Improving Prediction Accuracy Using
Class-specific Ensemble Feature Selection

by

Caio Vinicius Soares

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 13, 2010

Keywords: Machine Learning, Data Mining, Feature Selection,
Ensemble Machines, Classification

Approved by

Juan E. Gilbert, Chair, Professor of Computer Science and Software Engineering
Cheryl Seals, Associate Professor of Computer Science and Software Engineering
Gerry Dozier, Chair, Professor of Computer Science and Software Engineering

Abstract

As data accumulates at a speed significantly faster than can be processed, data preprocessing techniques such as feature selection become increasingly important and beneficial. Moreover, given the well-known gains of feature selection, any further improvements can positively affect a wide array of fields and applications. So, this research explores a novel feature selection architecture, Class-specific Ensemble Feature Selection (CEFS), which finds class-specific subsets of features optimal to each available classification in the dataset. Each subset is then combined with a classifier to create an ensemble feature selection model which is further used to predict unseen instances. CEFS attempts to provide the diversity and base classifier disagreement sought after in effective ensemble models by providing highly useful, yet highly exclusive feature subsets. CEFS is not a feature selection algorithm, but rather, a unique way of performing feature selection. Hence, it is also algorithm independent, suggesting that various machine learners and feature selection algorithms can benefit from the use of this architecture. To test this architecture, a comprehensive experiment is conducted, implementing the architecture under two different classifiers, three different feature selection algorithms, and under ten different datasets. The results of this experiment shows that the CEFS architecture outperforms the traditional feature selection architecture in every algorithmic combination and for every dataset. Moreover, the presence of high-dimensional datasets suggests that CEFS will scale up. Finally, the feature results obtained from the experiment suggest that vital class-specific information can be lost if feature selection is performed on the entire dataset as a whole, as opposed to a class-specific manner.

Acknowledgements

The author thanks the following:

The Almighty Lord for providing patience, wisdom, knowledge, and for placing such amazing people in my path.

My wife Lacey for her unyielding love, support, and endless encouragement. For being so sefless and for doing everything possible to help, every step of the way.

Dr. Juan Gilbert for the constant push and drive to finish, and to excel in all I do. Also, for caring enough to always check in and offer help, by whatever means necessary. This never would have gotten finished without your help.

My amazing family (Yara, George, Bruna, Junior, Lysandra, and Duda), for being the greatest family a person could ask for and for always making me believe I could do anything I set my mind to.

Winard Britt for his insightful ideas and great help, despite the long discussions.

The Southern Regional Education Board (SREB) and the State of Alabama for helping to fund the last three years of my life. The Doctoral Fellowship was the biggest blessing I could hope for and it helped in ways which can't be described.

And to so many others who helped along the way, thank you!!!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As data acquisition has become increasingly easier and data storage has become increasingly inexpensive, the usage and storage of exceedingly larger databases has become widespread. This has caused a dramatic shift in the mindset of data storage and acquisition; a shift which considered a one-megabyte database as being very large in 1989 to numerous multi-terabyte databases being mined regularly by the year 2000 [1]. In fact, approximately 15 petabytes of new data are created every day worldwide, more than eight times the amount of data contained in all U.S. libraries [2]. As a result, many routine information technology problems such as inventory management and systems automation have given way to multi-dimensional problems of analysis such as genomic research, deciphering global trade patterns from foreign exchange flows, or predicting disease states [2] [3].

Unfortunately, data processing has not kept up with data acquisition. That is, as computer and database technologies continue to improve, data accumulates in a speed significantly faster than the capacity of data processing [4]. Such data accumulation produces significantly large databases. These databases can be large due to a high number of instances[1], a high number of features[2], or a combination of both. An example of a high-instance database is Google index, which reported having indexed one billion unique URLs by 2000 and one trillion

[1] For the remainder of this work, instances may also be referred to as "cases", "vectors", "samples", or "observations".

[2] For the remainder of this work, features may also be referred to as "variables", "dimensions", "attributes", or "genes" (in case of microarray data).

by 2008 [5]. On the other hand, microarray datasets present a fitting example of high-dimensional datasets, often containing a small number of instances but each with thousands of attributes [6]. This research will focus solely on the issue of high-dimensional data as it pertains to classification tasks within supervised learning. More specifically, it will do so through the use of *feature selection*, a technique which can reduce the cost of recognition and often provide better classification accuracy by *reducing* the number of features [7].

Traditionally, feature selection has been carried out on the entire dataset at once, assuming that the most salient features for learning will be uniform among all instances, regardless of classification (i.e. class label). Research recently conducted in [8], however, has shown that classification accuracy can be improved when feature selection is conducted in a class-specific manner and organized in an ensemble machine for final classification. The research further states that the potential for improvement rests largely on how the ensemble's modules are recombined when they face disagreement [8]. Based on the aforementioned research, the hypothesis explored in this research contains four parts:

1. There exists a recombination technique such that, when applied to a Class-specific Ensemble Feature Selection (CEFS) architecture, it will allow the system to outperform a traditional feature selection architecture in terms of prediction accuracy.

2. The same CEFS architecture will scale up by continuing to outperform the traditional feature selection architecture when the number of features are increased, as well as, when the number of class labels are increased.

3. The union of the set of relevant[3] class-specific features may not be inclusive of the set of relevant features found traditionally for the entire dataset.

