19	4	yes
86	CustAttribute86	numeric	No	9	19	4	yes
87	CustAttribute87	numeric	No	9	19	4	yes
88	CustAttribute88	varchar	No	10			yes
89	CustAttribute89	numeric	No	9	19	4	yes
90	CustAttribute90	varchar	No	10			yes
91	CustAttribute91	datetime2	No	8	27	7	yes
92	CustAttribute92	varchar	No	1024			yes
93	CustAttribute93	varchar	No	1024			yes

Table Name: VehicleSales

	Column_name	Type	Length	Prec	Scale	Nullable
1	VSAttribute1	int	4	10	0	no
2	VSAttribute2	varchar	40			no
3	VSAttribute3	varchar	17			no
4	VSAttribute4	datetime2	8	27	7	yes
5	VSAttribute5	numeric	9	19	0	yes
6	VSAttribute6	numeric	9	19	4	yes
7	VSAttribute7	numeric	9	19	4	yes
8	VSAttribute8	varchar	17			yes
9	VSAttribute9	numeric	9	19	4	yes
10	VSAttribute10	varchar	17			yes
11	VSAttribute11	numeric	9	19	4	yes
12	VSAttribute12	numeric	9	19	4	yes
13	VSAttribute13	numeric	9	19	4	yes
14	VSAttribute14	varchar	17			yes
15	VSAttribute15	numeric	9	15	4	yes
16	VSAttribute16	numeric	9	15	4	yes
17	VSAttribute17	numeric	9	15	5	yes
18	VSAttribute18	varchar	20			yes
19	VSAttribute19	numeric	9	19	4	yes
20	VSAttribute20	numeric	9	19	4	yes
21	VSAttribute21	numeric	9	19	4	yes
22	VSAttribute22	varchar	17			yes
23	VSAttribute23	numeric	9	19	4	yes
24	VSAttribute24	varchar	50			yes
25	VSAttribute25	varchar	20			yes
26	VSAttribute26	numeric	9	19	4	yes
27	VSAttribute27	numeric	9	19	4	yes
28	VSAttribute28	numeric	9	19	4	yes
29	VSAttribute29	datetime2	8	27	7	yes
30	VSAttribute30	varchar	40			yes
31	VSAttribute31	datetime2	8	27	7	yes
32	VSAttribute32	varchar	40			yes
33	VSAttribute33	datetime2	8	27	7	yes
34	VSAttribute34	numeric	9	19	4	yes
35	VSAttribute35	varchar	17			yes
36	VSAttribute36	varchar	15			yes
37	VSAttribute37	datetime2	8	27	7	yes
38	VSAttribute38	varchar	15			yes
39	VSAttribute39	datetime2	8	27	7	yes

	Column_name	Type	Length	Prec	Scale	Nullable
40	VSAttribute40	varchar	15			yes
41	VSAttribute41	datetime2	8	27	7	yes
42	VSAttribute42	varchar	15			yes
43	VSAttribute43	datetime2	8	27	7	yes
44	VSAttribute44	varchar	10			yes
45	VSAttribute45	varchar	13			yes
46	VSAttribute46	varchar	10			yes
47	VSAttribute47	varchar	15			yes
48	VSAttribute48	numeric	9	19	4	yes
49	VSAttribute49	varchar	17			yes
50	VSAttribute50	varchar	17			yes
51	VSAttribute51	numeric	9	19	4	yes
52	VSAttribute52	varchar	10			yes
53	VSAttribute53	datetime2	8	27	7	yes
54	VSAttribute54	varchar	1			yes
55	VSAttribute55	numeric	9	19	4	yes
56	VSAttribute56	numeric	9	19	4	yes
57	VSAttribute57	numeric	9	19	4	yes
58	VSAttribute58	numeric	9	19	4	yes
59	VSAttribute59	numeric	9	19	4	yes
60	VSAttribute60	numeric	9	19	4	yes
61	VSAttribute61	numeric	9	19	4	yes
62	VSAttribute62	numeric	9	19	4	yes
63	VSAttribute63	datetime2	8	27	7	yes
64	VSAttribute64	numeric	9	19	4	yes
65	VSAttribute65	varchar	8			yes
66	VSAttribute66	varchar	10			yes
67	VSAttribute67	varchar	30			yes
68	VSAttribute68	numeric	9	19	4	yes
69	VSAttribute69	numeric	9	19	4	yes
70	VSAttribute70	varchar	25			yes
71	VSAttribute71	varchar	10			yes
72	VSAttribute72	varchar	30			yes
73	VSAttribute73	numeric	9	19	4	yes
74	VSAttribute74	numeric	9	19	0	yes
75	VSAttribute75	varchar	10			yes
76	VSAttribute76	datetime2	8	27	7	yes
77	VSAttribute77	datetime2	8	27	7	yes
78	VSAttribute78	datetime2	8	27	7	yes
79	VSAttribute79	numeric	9	19	4	yes

	Column_name	Type	Length	Prec	Scale	Nullable
80	VSAttribute80	numeric	9	19	4	yes
81	VSAttribute81	numeric	9	19	4	yes
82	VSAttribute82	numeric	9	19	0	yes
83	VSAttribute83	numeric	9	19	0	yes
84	VSAttribute84	numeric	9	19	0	yes
85	VSAttribute85	varchar	17			yes
86	VSAttribute86	datetime2	8	27	7	yes
87	VSAttribute87	varchar	17			yes
88	VSAttribute88	varchar	17			yes
89	VSAttribute89	varchar	17			yes
90	VSAttribute90	varchar	17			yes
91	VSAttribute91	numeric	9	15	4	yes
92	VSAttribute92	numeric	9	15	4	yes
93	VSAttribute93	numeric	9	15	5	yes
94	VSAttribute94	varchar	17			yes
95	VSAttribute95	numeric	9	19	4	yes
96	VSAttribute96	int	4	10	0	yes
97	VSAttribute97	numeric	9	19	4	yes
98	VSAttribute98	numeric	9	19	4	yes
99	VSAttribute99	varchar	10			yes
100	VSAttribute100	varchar	10			yes
101	VSAttribute101	numeric	9	19	4	yes
102	VSAttribute102	numeric	9	19	4	yes
103	VSAttribute103	varchar	17			yes
104	VSAttribute104	int	4	10	0	yes
105	VSAttribute105	numeric	9	19	4	yes
106	VSAttribute106	varchar	10			yes
107	VSAttribute107	varchar	10			yes
108	VSAttribute108	numeric	9	19	4	yes
109	VSAttribute109	numeric	9	19	4	yes
110	VSAttribute110	varchar	17			yes
111	VSAttribute111	int	4	10	0	yes
112	VSAttribute112	varchar	17			yes
113	VSAttribute113	int	4	10	0	yes
114	VSAttribute114	datetime2	8	27	7	yes
115	VSAttribute115	varchar	1024			yes
116	VSAttribute116	numeric	9	19	4	yes

Table: ServiceSalesClosed

	Column_name	Type	Length	Prec	Scale	Nullable
1	SSDAttribute1	int	4	10	0	no
2	SSDAttribute2	varchar	40			no
3	SSDAttribute3	varchar	17			yes
4	SSDAttribute4	numeric	9	12	2	yes
5	SSDAttribute5	numeric	9	12	2	yes
6	SSDAttribute6	numeric	9	12	2	yes
7	SSDAttribute7	numeric	9	12	2	yes
8	SSDAttribute8	datetime2	8	27	7	yes
9	SSDAttribute9	varchar	1			yes
10	SSDAttribute10	varchar	8			yes
11	SSDAttribute11	varchar	20			yes
12	SSDAttribute12	datetime2	8	27	7	yes
13	SSDAttribute13	varchar	1			yes
14	SSDAttribute14	varchar	17			yes
15	SSDAttribute15	int	4	10	0	yes
16	SSDAttribute16	int	4	10	0	yes
17	SSDAttribute17	int	4	10	0	yes
18	SSDAttribute18	numeric	9	19	4	yes
19	SSDAttribute19	numeric	9	19	4	yes
20	SSDAttribute20	numeric	9	19	4	yes
21	SSDAttribute21	numeric	9	19	4	yes
22	SSDAttribute22	numeric	9	19	4	yes
23	SSDAttribute23	numeric	9	19	4	yes
24	SSDAttribute24	numeric	9	19	4	yes
25	SSDAttribute25	numeric	9	19	4	yes
26	SSDAttribute26	numeric	9	19	0	yes
27	SSDAttribute27	numeric	9	19	4	yes
28	SSDAttribute28	numeric	9	19	4	yes
29	SSDAttribute29	numeric	9	19	4	yes
30	SSDAttribute30	numeric	9	19	4	yes
31	SSDAttribute31	numeric	9	19	4	yes
32	SSDAttribute32	numeric	9	19	4	yes
33	SSDAttribute33	numeric	9	19	4	yes
34	SSDAttribute34	numeric	9	19	4	yes
35	SSDAttribute35	datetime2	8	27	7	yes
36	SSDAttribute36	varchar	20			yes
37	SSDAttribute37	numeric	9	19	4	yes
38	SSDAttribute38	numeric	9	19	4	yes
39	SSDAttribute39	numeric	9	19	4	yes

	Column_name	Type	Length	Prec	Scale	Nullable
40	SSDAttribute40	numeric	9	19	4	yes
41	SSDAttribute41	numeric	9	19	4	yes
42	SSDAttribute42	numeric	9	19	4	yes
43	SSDAttribute43	numeric	9	19	4	yes
44	SSDAttribute44	numeric	9	19	4	yes
45	SSDAttribute45	varchar	12			yes
46	SSDAttribute46	varchar	17			yes
47	SSDAttribute47	numeric	9	12	2	yes
48	SSDAttribute48	numeric	9	12	2	yes
49	SSDAttribute49	numeric	9	12	2	yes
50	SSDAttribute50	numeric	9	12	2	yes
51	SSDAttribute51	numeric	9	19	4	yes
52	SSDAttribute52	numeric	9	19	4	yes
53	SSDAttribute53	numeric	9	19	4	yes
54	SSDAttribute54	numeric	9	19	4	yes
55	SSDAttribute55	numeric	9	19	4	yes
56	SSDAttribute56	numeric	9	19	4	yes
57	SSDAttribute57	numeric	9	12	2	yes
58	SSDAttribute58	numeric	9	12	2	yes
59	SSDAttribute59	numeric	9	12	2	yes
60	SSDAttribute60	numeric	9	12	2	yes
61	SSDAttribute61	varchar	17			yes
62	SSDAttribute62	datetime2	8	27	7	yes
63	SSDAttribute63	datetime2	8	27	7	yes
64	SSDAttribute64	datetime2	8	27	7	yes

Table: ServiceSalesDetailsClosed

	Column_name	Type	Computed	Length	Prec	Scale	Nullable
1	SSDCAttribute1	int	No	4	10	0	no
2	SSDCAttribute2	varchar	No	40			no
3	SSDCAttribute3	varchar	No	17			yes
4	SSDCAttribute4	numeric	No	9	14	2	yes
5	SSDCAttribute5	varchar	No	17			yes
6	SSDCAttribute6	varchar	No	55			yes
7	SSDCAttribute7	varchar	No	1			yes
8	SSDCAttribute8	varchar	No	20			yes
9	SSDCAttribute9	varchar	No	17			yes
10	SSDCAttribute10	varchar	No	17			yes
11	SSDCAttribute11	varchar	No	20			yes
12	SSDCAttribute12	varchar	No	30			yes
13	SSDCAttribute13	int	No	4	10	0	yes
14	SSDCAttribute14	varchar	No	21			yes
15	SSDCAttribute15	numeric	No	13	24	4	yes
16	SSDCAttribute16	numeric	No	13	24	4	yes
17	SSDCAttribute17	varchar	No	5			yes
18	SSDCAttribute18	varchar	No	3			yes
19	SSDCAttribute19	numeric	No	13	24	4	yes
20	SSDCAttribute20	numeric	No	13	24	4	yes
21	SSDCAttribute21	varchar	No	20			yes
22	SSDCAttribute22	varchar	No	70			yes
23	SSDCAttribute23	numeric	No	13	24	4	yes
24	SSDCAttribute24	numeric	No	13	24	4	yes
25	SSDCAttribute25	varchar	No	12			yes
26	SSDCAttribute26	varchar	No	250			yes
27	SSDCAttribute27	numeric	No	9	14	2	yes
28	SSDCAttribute28	numeric	No	9	14	2	yes
29	SSDCAttribute29	varchar	No	17			yes
30	SSDCAttribute30	datetime2	No	8	27	7	yes
31	SSDCAttribute31	varchar	No	1024			yes
32	SSDCAttribute32	varchar	No	1024			yes
33	SSDCAttribute33	varchar	No	1024			yes
34	SSDCAttribute34	varchar	No	12			no

Bank ARFF File

```

@relation 'bank-fullxl-
weka.filters.supervised.attribute.NominalToBinary-
weka.filters.supervised.attribute.Discretize-R1-
weka.filters.supervised.attribute.Discretize-R22,28,41,42,43'

@attribute age {'\*(-inf-25.5]\'', '\'(25.5-29.5]\'', '\'(29.5-
60.5]\'', '\'(60.5-inf)\''}
@attribute job=management numeric
@attribute job=technician numeric
@attribute job=entrepreneur numeric
@attribute job=blue-collar numeric
@attribute job=unknown numeric
@attribute job=retired numeric
@attribute job=admin. numeric
@attribute job=services numeric
@attribute job=self-employed numeric
@attribute job=unemployed numeric
@attribute job=housemaid numeric
@attribute job=student numeric
@attribute marital=married numeric
@attribute marital=single numeric
@attribute marital=divorced numeric
@attribute education=tertiary numeric
@attribute education=secondary numeric
@attribute education=unknown numeric
@attribute education=primary numeric
@attribute default numeric
@attribute balance {'\*(-inf--46.5]\'', '\'(-46.5-
105.5]\'', '\'(105.5-1578.5]\'', '\'(1578.5-inf)\''}
@attribute housing numeric
@attribute loan numeric
@attribute contact=unknown numeric
@attribute contact=cellular numeric
@attribute contact=telephone numeric
@attribute day {'\*(-inf-1.5]\'', '\'(1.5-4.5]\'', '\'(4.5-
9.5]\'', '\'(9.5-10.5]\'', '\'(10.5-16.5]\'', '\'(16.5-
21.5]\'', '\'(21.5-25.5]\'', '\'(25.5-27.5]\'', '\'(27.5-
29.5]\'', '\'(29.5-30.5]\'', '\'(30.5-inf)\''}
@attribute month=may numeric
@attribute month=jun numeric
@attribute month=jul numeric
@attribute month=aug numeric
@attribute month=oct numeric
@attribute month=nov numeric
@attribute month=dec numeric
@attribute month=jan numeric
@attribute month=feb numeric
@attribute month=mar numeric
@attribute month=apr numeric

```

```
@attribute month=sep numeric
@attribute duration {'\'}(-inf-77.5]\'', '\'(77.5-
130.5]\'', '\'(130.5-206.5]\'', '\'(206.5-259.5]\'', '\'(259.5-
410.5]\'', '\'(410.5-521.5]\'', '\'(521.5-647.5]\'', '\'(647.5-
827.5]\'', '\'(827.5-inf)\''}
@attribute campaign {'\'}(-inf-1.5]\'', '\'(1.5-3.5]\'', '\'(3.5-
11.5]\'', '\'(11.5-inf)\''}
@attribute pdays {'\'}(-inf-8.5]\'', '\'(8.5-86.5]\'', '\'(86.5-
99.5]\'', '\'(99.5-107.5]\'', '\'(107.5-177.5]\'', '\'(177.5-
184.5]\'', '\'(184.5-203.5]\'', '\'(203.5-316.5]\'', '\'(316.5-
373.5]\'', '\'(373.5-inf)\''}
@attribute previous numeric
@attribute poutcome=unknown numeric
@attribute poutcome=failure numeric
@attribute poutcome=other numeric
@attribute poutcome=success numeric
@attribute y {no,yes}
```