4. Conversely, the set of relevant features found traditionally for the entire dataset may not be inclusive to the union of the set of relevant features for each class label.

[3] The formal definition of *relevance* as it pertains to features will be detailed in Section 3.6 of this work.

Although all parts of the hypothesis revolve around exploring the notion of a class-specific feature selection architecture, each part will provide useful insight into the benefits afforded by such a system. Parts 1 and 2 of the hypothesis will explore the realization and development of a more accurate and scalable learning system through CEFS. Parts 3 and 4 will investigate the potential information that may be gained by CEFS, and conversely, information which may be missed or lost through the use of a traditional feature selection architecture.

Moreover, the combination of class-specific optimal features and an ensemble machine will allow the system to take advantage of the well documented benefits of ensemble methods [9] [10] [11] by providing the data diversity and classifier disagreement necessary for ensemble systems to succeed, thereby increasing prediction performance (better accuracy). Allowing each classification to have its own set of class-specific optimal features, and thus each its own ensemble classifier, will create an ensemble system that should allow for more accurate classification of unseen instances. As an added benefit, new information may be learned as to the relevance of certain features on certain classifications, information that may have been otherwise unknown due to performing feature selection on all instances, and thus, all classifications at once. This would be of particularly good use in problems such as cancer prediction [12] [13], where certain features may be optimal in detecting certain types of cancer, but irrelevant in other types.

1.1 Motivation

According to [14], a learning algorithm is good if it produces hypotheses that do a good job of predicting classifications of unseen examples. With that in mind, the primary goal of this research is to increase prediction accuracy, thereby creating a *better* learning algorithm. Note,

however, that the focus here is not on the learning algorithm itself, but rather on the pre-processing methodology used to prepare the training data for optimal learning. The first reason is because the architecture presented here is algorithm independent and can be combined with just about any learning algorithm. The second reason is research suggests that much of the power in classification comes not from the specific learning method, but from proper formulation of the problems and crafting the representation to make learning tractable [15]. In addition, the same research suggests that performance is roughly equivalent between various learning algorithms, across many domains. In other words, the preparation and representation of the data is of the utmost importance given that no single learning algorithm outperforms all others on every domain. In a nutshell, this represents the well-known theorems of "No Free Lunch" [16].

Although a dataset may be well prepared and represented, it may also fall victim to Belman's "curse of dimensionality", a problem which often plagues high-dimensional datasets. The curse of dimensionality refers to the exponential growth of hypervolume as a function of dimensionality [17]. That is, the more dimensions in a dataset, the more representation needed by the predicting model. For classification, this can mean that there are not enough data objects to allow the creation of a model that reliably assigns a class to all possible objects [18]. Moreover, when the set of features in the data is sufficiently large, many induction algorithms become simply intractable [19]. In the context of Neural Networks (NN), for example, the curse of dimensionality causes networks with irrelevant inputs to behave badly: the dimension of the input space is high, and the network uses almost all its resources to represent irrelevant portions of the space [20]. In [21], Silverman illustrates the difficulty of kernel estimation in high dimensions. As shown in Table 1, even at a small dimensionality of 10, roughly 840,000 samples are required to estimate the density at 0 with a given accuracy [21].

Table 1: Required Sample Size for given Number of Dimensions [21]

Dimensionality	Required Sample Size
1	4
2	19
5	786
7	10,700
10	842,000

In 2003, research conducted by Guyon and Elisseeff reported on gene expression datasets up to 60,000 variables wide, and text classification datasets 15,000 variables wide and nearly 1 million instances deep [22]. Those numbers continue to grow daily, as well as in other fields such as satellite imagery, hyperspectral imagery, financial data, consumer data, and etc [23]. In fact, the UCI Machine Learning Repository, where many of the datasets used in this research are attained from, contains two datasets 100,000 features wide and one dataset spanning over 3 million features [24]. Improving prediction accuracy within problems of high data dimensionality is the primary motivation for using feature selection.

Feature selection can also significantly improve the comprehensibility of resulting classifier models, generated rules, or relationships among features. For instance, methods such as ID3 [25] or C4.5 [26] can create very complex tree structures on high-dimensional datasets, making it almost impossible to comprehend with the naked eye. By removing many of the unnecessary and noisy features, the resulting structure will be simpler and comprehensible, allowing for a better understanding of the problem domain and the relationships between features.

The key benefit of feature selection is that it directly targets the curse of dimensionality, since it can eliminate irrelevant and redundant features and reduce noise in the dataset. Moreover, as the dimension of the feature space increases so does the probability of over-fitting,

and as shown by [27], feature selection provides a powerful means to avoid over-fitting. Feature selection can also lead to a more understandable model, allow the data to be more easily visualized, and with a reduction in dimensionality, can decrease the amount of time and memory required by learning algorithms [18]. The motivation for this research, however, goes deeper into the methodology of feature selection, a topic that will be further explored later in this work.

1.2 Problem Description

Although there are a plethora of feature selection algorithms, all differing in some shape or form, they all share a commonality; they all attempt to reduce the number of features by selecting an *optimal* subset of the original feature set. This subset, however, almost always contains features that are assumed to be optimal to the *entire* dataset. This then leads to the assumption that the most optimal features to the dataset must be optimal for instances across all classifications. Therein lays the problem and thus the basis for this research.