Service ARFF File

```
@relation ServicewMileage-
weka.filters.unsupervised.attribute.Remove-R1-2-
weka.filters.unsupervised.attribute.Normalize-S1.0-T0.0
```

```
@attribute Age numeric
@attribute Mileage numeric
@attribute 46 {0,1}
@attribute 46CTZ {0,1}
@attribute 46CVZ {0,1}
@attribute 11 {0,1}
@attribute 39 {0,1}
@attribute 29 {0,1}
@attribute SPO {0,1}
@attribute 03CTZ {0,1}
@attribute 22CTZ {0,1}
@attribute RT {0,1}
@attribute 11CTZ {0,1}
@attribute 29CTZ {0,1}
@attribute 46CTZROTATE {0,1}
@attribute 22 {0,1}
@attribute PDI {0,1}
@attribute 10 {0,1}
@attribute 25 {0,1}
@attribute 39CTZ22 {0,1}
@attribute 76CTZ01 {0,1}
@attribute 76CTZ {0,1}
@attribute 22CVZ {0,1}
@attribute 30CTZ {0,1}
@attribute 03CVZ {0,1}
@attribute 02CTZ {0,1}
@attribute 11CVZ {0,1}
@attribute 25CTZ {0,1}
@attribute 20 {0,1}
@attribute 06CTZ {0,1}
@attribute 20CTZ {0,1}
@attribute 46CVZROTATE {0,1}
@attribute 9999 {0,1}
@attribute 29CTZ1 {0,1}
@attribute 10CTZ {0,1}
@attribute 02CTZ029LOF {0,1}
@attribute 03CTZ1 {0,1}
@attribute NCK {0,1}
@attribute 22CTZ1 {0,1}
@attribute 29CVZ {0,1}
@attribute 76 {0,1}
@attribute NVP {0,1}
@attribute 30CTZ1 {0,1}
```

@attribute 02CVZ {0,1}
@attribute 02CTZFUELFILTER {0,1}
@attribute 76CTZDLRTRANS {0,1}
@attribute 16CTZ {0,1}
@attribute 11CTZ1 {0,1}
@attribute 5 {0,1}
@attribute 25CTZ1 {0,1}
@attribute 16 {0,1}
@attribute 10CVZ {0,1}
@attribute 02CVZ029LOF {0,1}
@attribute RB {0,1}
@attribute 06CVZ {0,1}
@attribute 11CTZWRNLTON {0,1}
@attribute 76CTZETCH {0,1}
@attribute 39CTZ21 {0,1}
@attribute 30 {0,1}
@attribute 01CTZ5 {0,1}
@attribute 29CTZ2 {0,1}
@attribute 39CVZ21 {0,1}
@attribute E7700 {0,1}
@attribute 02CTZROTTIRES {0,1}
@attribute 39CVZ22 {0,1}
@attribute 76CVZ01 {0,1}
@attribute 38CTZ1 {0,1}
@attribute 02CTZGTSTRB {0,1}
@attribute 76CTZ02 {0,1}
@attribute 22CTZ2 {0,1}
@attribute 76CVZ {0,1}
@attribute 76CVZ02 {0,1}
@attribute 03CTZ2 {0,1}
@attribute 22CVZ1 {0,1}
@attribute 20CVZ {0,1}
@attribute 40 {0,1}
@attribute 05CTZALIGN2 {0,1}
@attribute 39CTZ {0,1}
@attribute 30CVZ {0,1}
@attribute 1 {0,1}
@attribute 11CVZ1 {0,1}
@attribute 02CTZAIRFILTER {0,1}
@attribute 02CTZ00003K {0,1}
@attribute 03CVZ1 {0,1}
@attribute 20CTZ1 {0,1}
@attribute SPCL {0,1}
@attribute 39CVZ {0,1}
@attribute 01CTZ01 {0,1}
@attribute 29CVZ1 {0,1}
@attribute 10CTZCOOLLK {0,1}
@attribute MTF12 {0,1}
@attribute 6 {0,1}
@attribute 11CVZWRNLTON {0,1}
@attribute 10CTZ1 {0,1}
@attribute 02CTZ01 {0,1}

@attribute UCI {0,1}
@attribute 01CVZ {0,1}
@attribute ELE {0,1}
@attribute 76CVZETCH {0,1}
@attribute 38CVZ1 {0,1}
@attribute 10CVZCOOLLK {0,1}
@attribute ROT {0,1}
@attribute 25CVZ {0,1}
@attribute 46CTSFFILTER1 {0,1}
@attribute 11CTZ2 {0,1}
@attribute 02CVZFFILTER {0,1}
@attribute DRV {0,1}
@attribute 16CTZ1 {0,1}
@attribute N4180 {0,1}
@attribute 25CTZ2 {0,1}
@attribute 76CTZ1 {0,1}
@attribute 02CVZAIRFILTER {0,1}
@attribute 02CVZ00003K {0,1}
@attribute 39CTZ26 {0,1}
@attribute 16CVZ {0,1}
@attribute 03CTZ4 {0,1}
@attribute 02CTZ1 {0,1}
@attribute 03CVZ2 {0,1}
@attribute 22CVZ2 {0,1}
@attribute 02CTZBELTS {0,1}
@attribute 8 {0,1}
@attribute 05CTZ {0,1}
@attribute 01CTZ {0,1}
@attribute 01CVZ01 {0,1}
@attribute NWD {0,1}
@attribute 76CVZDLRTRANS {0,1}
@attribute 02CVZGTSTRB {0,1}
@attribute 02CVZROTTIRES {0,1}
@attribute N0110 {0,1}
@attribute 29CVZ2 {0,1}
@attribute TRM {0,1}
@attribute 10CTZOILLEAK {0,1}
@attribute ALI2 {0,1}
@attribute 01CVZ5 {0,1}
@attribute 30CVZ1 {0,1}
@attribute 39CTZ1 {0,1}
@attribute 10CVZOILLEAK {0,1}
@attribute 02CTZROTBAL {0,1}
@attribute 02CTBRONZE {0,1}
@attribute 76CVZ1 {0,1}
@attribute 46CTSSPEC {0,1}
@attribute 38CTZ {0,1}
@attribute 29CTZ3 {0,1}
@attribute 10CVZ1 {0,1}
@attribute 65CTZ1 {0,1}
@attribute 38 {0,1}
@attribute 02CTZMISC {0,1}

@attribute 60CTZ1 {0,1}
@attribute 39CTZ28 {0,1}
@attribute Y0124 {0,1}
@attribute GM {0,1}
@attribute 30CVZ4 {0,1}
@attribute 20CVZ1 {0,1}
@attribute 02CVZ01 {0,1}
@attribute NPF {0,1}
@attribute SI {0,1}
@attribute 29CTZ4 {0,1}
@attribute 01CTZ6 {0,1}
@attribute SAFE {0,1}
@attribute R0760 {0,1}
@attribute 02CTZWIPBLADES {0,1}
@attribute MCFT {0,1}
@attribute 05CVZALIGN2 {0,1}
@attribute 01CTZ1 {0,1}
@attribute 03CTZMOUNTTIRE4 {0,1}
@attribute 25CVZ1 {0,1}
@attribute MCFS {0,1}
@attribute 03CVZMOUNTTIRE4 {0,1}
@attribute 47CTZPMA {0,1}
@attribute 22CTZ4 {0,1}
@attribute 39CTZ24 {0,1}
@attribute R4490 {0,1}
@attribute 03CVZ4 {0,1}
@attribute V1508 {0,1}
@attribute 02CVZBELTS {0,1}
@attribute 47CTZ {0,1}
@attribute V1382 {0,1}
@attribute 03CTZ3 {0,1}
@attribute 05CVZALIGN4 {0,1}
@attribute 02CTZENGTU8 {0,1}
@attribute 11CTZ4 {0,1}
@attribute N1720 {0,1}
@attribute 39CTZ23 {0,1}
@attribute 02CVZROTBAL {0,1}
@attribute D1002 {0,1}
@attribute H0122 {0,1}
@attribute 30CTZ2 {0,1}
@attribute 10CTZ2 {0,1}
@attribute SOP {0,1}
@attribute 02CVZ1 {0,1}
@attribute 05CTZ1 {0,1}
@attribute MCFC {0,1}
@attribute 02CTZATSER {0,1}
@attribute L1020 {0,1}
@attribute 46SYN {0,1}
@attribute 46OWN {0,1}
@attribute 01CTZ20 {0,1}
@attribute L2300 {0,1}
@attribute E0716 {0,1}


```
@attribute 39CVZ1 {0,1}  
@attribute Class {0,1}
```

Appendix C

WEKA Functions and Parameters

The software used for the evaluation was WEKA (Waikato Environment for Knowledge Analysis) (Witten et al., 2004) an open source machine learning workbench. WEKA has an extensive collection of pre-processing methods and machine learning algorithms implemented in java as classes with an optional graphical user interface. WEKA Version 3.7.7 was used in this study. The functions and parameters used are detailed in this section.

Missing Values

WEKA's *ReplaceMissingValues* preprocessing filter to account for missing values. This filter replaces all missing values for nominal and numeric attributes in a dataset with the modes and means from the training data.

Data Normalization

Some algorithms, such as Support Vector Machines, may require that the data be normalized to increase the efficiency of the algorithm. The normalization will prevent any variation in distance measures that may occur had the data not been normalized.