The hypothesis presented in this work is based on the notion that by allowing the feature selection algorithm to focus its attention on the prediction accuracy of a single classification at a time, the algorithm will produce a separate subset of features optimal to each classification, and thus create a more robust system, one better fit to classify unseen instances. These class-specific subsets can then be utilized to create separate classification models, thereby taking advantage of another technique proven to improve classification accuracy; ensemble learning (or in this case, ensemble feature selection). In summary, this research will attempt to improve on the problem of prediction accuracy by successfully developing and analyzing a class-specific ensemble feature selection architecture.

1.3 Goals and Contributions

As is the case with most novel learning systems, the primary goal of this research is to improve prediction accuracy. This will be done through the development and analysis of the aforementioned ensemble feature selection architecture. Accordingly, this research will attempt to achieve the following detailed goals through the use of a Class-specific Ensemble Feature Selection (CEFS) architecture:

- **Find an effective recombination technique for disagreeable occurrences between the ensemble classifiers**
- **Improve prediction accuracy over baseline traditional feature selection methods**
- **Examine scalability of CEFS in terms of number of features and number of classifications**
- **Explore the inclusiveness/exclusiveness between feature subsets attained through CEFS versus the ones attained though traditional feature selection methods**

The immediate contribution of this research will be in the introduction of a novel architecture to the areas of feature selection, ensemble learning, and machine learning. Additionally, as dataset sizes continue to grow in every discipline, the proposed architecture can further contribute in a much more widespread manner as it is data and algorithm independent. Given its class-specific design, the architecture may also have far reaching implications to fields in science, medicine, vision, finance, security, biometrics, etc.

1.4 Organization

The outline of this thesis first began with a brief introduction on the disciplines and methods explored in this research. The remainder of this thesis is organized as described below.

Chapter 2 begins by discussing classification tasks as part of supervised learning. The chapter provides a brief background on classification, as well as, a general procedure for conducting classification tasks. It then details classifier performance metrics and presents a variety of classification algorithms. It further provides a brief discussion of overfitting within supervised learning tasks and concludes the chapter with key background information on ensemble learning machines.

Chapter 3 explains feature selection in detail. Following a brief introduction and background on feature selection, a general procedure for feature selection is provided. Then, the chapter describes embedded feature selection methods and provides various other feature selection algorithms. Sections on common misconceptions, and relevance and optimality of features are also presented. The use of feature selection in unsupervised and supervised learning is then discussed, concluding with thorough explanations of the different types of feature selection algorithms and existent work in ensemble feature selection.

Chapter 4 provides an in depth explanation of the methodology behind Class-specific Ensemble Feature Selection (CEFS). This chapter includes the theoretical, architectural, and algorithmic frameworks necessary for the design and implementation of CEFS.

Chapter 5 presents an implementation of the suggested architecture, along with results attained from an experiment spanning several algorithms and ten datasets. Moreover, experiment parameters, analysis, and conclusions are discussed. Lastly, Chapter 6 concludes with final observations and analysis of the conducted research, and possible future work.

Chapter 2

Supervised Learning: Classification

The field of machine learning is often divided into three general disciplines: supervised, unsupervised, and reinforcement learning. Supervised learning involves learning a function from examples of its inputs and outputs. Unsupervised learning involves learning patterns in the input when no specific output values are supplied. Reinforcement learning is typically the broadest of the three; rather than being told what to do by a teacher, a reinforcement learning agent must learn from reinforcement, often including the subproblem of learning how the environment works [14].

Supervised learning can further be divided based on the function's output values. Learning a function with continuous output values is called regression learning; conversely, learning a discrete valued function is called classification learning [14]. The architecture presented in this research will concentrate on classification learning only. Accordingly, this chapter will be organized as follows. First, background information on classification learning is provided. Next, a general process for classification tasks is detailed, followed by common metrics for evaluating classifier performance. Then, an algorithmic framework for classifiers is presented, describing a variety of different classifiers. The chapter will then discuss model overfitting and conclude with general information regarding ensemble learning methods.

2.1 Background

Classification is a well studied and thoroughly researched pervasive problem which encompasses many diverse applications. Examples include cancer prediction based on MRIs, galaxy classification based on galaxy shapes, and flower species categorization based on petal size and shape [18]. The input data in classification tasks is a collection of records or instances. Each instance is represented by a tuple (x, y) where x is the feature set and y is the target output or class label. Formally, classification is defined as the task of learning a target function f that maps each feature set x to one of the predefined class labels y. The target function can also be referred to as a classification model [18]. Table 2 displays a sample dataset used for predicting whether or not to play tennis based on weather outlook, temperature, humidity, and wind.

Table 2: Play Tennis dataset [26]

Features | Class Label

Instances

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	High	Strong	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Strong	Yes
D9	Sunny	Cool	Normal	Weak	No
D10	Rain	Mild	Normal	Weak	No

Classification also requires that the class labels be of a categorical nature. Although target values are most often already finite and discrete, they can also sometimes be continuous. In that case, a process called discretization must be performed to transform the output values from continuous to categorical. In this process, the continuous range of output values is typically

divided into a fixed set of intervals, each then represented by a discrete class label. Additionally, classification methods are best suited for predicting datasets with unordered nominal or binary class labels. They are less effective in predicting ordered categories (e.g., classifying humidity as high, medium, or low), as they do not consider implicit ordering among the categories.