When needed, we will apply WEKA's *Normalize* filter to normalize the data using the following parameters:

Parameters for Normalize filter

Option	Description	Value
-L num	The Lnorm to be used on the normalization	2.0
-N num	The norm of the instances after normalization	1.0

Feature Selection

We began our feature selection testing by using a Correlation Based Feature selection method. This will be accomplished by using a *BestFirst* (Forward) search method on a *CfsSubsetEval* attribute evaluator in WEKA.

The second filter method to be tested was based on Information Gain. WEKA's implementation, *InfoGainAttributeEval*, evaluates the attribute's worth by measuring the information gain with respect to the class.

$$\text{InfoGain}(\text{Class}, \text{Attribute}) = H(\text{Class}) - H(\text{Class} | \text{Attribute})$$

Before using this filter method we were required to discretize continuous values beforehand.

The last filter method studied was the Relief-F method. The attribute evaluator used for this method will be *ReliefAttributeEval*. This method evaluates the worth of the attribute being tested by sampling an instance and detecting the nearest class.

Parameters for Relief method

Option	Description	Value
-M num	Number of instances to Sample	All
-D num	Seed for randomly sampling instances	1.0
-W	Weight nearest neighbors by distance	
-A num	Sigma value by which distant instances decrease. Use with – W option.	2.0

InfoGainAttributeEval and *Relief-F* attribute evaluators use a ranking search method to rank the attributes. Table 4 shows a listing of the Ranker operator options.

Parameters for Ranker operator

Option	Description	Value
-P set	Starting set of attributes to ignore	None
-T num	Threshold used to discard an attribute. Determined after first run with no threshold.	X
-N num	Number of attributes to select	All, 30, 20

Wrappers

Wrapper methods use the classifying algorithm as part of the selection process. For our experiments we will use the *WrapperSubsetEval* evaluator. This evaluator uses cross validation to estimate the accuracy of the classifying algorithm for a given set of attributes.

Parameters for Wrapper method

Option	Description	Value
-B	Class name of the base learner	Varies
-F num	Number of cross validations to use	5.0
-T num	Threshold used to initiate next cross validation (StdDev as Percentage)	.01
-E	Performance evaluation measure to use (acc,rmse,mae,f-meas,auc,auprc)	Accuracy

Decision Tree

The decision tree algorithm used in this study is implemented using WEKA's J48 decision tree classifier. J48 is WEKA's implementation of the C4.5 (Quinlan, 1993) decision tree algorithm. We'll test the J48 classifier with a confidence factor ranging from 0.1 to 1.0 incremented by 0.2. A lower confidence factor will equate to a larger error estimate at each node thus increasing the chances that the node will be pruned. The

number of minimum instances per node (minNumObj) was set at 2, and cross validation folds for the Testing Set (crossValidationFolds) was set at 10. All options for the J48 are shown below.

Decision Tree options

Option	Description	Value
binarySplits	Whether to use binary splits on nominal attributes when building the trees.	False
confidenceFactor	The confidence factor used for pruning (smaller values incur more pruning).	0.25
debug	If set to true, classifier may output additional info to the console.	False
minNumObj	The minimum number of instances per leaf.	2
numFolds	Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the tree.	3
reducedErrorPruning	Whether reduced-error pruning is used instead of C.4.5 pruning.	False
saveInstanceData	Whether to save the training data for visualization.	True
seed	The seed used for randomizing the data when reduced-error pruning is used.	1
subtreeRaising	Whether to consider the subtree raising operation when pruning.	True
unpruned	Whether pruning is performed.	T/F
useLaplace	Whether counts at leaves are smoothed based on Laplace.	False

k-Nearest Neighbor (k-NN)

For our k-NN classification testing we will use the Instance Based k (*IBk*) classifier (Kibler, 1991) in WEKA. We will test using Euclidean distance and a K value of 1. A different value for K will also be used. This value was selected by the system using the

crossValidate parameter. If this parameter is set, the system will find the optimal K value between 1 and the K value entered.

k-Nearest Neighbor parameter settings

Option	Description	Value
K-NN	The number of neighbors to use.	Varies
crossValidate	Whether hold-one-out cross-validation will be used to select the best k value.	True
debug	If set to true, classifier may output additional info to the console.	False
distanceWeighting	Gets the distance weighting method used.	Eq. Wt
meanSquared	Whether the mean squared error is used rather than mean absolute error when doing cross-validation for regression problems.	False
nearestNeighbourSearchAlgorithm	The nearest neighbor search algorithm to use (Default: weka.core.neighboursearch.LinearNNSearch).	Linear
windowSize	Gets the maximum number of instances allowed in the training pool. The addition of new instances above this value will result in old instances being removed. A value of 0 signifies no limit to the number of training instances.	0

Support Vector Machine (SVM)

The Support Vector Machine classifier used in our experiments was the WEKA LibSVM classifier (El-Manzalawy & Honavar, 2005). This SVM classifier is more efficient than WEKA's SMO and supports several SVM methods (e.g. One-Class SVM, nu-SVM, and epsilon-SVR). Values to be used in our tests are shown below.

Support Vector Machine runtime options

Option	Description	Value
SVMType	The type of SVM to use.	C-SVC
cacheSize	The cache size in MB.	80
coef0	The coefficient to use.	0
cost	The cost parameter C for C-SVC, epsilon-SVR and nu-SVR.	1
debug	If set to true, classifier may output additional info to the console.	F
degree	The degree of the kernel.	3
eps	The tolerance of the termination criterion.	.001
gamma	The gamma to use, if 0 then 1/max_index is used.	1/k
kernelType	The type of kernel to use	Radial
loss	The epsilon for the loss function in epsilon-SVR.	NA
normalize	Whether to normalize the data.	0
nu	The value of nu for nu-SVC, one-class SVM and nu-SVR.	NA
probabilityEstimates	Whether to generate probability estimates instead of -1/+1 for classification problems.	0
shrinking	Whether to use the shrinking heuristic.	1
weights	The weights to use for the classes, if empty 1 is used by default.	1

Appendix D

Bank Data Classification Results

Bank Data				
	Test	Accuracy	AUC	F-Meas
1	J48ALL	0.847	0.862	0.847
	no	0.821		0.843
	yes	0.873		0.851
2	J48IG	0.833	0.866	0.833
		0.803		0.828
		0.863		0.838
3	J48RLF	0.858	0.901	0.858
		0.834		0.855
		0.882		0.861
4	J48CFS	0.821	0.882	0.821
		0.802		0.818
		0.84		0.824
5	J48WRP	0.862	0.899	0.862
		0.839		0.859
		0.886		0.865
6	PS_J48IG	0.839	0.883	0.839
		0.815		0.835
		0.862		0.842
7	PS_J48RLF	0.86	0.904	0.859
		0.837		0.856
		0.882		0.863
8	PS_J48CFS	0.822	0.882	0.822
		0.803		0.803
		0.84		0.84
9	PS_J48WRP	0.864	0.905	0.864
		0.835		0.86
		0.894		0.868
10	1NNALL	0.735	0.801	0.735
		0.787		0.748
		0.684		0.721
11	1NNIG	0.776	0.775	0.776
		0.792		0.779
		0.759		0.772
12	1NNRLF	0.807	0.809	0.807

		0.81		0.807
		0.804		0.806
13	1NNCFS	0.761	0.758	0.761
		0.767		0.762
		0.755		0.761
14	1NNWRP	0.796	0.794	0.796
		0.806		0.798
		0.787		0.794
15	5NNALL	0.735	0.801	0.735
		0.787		0.748
		0.684		0.721
16	5NNIG	0.809	0.87	0.809
		0.835		0.835
		0.782		0.782
17	5NNRLF	0.842	0.898	0.842
		0.839		0.842
		0.845		0.842
18	5NNCFS	0.809	0.867	0.809
		0.808		0.809
		0.81		0.809
19	5NNWRP	0.849	0.905	0.849
		0.832		0.846
		0.866		0.851
20	10NNALL	0.734	0.812	0.73
		0.852		0.762
		0.615		0.698
21	10NNIG	0.798	0.879	0.797
		0.867		0.811
		0.729		0.783
22	10NNRLF	0.842	0.898	0.842
		0.839		0.842
		0.845		0.842
23	10NNCFS	0.811	0.884	0.811
		0.84		0.816
		0.782		0.805
24	10NNWRP	0.83	0.9	0.83
		0.859		0.835
		0.801		0.825
25	PS_NNALL	0.74	0.81	0.73
		0.851		0.762
		0.61		0.698

26	PS_NNIG	0.806	0.875	0.806
		0.836		0.812
		0.777		0.8
27	PS_NNRLF	0.841	0.901	0.841
		0.838		0.841
		0.845		0.842
28	PS_NNCFS	0.815	0.882	0.815
		0.815		0.815
		0.815		0.815
29	PS_NNWRP	0.835	0.889	0.835
		0.83		0.834
		0.841		0.836
30	LibSVMALL	0.785	0.785	0.784
		0.86		0.8
		0.71		0.768
31	LibSVMIG	0.814	0.814	0.813
		0.872		0.824
		0.756		0.802
32	LibSVMRLF	0.819	0.819	0.819
		0.863		0.827
		0.775		0.811
33	LibSVMCFS	0.785	0.785	0.783
		0.89		0.805
		0.68		0.76
34	LibSVMWRP	0.818	0.818	0.818
		0.854		0.825
		0.782		0.812
35	GS_LibSVMIG	0.814	0.814	0.813
		0.872		0.824
		0.756		0.802
36	GS_LibSVMRLF	0.844	0.844	0.844
		0.815		0.839
		0.872		0.848
37	GS_LibSVMCFS	0.841	0.841	0.841
		0.836		0.841
		0.847		0.842
38	GS_LibSVMWRP	0.838	0.838	0.838
		0.845		0.839
		0.831		0.837

Service Data Set Classification Results

Service Data				
	Test	Accuracy	AUC	F-Meas
1	J48ALL	0.798	0.778	0.787
	no	0.966		0.845
	yes	0.575		0.71
2	J48IG	0.778	0.75	0.765
		0.954		0.831
		0.546		0.679
3	J48RLF	0.763	0.723	0.743
		0.977		0.977
		0.48		0.48
4	J48CFS	0.779	0.749	0.765
		0.964		0.832
		0.534		0.375
5	J48WRP	0.797	0.778	0.797
		0.964		0.844
		0.577		0.71
6	PS_J48IG	0.782	0.759	0.769
		0.963		0.834
		0.542		0.682
7	PS_J48RLF	0.762	0.723	0.743
		0.976		0.824
		0.48		0.635
8	PS_J48CFS	0.779	0.749	0.765
		0.963		0.832
		0.534		0.675
9	PS_J48WRP	0.797	0.778	0.787
		0.965		0.844
		0.576		0.71
10	1NNALL	0.702	0.696	0.701
		0.759		0.744
		0.628		0.645
11	1NNIG	0.683	0.674	0.683
		0.722		0.722
		0.631		0.631
12	1NNRLF	0.763	0.74	0.742
		0.979		0.824
		0.476		0.633

13	1NNCFS	0.693	0.689	0.692
		0.741		0.733
		0.629		0.638
14	1NNWRP	0.795	0.796	0.783
		0.97		0.844
		0.564		0.703
15	5NNALL	0.749	0.775	0.741
		0.874		0.798
		0.583		0.666
16	5NNIG	0.741	0.757	0.735
		0.857		0.857
		0.587		0.587
17	5NNRLF	0.761	0.74	0.741
		0.978		0.824
		0.475		0.632
18	5NNCFS	0.744	0.76	0.738
		0.858		0.792
		0.592		0.665
19	5NNWRP	0.791	0.795	0.778
		0.97		0.841
		0.553		0.695
20	10NNALL	0.762	0.786	0.749
		0.933		0.817
		0.535		0.749
21	10NNIG	0.771	0.764	0.759
		0.94		0.824
		0.548		0.673
22	10NNRLF	0.759	0.74	0.738
		0.976		0.822
		0.472		0.628
23	10NNCFS	0.772	0.768	0.761
		0.936		0.824
		0.556		0.677
24	10NNWRP	0.787	0.794	0.773
		0.971		0.839
		0.543		0.687
25	PS_NNALL	0.762	0.785	0.749
		0.933		0.817
		0.535		0.659
26	PS_NNIG	0.771	0.764	0.759
		0.94		0.824

		0.548		0.673
27	PS_NNRLF	0.762	0.74	0.742
		0.979		0.824
		0.476		0.633
28	PS_NNCFS	0.772	0.768	0.761
		0.936		0.824
		0.556		0.677
29	PS_NNWRP	0.795	0.796	0.783
		0.97		0.844
		0.564		0.703
30	LibSVMALL	0.766	0.733	0.748
		0.973		0.826
		0.493		0.645
31	LibSVMIG	0.764	0.73	0.745
		0.972		0.824
		0.489		0.641
32	LibSVMRLF	0.763	0.729	0.743
		0.977		0.825
		0.48		0.636
33	LibSVMCFS	0.761	0.727	0.742
		0.974		0.823
		0.48		0.634
34	LibSVMWRP	0.793	0.765	0.781
		0.964		0.841
		0.567		0.702
35	GS_LibSVMIG	0.764	0.726	0.745
		0.972		0.824
		0.489		0.641
36	GS_LibSVMRLF	0.763	0.729	0.743
		0.977		0.825
		0.48		0.636
37	GS_LibSVMCFS	0.759	0.724	0.739
		0.976		0.822
		0.473		0.628
38	GS_LibSVMWRP	0.798	0.771	0.787
		0.965		0.845
		0.576		0.711

Reference List

- Acuna, E., & Rodriguez, C. (2004). The treatment of missing values and its effect on classifier accuracy. *Classification, Clustering, and Data Mining Applications*, pp. 639–647.
- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, I. (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 307–328. Menlo Park, Calif.: AAAI Press.
- Anunciacao, O., Gomes, B., Vinga, S., Gaspar, J., Oliveira, A. & Rueff, J. (2010). A data mining approach for the detection of high-risk breast cancer groups, *Advances in Bioinformatics*, pp. 43–51, Springer. <http://kdbio.inesc-id.pt/~orlando/publications/10-agvgor-iwpacbb.pdf>
- Anyanwu, M. N., & Shiva, S. G. (2009). Comparative analysis of serial decision tree classification algorithms. *International Journal of Computer Science and Security*, 3(3), pp. 230–240.
- Becchetti, L., Castillo, C., Donato, D., Leonardi, S. & Baeza-Yates, R. (2009). Web spam detection: Link-based and content-based techniques, *The European Integrated Project Dynamically Evolving, Large Scale Information Systems (DELIS): proceedings of the final workshop*, 222, pp. 99–113. http://www.chato.cl/papers/becchetti_2008_link_spam_techniques.pdf
- Bermejo, P., de la Ossa, L., Gámez, J. A., & Puerta, J. M. (2012). Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking. *Knowledge-Based Systems*, 25(1), pp. 35–44.
- Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations Newsletter*, 2(2), pp. 1–13.
- Bhatia, N. & Vendana (2010). Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8(2), pp. 302–305. <http://arxiv.org/ftp/arxiv/papers/1007/1007.0085.pdf>
- Blum, A. L. & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97, pp. 245–271. <http://www.cin.ufpe.br/~dclv/artigos/dm/blum-selection.pdf>
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30, pp. 1145–1159.
- Bramer, M. (2007). Principles of data mining. Springer.

- Brown, M.L. & Kros, J.F. (2003). The impact of missing data on data mining, *Data mining: opportunities and challenges*, 1, pp. 174–198, IRM Press.
- Breiman, L., Friedman, J., Olshen, L & Stone, J. (1984). Classification and regression trees. Wadsworth Statistics/Probability series. CRC press Boca Raton, Florida, USA.
- Brekke, C., & Solberg, A. H. (2008). Classifiers and confidence estimation for oil spill detection in ENVISAT ASAR images. *Geoscience and Remote Sensing Letters*, IEEE, 5(1), pp. 65–69.
- Boser, B. E., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers, *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152. ACM Press, <http://doi.acm.org/10.1145/130385.130401>, doi = 10.1145/130385.130401
- Carrizosa, E., Martin-Barragan, B., & Morales, D. R. (2010). Binarized support vector machines. *INFORMS Journal on Computing*, 22(1), pp. 154–167. <http://search.proquest.com.ezproxylocal.library.nova.edu/docview/200527413?accountid=6579>
- Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), pp. 27–28.
- Chang, C., Verhaegen, P. A., & Duflou, J. R. (2014). A comparison of classifiers for intelligent machine usage prediction. *Intelligent Environments (IE)*, pp. 198–201.
- Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1), pp. 131–159. <http://research.microsoft.com/en-us/um/people/manik/projects/trade-off/papers/ChapelleML02.pdf>
- Chawla, N. V. (2005). Data mining for imbalanced datasets: An overview. *In Data mining and knowledge discovery handbook*, pp. 853–867. Springer US.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, pp. 321–357.
- Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), pp. 866–883.
- Chen, X., Wang, M., & Zhang, H. (2011). The use of classification trees for bioinformatics, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), pp. 55–63, Wiley Online Library. <http://moult.ibbr.umd.edu/JournalClubPresentations/Maya/Maya-04Feb2011-paper.pdf>

- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), pp. 273–297. http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf
- Cover, T.M. & Hart, P.E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), pp. 21–27. doi:10.1109/TIT.1967.1053964
- Cunningham, P. & Delany, S.J. (2007). k-Nearest neighbour classifiers. *Multiple Classifier Systems*, pp. 1–17. <http://www.csi.ucd.ie/UserFiles/publications/UCD-CSE-2007-4.pdf>
- Dash, M. & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis: An International Journal*, 1(3), pp. 131–156.
- Devijver, P. A. & Kittler, J. (1982). *Pattern Recognition: A Statistical Approach*. New York: Prentice-Hall.
- Dougherty, E. R., Hua, J., & Sima, C. (2009). Performance of feature selection methods. *Current Genomics*, 10(6), p. 365.
- Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *In Proc. Twelfth International Conference on Machine Learning*. Los Altos, CA: Morgan Kaufmann, pp. 194–202.
- Drummond, C., & Holte, R. C. (2005). Severe class imbalance: Why better algorithms aren't the answer. *In Machine Learning: ECML 2005*, pp. 539–546. Springer Berlin Heidelberg. <http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=9190916>
- Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, (4), pp. 325–327.
- Farhangfar, A., Kurgan, L., & Dy, J. (2008). Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12), pp. 3692–3705. <http://sci2s.ugr.es/MVDM/pdf/2008-Farhangfar-PR.pdf>
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), pp. 861–874.
- Fayyad, U. M. (1996). Data mining and knowledge discovery: Making sense out of data. *IEEE Intelligent Systems*, (5), pp. 20–25.
- Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery: An overview, in U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Ulthurusamy (eds.) *Advances in Knowledge Discovery and Data Mining*, Cambridge, MA: MIT Press, pp. 1–34.

- Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
- Frank, E. (2000). Machine Learning Techniques for Data Mining. Lecture Notes. Eibe Frank, University of Waikato, New Zealand, 10, p. 25.
- Ge, G. & Wong, G.W. (2008). Classification of premalignant pancreatic cancer mass-spectrometry data using decision tree ensembles, *BMC bioinformatics*, 9(1), pp. 275–284, BioMed Central Ltd. doi:10.1186/1471-2105-9-275
- Gheyas, I. A. & Smith, L. S. (2010). Feature subset selection in large dimensionality domains, *Pattern Recognition*, 43(1), pp. 5–13, ISSN 0031-3203, 10.1016/j.patcog.2009.06.009. <http://www.sciencedirect.com/science/article/pii/S0031320309002520>
- Gu, Q., Cai, Z, Zhu, L., & Huang, B. (2008). Data Mining on Imbalanced Data Sets. *Proceedings of International Conference on Advanced Computer Theory and Engineering*, Phuket, pp. 1020–1024.
- Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3, pp. 1157–1182.
- Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, pp. 1157–1182. <http://dl.acm.org/citation.cfm?id=944919.944968>
- Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning, *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 359–366.
- Hall M. A. & Holmes G. (2003). Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, *IEEE Transactions on Knowledge and Data Engineering*, 15(6), pp. 1437–1447. <http://ourmine.googlecode.com/svn/trunk/share/pdf/03hall.pdf>
- Hall, M. A. & Smith, L. A. (1998). Practical feature subset selection for machine learning. *Proceedings of the 21st Australian Computer Science Conference*. <http://researchcommons.waikato.ac.nz/bitstream/handle/10289/1512/Practical%20feature%20subset?sequence=1>
- Hamerly, G. & Speegle, G. (2010). Efficient Model Selection for Large-Scale Nearest-Neighbor Data Mining. *Proceedings of the 2010 British National Conference on Databases (BNCOD 2010)*.
- Han, E.H., Karypis, G., & Kumar, V. (2001). Text categorization using weight adjusted k-nearest neighbor classification, *Advances in Knowledge Discovery and Data Mining*, pp. 53–65. <http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA439688>

- Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*, 2nd ed. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers. pp. 336–337. ISBN 1-55860-901-6
- Hassan, M., Hossain, M., Bailey, J. & Ramamohanarao, K. (2008). Improving k-nearest neighbour classification with distance functions based on receiver operating characteristics. *Machine Learning and Knowledge Discovery in Databases*, pp. 489–504.
- Herbrich, R. (2001). *Learning kernel classifiers: theory and algorithms*. MIT Press.
- Hsu, C. W., Chang, C. C., & Lin, C. J. (2009). A practical guide to support vector classification. Department of Computer Science, National Taiwan University. <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
- Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3), pp. 299-310.
- Irani, D., Webb, S., Pu, C. & Li, K. (2010). Study of trend-stuffing on twitter through text classification, Collaboration, Electronic messaging, *Anti-Abuse and Spam Conference (CEAS)*. http://www.cc.gatech.edu/~danesh/download/DIrani_CEAS_2010.pdf
- John, G. H., Kohavi, R., & Pflieger, K. (1994) Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 121–129. New Brunswick, NJ. Morgan Kaufmann <http://machine-learning.martinsewell.com/feature-selection/JohnKohaviPflieger1994.pdf>
- Karegowda, A. G., Manjunath, A. S., & Jayaram, M. A. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2), pp. 271–277.
- Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2), pp. 119–127.
- Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7), pp. 1667–1689.
- Kira, K. & Rendell, L. (1992). A practical approach to feature selection. *Proceedings of the Ninth International Conference on Machine Learning*. Aberdeen Scotland. Morgan Kaufmann, pp. 249–256.
- Kohavi, R. & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97, pp. 273–324. https://intranet.cs.aau.dk/fileadmin/user_upload/Education/Courses/2010/DWML/papers/kohavi-john-wrappers.pdf

- Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. *Proceedings of European Conference on Machine Learning (ECML) '94*, pp. 171–182. Springer-Verlag New York, Inc.
- Kordos, M., Blachnik, M., and Strzempa, D. (2010). Do we need whatever more than K-NN?, *Artificial Intelligence and Soft Computing*, pp. 414–421, Springer. <http://www.kordos.com/pdf/K-NN.pdf>
- Kubat, M., Holte, R., & Matwin, S. (1998). Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30, pp. 195–215.
- Kudo, M. & Sklansky, J. (1998). Classifier-independent feature selection for two-stage feature selection, *Lecture Notes in Computer Science*, Springer, Berlin (Advances in Pattern Recognition 1998), 1451, pp. 548–554.
- Kuo, F. Y., & Sloan, I. H. (2005). Lifting the curse of dimensionality. *Notices of the AMS*, 52(11), pp. 1320–1328.
- Ladha, L., & Deepa, T. (2011). Feature selection methods and algorithms. *International Journal on Computer Science and Engineering*, 3(5), pp. 1787–1797.
- Lam, H. K., Ekong, U., Liu, H., Xiao, B., Araujo, H., Ling, S. H., & Chan, K. Y. (2014). A study of neural-network-based classifiers for material classification. *Neurocomputing*, 144, pp. 367–377.
- Lewis, D. & Gale, W. (1994) Training text classifiers by uncertainty sampling. *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Li, T., Zhang, C., & Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15), pp. 2429–2437.
- Li, Y., & Lu, B. L. (2009). Feature selection based on loss-margin of nearest neighbor classification. *Pattern Recognition*, 42(9), pp. 1914–1921. http://bcmi.sjtu.edu.cn/~blu/papers/2009/YunLi_Pattern-Recognition_2009.pdf
- Lichman, M. (2013). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.
- Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering, *IEEE Transactions on Knowledge and Data Engineering*, 17(4), pp. 491–592. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.8392&rep=rep1&type=pdf>