2.2 General Classification Process

Most classification techniques follow a similar progression of steps when performing classification tasks. This progression is shown in Figure 1. The classification technique itself (i.e. classifier) is a systematic approach, which builds classification models from a training set with known target outcomes. The training set is first sent to the classifier so that induction can take place. The classifier then builds a model that best fits the relationship between the training data's feature set and its target outcomes. The finished model then receives a test set for deduction and final performance validation.

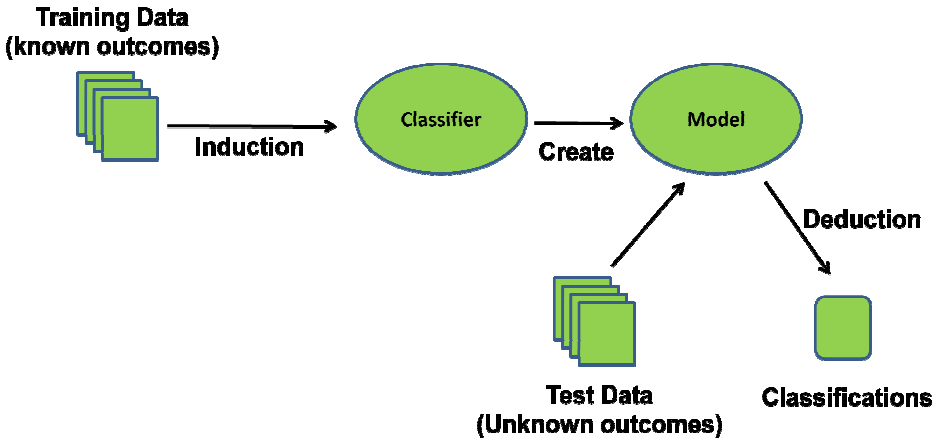

Figure 1: General classification process

It is also of vital importance that the model generated by the classifier fits both the training data and correctly predicts the class labels of the test data. Thus, a key objective of the

classifier is to build models that generalize well, i.e., models that accurately predict unseen instances. Models that fail to provide this generalizing capability are said to overfit the data. A more thorough discussion of overfitting will be provided in Section 2.5 of this chapter.

2.3 Performance Evaluation

Evaluation of the performance of the model is based on the number of instances correctly and incorrectly classified by the model [18]. These counts are organized into a table known as a confusion matrix. Table 3 displays a confusion matrix for a two-class (i.e. binary) classification problem. Each entry f_{ij} in Table 3 represents the number of instances from class i classified as class j. Based on these entries, the total number of correct predictions by the model is ($f_{11} + f_{00}$), and conversely, the total number of incorrect predictions is ($f_{10} + f_{01}$).

Table 3: Confusion Matrix for a two-class prediction problem (with class labels)

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Although a confusion matrix provides the necessary information to gauge performance, it is much more useful to have a single measure which can summarize the matrix's information, even more so on problems containing a higher number of classifications. Performance metrics such as accuracy and error rate, as defined below, provide this useful measure of performance.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{f_{11}+f_{00}}{f_{11}+f_{00}+f_{10}+f_{01}} \qquad \text{(Eq. 1)}$$

$$Error\ rate = \frac{Number\ of\ incorrect\ predictions}{Total\ number\ of\ predictions} = \frac{f_{10}+f_{01}}{f_{11}+f_{00}+f_{10}+f_{01}} \qquad \text{(Eq. 2)}$$

A slightly different representation of the confusion matrix is displayed in Table 4. Although similar information is provided, it is important to note the different terminology as it can be prevalent in machine learning research. Moreover, certain classification tasks, such as cancer prediction, may actually measure performance not solely based on accuracy or error rate, but also on the number of false negatives. For example, diagnosing an individual with cancer when they in fact they don't have it (false positive) will certainly bring emotional pain and anguish, however, diagnosing an individual as cancer free when in fact they have the disease (false negative) may very well cost that individual's life. Hence, many classification tasks often attempt to minimize the number of false negatives, also known as Type II Error. Conversely, the number of false positives is also known as Type I Error.

Table 4: Confusion Matrix for a two-class prediction problem (statistical representation)

		Prediction	
		YES	NO
Actual	YES	*tp* (true positive)	*fn* (false negative) *Type II Error
	NO	*fp* (false positive) *Type I Error	*tn* (true negative)

Although a variety of other performance metrics exist (e.g. Balanced Error Rate or Area under the ROC [28]), two other important metrics must be further defined, as they are discussed later in this document. These two metrics are precision and recall. Precision can be seen as a measure of fidelity, whereas recall is a measure of completeness. In a classification task, a precision score of 1.0 for a class *C* means that every item labeled as belonging to class *C* does indeed belong to class *C* (but does not specify the number of items from class *C* that were not labeled correctly). Conversely, a recall score of 1.0 suggests that every item from class *C* was

labeled as belonging to class *C* (but says nothing about how many other items were also incorrectly labeled as belonging to class *C*). Often, the two have an inverse relationship, where an increase of one will cost the reduction of the other. Precision and recall are formally defined below through Eqs. 3 and 4, respectively [29] [28].

$$Precision = \frac{total\ number\ of\ true\ positives}{total\ number\ of\ predicted\ positives} = \frac{tp}{tp+fp} \quad \text{(Eq. 3)}$$

$$Recall = \frac{total\ number\ of\ true\ positives}{total\ number\ of\ actual\ positives} = \frac{tp}{tp+fn} \quad \text{(Eq. 4)}$$

2.4 Algorithmic Framework

Given the plethora of induction algorithms existent within the machine learning canon, a complete and detailed assessment of the algorithmic framework in classification is outside the scope of this research. However, with the extensibility and flexibility of the CEFS architecture, it is important to provide a broad collection of learners which adequately represent the discipline. Accordingly, the remainder of this section will discuss logic based algorithms (i.e. Decision Trees and other rule-based classifiers), perceptron based algorithms (i.e. Neural Networks), statistical based algorithms (i.e. Naïve Bayes classifiers), instance based algorithms (i.e. *k*-nearest neighbor), and Support Vector Machines.

2.4.1 Logic based Algorithms

2.4.1.1 Decision Trees

Decision Trees (DT) [30] [25] [26] are perhaps one of the two most recognizable classification algorithms (Neural Networks being the other). They are trees which classify

instances by sorting them based on feature values, where each node in the tree represents a feature in an instance to be classified, and each branch represents a value which the node can assume. The feature best dividing the training data becomes the root node of the tree. The next best feature becomes an internal node, if it contains outgoing branches, or a leaf node, if it contains no outgoing branches. There are numerous ways of measuring how well features divide the training data (e.g. Information Gain [31] or GINI [30]). Instances are then classified beginning at the root node and sorted based on their feature values [32]. Figure 2 depicts a decision tree generated using an ID3 implementation [25], and based on the Play Tennis dataset shown previously on Table 2. As seen, the biggest advantage of DTs is their comprehensibility.

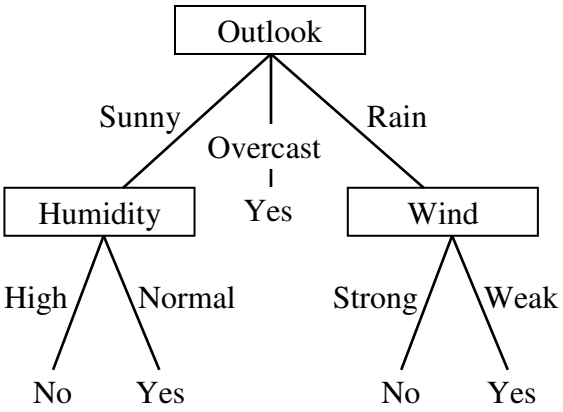

Figure 2: ID3 Decision Tree

Using the decision tree displayed in Figure 2, the instance (Overlook = Rain, Temperature = Hot, Humidity = Normal, and Wind = Weak) would be classified as "Yes". Also note that the Temperature feature is not part of the generated tree. This can often be due to techniques such as pruning, which removes nodes to prevent overfitting and reduce tree complexity [32]. It may also be the case that Temperature just does not offer any further discriminating power than what's already on the tree, thus it is not included.

The problem of constructing optimal binary decision trees is an NP-complete problem. Moreover, decision trees are typically univariate and do not perform well on problems which require diagonal partitioning. They also tend to perform better in problems dealing with discrete/categorical features [32], as continuous valued features must go first through discretization in order to be utilized in the algorithm. For further and more complete reviews of Decision Trees, refer to [33] or [34].

2.4.1.2 Other rule-based systems

Although decision trees can create a set of rules by traversing down each branch of the tree, rule sets can also be directly induced from a variety of rule-based algorithms. Rules come in the form *if* antecedent, *then* consequence. The antecedent contains values taken from the feature set, whereas the consequence contains the target outcome [35]. The difference between the heuristics for decision trees and most other rule-based learners is that the former evaluate the quality of disjoint sets, while the latter evaluate the quality of the set of instances covered by each candidate rule. Thus, it is important for rule-based systems to generate rules which have high predictability or reliability. It is also important to build the smallest rule set that is consistent with the training data, as a large number of rules is often a sign that the algorithm is attempting to "remember" the training data, thereby not generalizing well (i.e. overfitting) [32].

RIPPER [36], a well-known rule-based algorithm, forms rules through repeated growing and pruning. In the growing phase, rules are made more restrictive in order to fit the training data as closely as possible. Conversely, in the pruning phase, rules are made less restrictive in order to avoid overfitting. Much like DTs, the most appealing characteristic of rule-based systems is their easy comprehensibility. For more on rules based systems, please refer to [37].

2.4.2 Perceptron-based Algorithms

As aforementioned, perceptron-based algorithms (e.g. Neural Networks) are among the most well known and often implemented classification algorithms. A single layered perceptron can be described as follows:

If x_1 through x_n are input feature values and w_1 through w_n are connection weights between the values and the predicted output, then the output of the perceptron is computed by its weighted sum, i.e., $\sum_i^n x_i w_i$, followed by an adjustable threshold function. If the sum is above the threshold, the output is 1, otherwise it's 0 [32].

Single layer perceptrons are limited, however, in that the algorithm can only properly classify problems which are linearly separable [14]. In an attempt to solve this problem, multilayered perceptrons (i.e. Artificial Neural Networks) were devised [38]. Multilayer artificial neural networks employ the addition of hidden units between the input and output units. The advantage of adding hidden layers is that it enlarges the hypotheses space which the network can represent. These hidden units can be thought of as a perceptron that represents a soft threshold function in the input space. Then, think of the output as a soft-threshold linear combination of several functions [14]. Figure 3 displays a common multilayer neural network with one hidden layer and four inputs.