- Lin, H. T., & Lin, C. J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *Neural Computation*, pp. 1–32.
<http://home.caltech.edu/~htlin/publication/doc/tanh.pdf>
- López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250, pp. 113–141.
- Maldonado, S. & Weber, R. (2009). A wrapper method for feature selection using Support Vector Machines, *Information Sciences: an International Journal*, 179(13), pp. 2208–2217.
- Mehta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. *Advances in Database Technology (EDBT 96)*, Avignon, France
- Mendenhall, W., Beaver, R. & Beaver, B., (1996). A Course in Business Statistics. fourth edition, ITP.
- Major, J.A. & Riedinger, D. A. (1992). EFD—A hybrid knowledge statistical based system for the detection of fraud, *International Journal of Intelligent Systems*, 7(1), pp. 687–703.
- Moro, R. L., Cortez, P., & Rita, P. (2011). Bank Marketing Data Set. Retrieved November 10, 2012, from UC Irvine Machine Learning Repository:
<http://archive.ics.uci.edu/ml/datasets/Bank+Marketing>
- Ngai, E.W.T., Xiu, L., & Chau, DCK (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications*, 36(2), pp. 2592–2602.
- Otero, F. E., Freitas, A. A., & Johnson, C. G. (2012). Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, 12 (11), pp. 3615–3626.
<http://www.cs.kent.ac.uk/people/staff/febo/papers/otero-asoc-2012.pdf>
- Palaniappan, S. & Awang, R. (2008). Intelligent heart disease prediction system using data mining techniques., *IEEE/ACS International Conference on Computer Systems and Applications, 2008. AICCSA 2008*, pp.108–115, March 31 2008-April 4 2008 doi: 10.1109/AICCSA.2008.4493524 URL:
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4493524&isnumber=4493499>
- Powell, W. B. (2007). Approximate dynamic programming: Solving the curses of dimensionality, Wiley-Interscience, 1st ed.
- Powers, David M W (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 2(1), pp. 37–63.

- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann.
- Quinlan, J.R. (1987). Simplifying decision trees. *International Journal Man-Mach. Studies*, 27, pp. 221–234.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), pp. 81–106.
- Rich, E., & Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill, Incorporated.
- Rokach, L. & Maimon, O. (2005). Top–Down Induction of Decision Trees Classifiers—A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on.*, 35(4), pp. 476–487.
- Saeys, Y., Inza, I. & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics, *Bioinformatics*, 23(19), pp. 2507–2517.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27, pp. 379–423, pp. 623–56.
- Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of Data Warehousing*, 5(4), pp. 13–22.
- Shmueli, G., Patel, N. R., & Bruce, P. C. (2011). *Data mining for business intelligence: Concepts, techniques, and applications in Microsoft Office Excel with XLMiner*. Wiley, pp. 23–24.
- Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1), pp. 185–197.
- Tsang, S., Kao, B., Yip, K.Y., Ho, W.S., & Lee, S.D. (2011). Decision trees for uncertain data, *IEEE Transactions on Knowledge and Data Engineering*, 23(1), pp. 64–78. http://www.sundaychennai.com/java_ieee2011/Decision%20Trees%20for%20Uncertain%20Data.pdf
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.
- Wang, Y, & Makedon, F. (2004). Application of Relief-F feature filtering algorithm to selecting informative genes for cancer classification using microarray data. *Proceedings of the IEEE Conference on Computational Systems Bioinformatics (CSB'04)*, IEEE Press, Stanford, CA, USA, pp. 497–498.
- Watson, H.J., & Wixom, B.H. (2009). The Current State of Business Intelligence, *Computer*, 40(9), pp. 96–99, Sept doi: 10.1109/MC.2007.331 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4302625&isnumber=4302594>

- Wiiten, I. H., Frank, E., & Hall, M. A. (2011), *Data mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 3rd Edition, San Francisco.
- Wu, X., Kumar, V., Ross, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z., Steinbach, M., Hand, D., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1), pp. 1–37, DOI: 10.1007/s10115-007-0114-2, Springer London.
- Xiaoliang, Z., Jian, W., Hongcan, Y., & Shangzhuo, W. (2009). Research and application of the improved algorithm C4. 5 on Decision tree. *In Test and Measurement*, 2009. ICTM'09. International Conference on, 2, pp. 184–187.
- Yang, Y. & Liu, X. (1999). A Re-examination of Text Categorization Methods. *In Proceedings of the Twenty-Second International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 42–49.
- Yu, L. & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution, *International Conference on Machine Learning (ICML)*, 20(2), pp. 856–894.
- Zweig, M. H. & Campbell, G. (1993). Receiver-operating characteristic (ROC) plots. *Clinical Chemistry*, 29, pp. 561–577.

Bibliography

- Aha, D., Kibler, D. (1991). Instance-based learning algorithms. *Journal of Machine Learning Research*. 6, pp. 37–67.
- Albisua, I. Arbelaitz, O., Gurrutxaga, I., Martín, J. I., Mugerza, J., Pérez, J.M., & Perona, I. (2010). Obtaining optimal class distribution for decision trees: Comparative analysis of CTC and C4.5. *Current Topics in Artificial Intelligence Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 5988, 101–110, Url: http://dx.doi.org/10.1007/978-3-642-14264-2_11 Doi: 10.1007/978-3-642-14264-2_11
- AL-Nabi, D. L. A., & Ahmed, S. S. (2013). Survey on Classification Algorithms for Data Mining:(Comparison and Evaluation). *Computer Engineering and Intelligent Systems*, 4(8), pp. 18–24.
- Bhargavi, P., & Jyothi, S. (2009). Applying Naïve Bayes Data Mining Technique for Classification of Agricultural Land Soils. *International Journal of Computer Science and Network Security*, 9 (8), pp. 117–122.
- Bi, J., Bennett, K., Embrechts, M., Breneman, C., & Song, M.(2003). Dimensionality Reduction via Sparse Support Vector Machines; *Journal of Machine Learning Research*, 3, pp. 1229–1243.
<http://jmlr.csail.mit.edu/papers/volume3/bi03a/bi03a.pdf>
- Bichler, M. & Kiss, C. (2004). Comparing classification methods for campaign management: A comparison of logistic regression, k-nearest neighbour, and decision tree induction, *Proceedings of the Tenth Americas Conference on Information Systems*, New York, New York, August 2004.
- Borges, L. C., Marques, V. M., & Bernardino, J. (2013). Comparison of data mining techniques and tools for data classification. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pp. 113–116. ACM.
- Caruana, R. & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning (ICML '06)*. ACM, New York, NY, USA, pp. 161–168. DOI=10.1145/1143844.1143865 <http://doi.acm.org/10.1145/1143844.1143865>
- Chang, T. (2011). A comparative study of artificial neural networks, and decision trees for digital game content stocks price prediction, *Expert Systems with Applications*, 38(12), pp. 14846–14851, ISSN 0957-4174, 10.1016/j.eswa.2011.05.063.
<http://www.sciencedirect.com/science/article/pii/S0957417411008475>
- Chen, X. & Wasikowski, M.. (2008). FAST: A ROC-based feature selection metric for small samples and imbalanced data classification problems. *Proceedings of the*

14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, pp. 124–132.
DOI=10.1145/1401890.1401910 <http://doi.acm.org/10.1145/1401890.1401910>

- Cristianini, N., & Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines*. Cambridge: Cambridge University Press.
- Coenen, F. (2011). Data mining: Past, present and future. *The Knowledge Engineering Review*, 26(1), pp. 25–29. doi:10.1017/S0269888910000378
- Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection, *Machine Learning-International Workshop Then Conference*, pp. 74–81
<http://www.cs.rpi.edu/~sanmay/papers/icml01.pdf>
- Delen, D., Cogdell, D., & Kasap, N. (2012). A comparative analysis of data mining methods in predicting NCAA bowl outcomes. *International Journal of Forecasting*, 28(2), pp. 543–552.
- Dorre, J., Gerstl, P., & Seiffert, R..(1999). Text mining: finding nuggets in mountains of textual data. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '99). ACM, New York, NY, USA, pp. 398–401. DOI=10.1145/312129.312299
<http://doi.acm.org/10.1145/312129.312299>
- EL-Manzalawy, Y. & Honavar, V. (2005). WLSVM : Integrating LibSVM into WEKA Environment. Software available at <http://www.cs.iastate.edu/~yasser/wlsvm>
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27, pp. 861–874.
- Ferri, C., Hernandez-Orallo, J., & Modroi, R. (2009). An experimental comparison of performance measures for classification, *Pattern Recognition Letters*,30(1), pp. 27–38. <http://users.dsic.upv.es/~flip/papers/PRL2009.pdf>
- Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, pp. 1289–1305. url = {<http://dl.acm.org/citation.cfm?id=944919.944974>},
- Forman, G. (2002). Choose Your Words Carefully: An Empirical Study of Feature Selection Metrics for Text Classification. *Proceedings of Principles of Data Mining and Knowledge Discovery* (PKDD'02), pp.150-162. Url: http://dx.doi.org/10.1007/3-540-45681-3_13 Doi: 10.1007/3-540-45681-3_13
- Gao, Z., Xu, Y., Meng, F., Qi, F., & Lin, Z. (2014). Improved information gain-based feature selection for text categorization. *International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems* (VITAE), pp. 1–5.