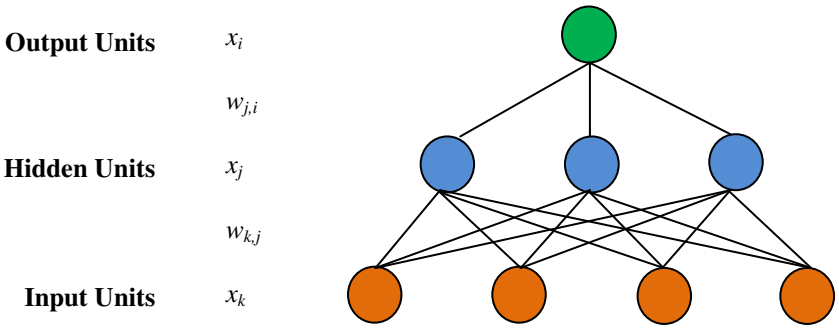

Figure 3: Multilayer Neural Network with one hidden layer and four inputs

Many variations can be implemented in neural networks. For example, any given implementation can have a different number of hidden units, a different number of connections, extra neurons, a different number layers, different activation functions, and beyond. Moreover, the weights between each layer have to be trained before classification can take place. A wide variety of weight training techniques also exist (e.g. back propagation [14] or training weights using Genetic Algorithms [39]). Although Neural Networks can be trained to be great learners, given all of its varying characteristics, its biggest drawback becomes its training time. A more in depth review of Neural Networks can be found in [40].

2.4.3 Statistical-based Algorithms

Statistical learning algorithms are unique in that they provide a probability that an instance belongs to each class, rather than simply predicting a specific classification. Maximum Entropy, for instance, works on the notion that when nothing is known, the distribution should be as uniform as possible. In this technique, training data is used to derive a constraint set for the model which calculates the class-specific expectations for the distribution. Likely the most well known statistical learning algorithm is a Bayesian Network [32].

Bayesian systems are based first on the notion of conditional probabilities. The notation used for conditional probabilities is $P(a|b)$ where a and b are any propositions. They are defined in Eq. 5. From this definition, a Bayes rule (Eq. 6) can be derived. This simple rule underlies all modern AI systems for probabilistic inference and is the second basis for Bayesian systems [14].

$$P(x|y) = \frac{P(x \wedge y)}{P(y)}. \quad \text{(Eq. 5)}$$

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}. \quad \text{(Eq. 6)}$$

A Bayesian Network is then made up of a structural model and a set of conditional probabilities. The model is a directed graph with nodes representing features and arcs representing feature dependencies. These dependencies are quantified by conditional probabilities for each node given its parents [41].

So, first assume that $A_1, A_2, \ldots, A_n$ are n features in the training set. An example E can be represented by $< a_1, a_2, \ldots, a_n >$, where a_i is the value of A_i. Also, let C represent the class label in the training set. So, c can be used to represent the value that C takes and $c(E)$ to denote the class of E. The classifier represented by a general Bayesian network can then be defined according to Eq. 7. Furthermore, assume that all features are independent given the class (assumption of conditional independence), and the resulting classifier is a Naïve Bayes (Eq. 8) [41].

$$c(E) = \underset{c \in C}{argmax}\ P(c)P(a_1, a_2, \ldots, a_n|c) \quad \text{(Eq. 7)}$$

$$c(E) = \underset{c \in C}{argmax}\ P(c)\prod_{i=1}^{n} P(a_i|c) \quad \text{(Eq. 8)}$$

Simply put, the conditional independence assumption assumes that each variable in the dataset is conditionally independent of each other. Given the assumption of conditional independence, instead of computing the class-conditional probability of every combination of a, the algorithm needs only to estimate the conditional probability of each a_i, given c. This approach is more practical since it does not require a large training set to obtain a good estimate of the probability [18]. However, this approach is also what often makes the algorithm less accurate than many of its counterparts (e.g. Neural Networks) [32]. This is because most domains exhibit dependent relationships within their features, breaking down the foundation of conditional independence among variables, and thus negatively affecting the classifier.

2.4.4 Instance-based Algorithms

Instance-based algorithms typically differ from other learning algorithms in that no training phase is necessary and no explicit model of the training data is created. That is, an instance-based learner reuses the training set each time a new instance needs to be classified. For this reason, instance-based learners are also called lazy learners, as the training data is stored at training time and the learning is delayed until classification time [42] [43].

Perhaps the most unique characteristic of an instance-based algorithm is its use of a proximity measure [18]. This measure determines how close (distance) or how similar one instance is to another. The most common way to measure this is by calculating the Euclidean distance, which is the distance between two points in *n*-dimensional space. Instance-based algorithms are implemented under the assumption that unseen instances will be similar to the classification of other instances that are nearby in Euclidean space [42]. The standard Euclidean distance $d(x, y)$ between instances x and y in n-dimensional space is defined as follows [43]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}\,. \qquad \text{(Eq. 9)}$$

Based on the aforementioned assumption of distance, the basic approach of a *k*-nearest neighbor (*k*NN) classifier [44] is to calculate the distance (or similarity) between each test case $z = (x', y')$ and all of the training cases $(x, y) \in D$ to determine its nearest neighbor list, D_z [18]. Then from D_z (which contains the k closest training instances to z) the classification of the new instance is computed. In other words, the outcome of the *k*-nearest training instances will determine the outcome of the new instance that needs to be classified.