- Goua, J., Dub, L., Zhanga, Y., & Xionga, T. (2012). A New Distance-weighted k-nearest Neighbor Classifier. *Journal of Information & Computational Science*, 9(6), pp. 1429–1436. http://www.joics.com/publishedpapers/2012_9_6_1429_1436.pdf
- Hall P, Park B.U., & Samworth, R.J. (2008). "Choice of neighbor order in nearest-neighbor classification". *Annals of Statistics*, 36 (5), pp. 2135–2152. doi:10.1214/07-AOS537
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009); The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, 11(1).
- Hand, D.J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, 77, pp. 103–123.
- Hand, D. J. (2007), Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner by Galit Shmueli, Nitin R. Patel, Peter C. Bruce. *International Statistical Review*, 75: 256. doi: 10.1111/j.1751-5823.2007.00015_9.x
- Hua, J., Tembe, W. D., & Dougherty, E. R. (2009). Performance of feature-selection methods in the classification of high-dimension data, *Pattern Recognition*, 42(3), pp. 409–424, doi = 10.1016/j.patcog.2008.08.001. <http://www.sciencedirect.com/science/article/pii/S0031320308003142>
- Jamain, A. & Hand, D.J. (2009). Where are the large and difficult data sets? *Advances in Data Analysis and Classification*, 3(1), pp. 25–38. <http://dx.doi.org/10.1007/s11634-009-0037-8>
- Janecek, A.G.K. & Gansterer, W.N. and Demel, M. and Ecker, G.F. (2008). On the relationship between feature selection and classification accuracy, *Journal of Machine Learning Research: Workshop and Conference Proceedings*, 4, pp. 90–105.
- Kesavaraj, G., & Sukumaran, S. (2013). A study on classification techniques in data mining. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)* pp. 1-7.
- Kim, H., Claffy, K.C., Fomenkov, M., Barman, D., Faloutsos, M., & Lee, K.Y. (2008). Internet traffic classification demystified: myths, caveats, and the best practices, *Proceedings of the 2008 ACM CoNEXT Conference*, pp. 11–22. http://alumni.cs.ucr.edu/~dhiman/papers/conext_08.pdf
- Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1137–1145.
- Koller, D. & Sahami, M. (1996). Towards optimal feature selection. *International Conference on Machine Learning (ICML)*, pp. 284–292.

- Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. *Proceedings of European Conference on Machine Learning (ECML) '94*, pp. 171–182. Springer-Verlag New York, Inc.
- Kubat, M. & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One Sided Selection. *Proceedings of the Fourteenth International Conference on Machine Learning*, pp 179–186, Nashville, Tennessee. Morgan Kaufmann.
- Liu, H. & Motoda, H. (1998). Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Norwell, MA, USA.
- Liu, H., Motoda, H., and Yu, L. (2002). Feature selection with selective sampling, *Machine Learning – International Workshop Then Conference*, pp. 395–402. <http://www.cs.binghamton.edu/~lyu/publications/Liu-etal02ICML.pdf>
- Liu, H., Dougherty, E.R., Dy, J.G., Torkkola, K., Tuv, E., Peng, H., Ding, C., Long, F., Berens, M., Parsons, L., Zhao, Z., Yu, L., & Forman, G. (2005). Evolving feature selection. *Intelligent Systems, IEEE*, 20(6), pp. 64–76. doi: 10.1109/MIS.2005.105
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1556517&isnumber=33103>
- Major, J.A. & Riedinger, D. A. (1992). EFD—A hybrid knowledge statistical based system for the detection of fraud, *International Journal of Intelligent Systems*, 7(1), pp. 687–703.
- Mason, S. J. & Graham, N. E. (2002). "Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation". *Quarterly Journal of the Royal Meteorological Society*, 128, pp. 2145–2166.
http://reia.inmet.gov.br/documentos/cursosI_INMET_IRI/Climate_Information_Course/References/Mason+Graham_2002.pdf.
- Michie, E. D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine Learning, Neural and Statistical Classification. (D. Michie, D. J. Spiegelhalter, & C. C. Taylor, Eds.) *Technometrics*, 37(4), 459. Ellis Horwood. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.3615>
- Musicant, D., Kumar, V., & Ozgur, A. (2003). Optimizing F-Measure with Support Vector Machines. American Association for Artificial Intelligence.
- Peterson, W.W. & Birdsall, T. G. (1953). The theory of signal detectability: Part I. The general theory. Electronic Defense Group, Technical Report 13, June 1953. Available from EECS Systems Of University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122 USA

- Provost, F., Fawcett, T. & Kohavi, R (1998). The case against accuracy estimation for comparing induction algorithms, *Proceedings of the Fifteenth International Conference on Machine Learning*, 445.
<http://eecs.wsu.edu/~holder/courses/cse6363/spr04/pubs/Provost98.pdf>
- Radovanovi, M., Nanopoulos, A., & Ivanovi, M. (2009). Nearest neighbors in high-dimensional data: The emergence and influence of hubs, *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 865—872.
- Reinartz, T.P., & Wirth, R. (1995). The need for a task model for knowledge discovery in databases. in: Kodratoff, Y., Nakhaiezhadeh, G., & Taylor, C. (eds.) Workshop notes Statistics, *Machine Learning, and Knowledge Discovery in Databases*. MLNet Familiarisation Workshop, Heraklion, Crete, pp. 19–24.
- Santoro, D.M., Nicoletti, M.C., & Hruschka, E.R. (2007). C-Focus-3: a C-Focus with a New Heuristic Search Strategy, *International Conference on Intelligent Systems Design and Applications*, 10, pp. 479–484.
- Santoro, D.M., Hruschka, E.R., Jr., & do Carmo Nicoletti, M. (2005). Selecting feature subsets for inducing classifiers using a committee of heterogeneous methods, *IEEE International Conference on Systems, Man and Cybernetics*, 1, pp. 375–380, doi: 10.1109/ICSMC.2005.1571175 URL:
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1571175&isnumber=33254>
- Shahid, S. & Manarvi, I. (2009). A methodology of predicting automotive sales trends through data mining, *International Conference on Computers & Industrial Engineering* (CIE 2009). 6-9 July 2009, pp. 1464–1469.
- Shreve, J., Schneider, H., & Soysal, O. (2011). A methodology for comparing classification methods through the assessment of model stability and validity in variable selection, *Decision Support Systems*, 52(1), pp. 247–257, ISSN 0167-9236, 10.1016/j.dss.2011.08.001.
<http://www.sciencedirect.com/science/article/pii/S016792361100128X>
- Spackman, Kent A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning. *Proceedings of the Sixth International Workshop on Machine Learning*. San Mateo, CA: Morgan Kaufmann., pp. 160–163.
- Szafranski, K., Megraw, M., Reczko, M., & Hatzigeorgiou, G. H. (2006). Support vector machine for predicting microRNA hairpins. In Proc. *The 2006 International Conference on Bioinformatics and Computational Biology* (pp. 270-276).
<http://ww1.ucmss.com/books/LFS/CSREA2006/BIC4985.pdf>
- Wettschereck, D., Aha, D.W. & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, *Artificial Intelligence Review*, 11(1), pp. 273–314.

- Wu, J., Cai, Z., & Gao, Z. (2010, August). Dynamic K-Nearest-Neighbor with Distance and attribute weighted for classification. *International Conference On Electronics and Information Engineering (ICEIE)*, 1, pp. 356–360.
- Yang, C., Chuang, L., Chen, Y., & Yang, C. (2008). Feature selection using memetic algorithms, *Third International Conference on Convergence and Hybrid Information Technology, 2008. ICCIT '08.*, 1, pp. 416–423, doi: 10.1109/ICCIT.2008.81
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4682062&isnumber=4681984>
- Yu, H., Huang, X., Hu, X., & Cai, H. (2010). A comparative study on data mining algorithms for individual credit risk evaluation. *2010 Fourth International Conference on Management of e-Commerce and e-Government (ICMeCG)*, pp.35–38, 23-24 Oct. 2010 doi: 10.1109/ICMeCG.2010.16
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5628627&isnumber=5628559>