Perhaps the most common way of determining the final output is by majority voting, also known as Discrete *k*NN. The Discrete *k*-nearest neighbor algorithm specifies that the most

common outcome of the *k*-nearest instances will be used as the outcome for the new instance. In other words, whichever outcome occurs most frequently, will be the outcome for the instance to be classified. Once D_z is obtained, the outcome of the test case is based on the majority vote of its nearest neighbors, which is formally defined as:

$$y' = \underset{v}{argmax} \sum_{(x_i,y_i)\in D_z} I(v = y_i). \quad \text{(Eq. 10)}$$

In Eq. 10, v represents all of the possible outcomes (classes), y_i represents the class label for one nearest neighbor and $I(\cdot)$ is an indicator function that returns the value one (1) if its argument is true and zero (0) if not. The class with the highest value is then chosen as the outcome. Using majority voting, each neighbor affects the classification equally [18]. The majority voting approach is applied solely to the Discrete *k*NN algorithm.

2.4.5 Support Vector Machines

A considerable amount of attention has been afforded to Support Vector Machines (SVM) recently. In part, this is because SVMs work so well with high-dimensional data by avoiding the curse of dimensionality [18].

SVMs revolve around the concept of a maximal margin hyperplane. That is, if the dataset is linearly separable, then there exists a hyperplane such that all of the instances from one class can be separated from all instances of another. An example of this is shown in Figure 4 [32]. However, as also seen from this example, there exists an infinite number of hyperplanes with the same property. Moreover, although their training errors are zero, there is no guarantee that the hyperplanes will perform as well at test time. So, the algorithm searches for the hyperplane which can offer the greatest class separation, i.e., the maximum margin hyperplane.

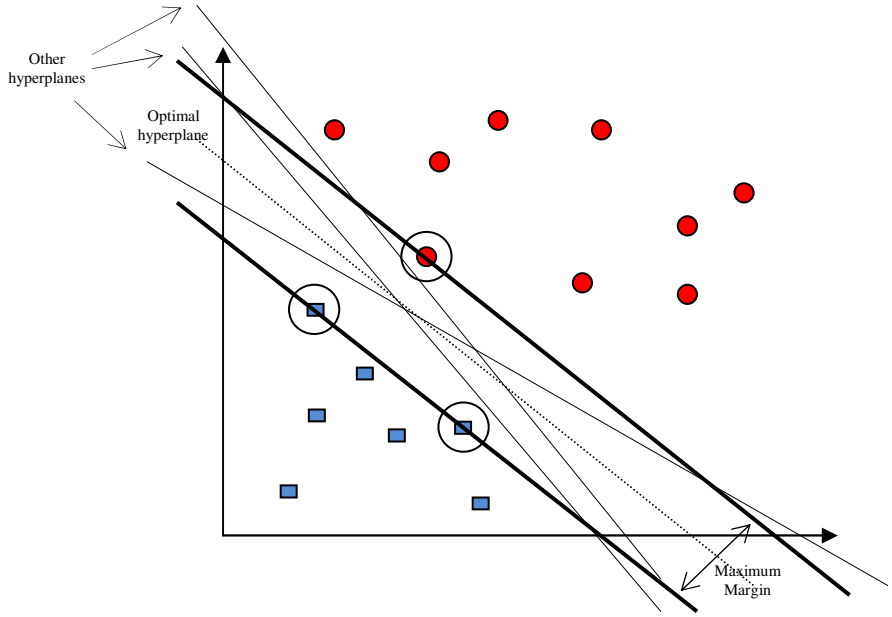

Figure 4: Maximum Margin Hyperplane

The rationale of choosing the hyperplane with the largest margin is that decision boundaries with large margins tend to generalize better than those with small margins. Intuitively, if the margin is small, any slight changes in the decision boundary can significantly affect the system's performance. Hence, classifiers that produce decision boundaries with small margins are more susceptible to poor generalization and overfitting. Linear separability is not a requirement for SVMs however. A soft margin approach allows SVMs to construct a linear decision boundary even in situations where the classes are not linearly separable [18]. More information on SVMs can be found in [45] or [46].

2.5 Overfitting

Errors committed by a classifier can be categorized as either training or generalization errors. Training errors are instances misclassified during training and generalization errors are unseen instances misclassified during test time. As aforementioned, it is important for a model

to not only fit the training data well, but to also accurately classify unseen records. When a classifier creates a model that fits the training data too well, essentially just "remembering" the training data, it can result in poor generalization. This is known as overfitting [18].

On the path to understanding model overfitting, note that the training error can typically be reduced by simply increasing the model complexity. For instance, the leaf nodes of a decision tree can be expanded until they perfectly fit the training data. Such a tree would yield a training error of zero, but at the cost of a much more complex tree and the high likelihood of a significant increase in generalization error. This is due to the addition of nodes which accidently fit noise points in the training data, thus degrading generalization performance [18].

Overfitting can occur for a variety of reasons: noise in the training set, lack of representative samples, etc. However, there are also steps for decreasing the chances of generating an overfit model. Perhaps the most prevalent step is to reduce model complexity (e.g. pruning). The system can also use a separate validation set for calculating its training error instead of the training set itself. Regardless of the technique, model overfitting is a problem which must be carefully watched in any classification task.

2.6 Ensemble Methods

The main goal of an ensemble is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown instances. Figure 5 shows a basic view of an ensemble method [18]. As depicted, three main steps exist: training set generation, learning, and integration. Step 1 begins with the original training set D. From this training set, t data subsets are created (D_1, D_2, …, D_t). Bagging and boosting are common ways to accomplish this step [18]. Then in Step 2, t base classifiers are generated (C_1, C_2, …, C_t).

These classifiers may all be the same, all different, or contain any combination of the same or different classifiers. Each classifier C_i is trained using the subset D_i. Finally in Step 3, the prediction of each classifier is combined in a predetermined way to produce the resulting classification.

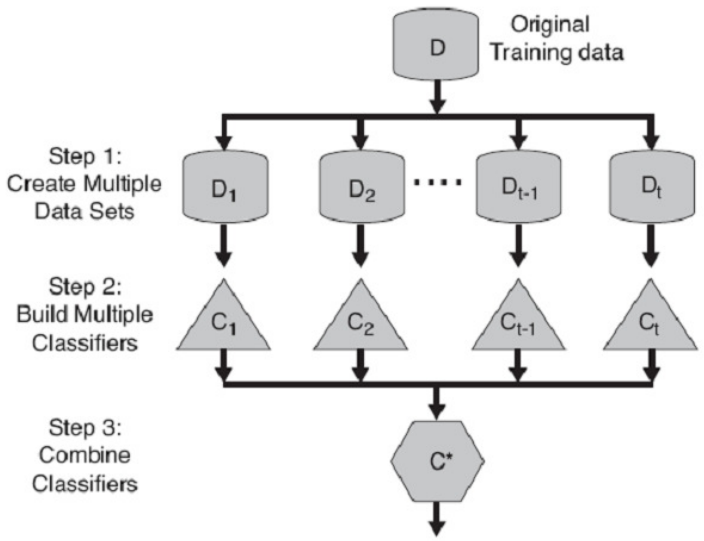

Figure 5: A logical view of the ensemble learning method [18]

Two primary approaches exist to the integration phase: combination and selection. In the combination approach, the base classifiers produce their class predictions and the final outcome is composed using those predictions. In the selection approach, one of the classifiers is selected and the final prediction is the one produced by it [47]. The simplest and most common combination method is voting, also known as majority voting. In voting, the classification predicted by a base classifier is counted as a vote for that particular class value. The class value with the most votes becomes the final classification [47]. A simple and popular selection method is cross-validation majority (CVM) [48], which estimates the accuracy of each base classifier using cross-validation and selects the classifier with the highest accuracy. Although CVM is a

selection method which chooses one classifier for the whole data space, more sophisticated selection methods which estimate local accuracy [49] or meta-level classifiers do exist [50]. Perhaps the most commonly used integration techniques are voting [9], simple and weighted averaging, and a posteriori [51] [52].

According to [53], the main objective when building the base classifiers is to maximize the coverage of the data, which is the percentage of the instances which at least one base classifier can classify correctly. Reaching coverage greater than the accuracy of the best classifier, however, requires diversity among the base classifiers [11] [51] [52]. Although research is always ongoing on new approaches to increase diversity, some methods include training on different subsets of the training set, using different learning algorithms, injecting randomness, and training on different sets of input features [54]. The latter is where ensemble feature selection is successfully applied.

Chapter 3

Feature Selection

Recall from Chapter 1 that research has shown that much of the power in classification comes not from the specific learning method, but from proper formulation of the problems and crafting the representation to make learning tractable. This suggests that the preparation and representation of the data is of the utmost importance for successful learning, given that many learning algorithms perform similarly on most domains [15]. In the context of machine learning, the formulation and transformation of data prior to learning is known as data preprocessing.

Data preprocessing is itself a broad research area consisting of a number of different strategies and techniques interrelated in complex ways. These include techniques such as aggregation, sampling, and variable transformation. The key commonality within these and other data preprocessing techniques is that they all seek to improve learning with respect to time, cost, and quality. Simply put, data is of high quality if it is suitable for its intended use [18]. This is analogous to statistics and experimental sciences, where emphasis is especially enforced on the careful design of experiments to collect data relevant to the specific proposed hypothesis. Similarly, data preprocessing techniques improve the quality and representation of data so that classification tasks can be more efficient and effective. Perhaps one of the most studied and explored data preprocessing techniques is feature selection.

The remainder of this chapter is organized as follows. First, background information is provided on feature selection, followed by an in depth explanation of the general feature

selection process. Next, feature selection embedded methods are discussed. A general algorithmic framework and common feature selection misconceptions are addressed next. Then, a thorough discussion on the subject of relevance and optimality as it pertains to feature selection is presented. Next, brief sections on feature selection in unsupervised and supervised learning follow. Finally, previous research conducted in ensemble feature selection is summarized and explained, as it pertains highly to the architecture explored here.

3.1 Background

Feature selection can be described as a technique which finds a *good* subset of the original input features under some objective measure, such as prediction accuracy, structure size, or minimal use of input features. Consistent with notions such as Occam's razor [55], minimum description length [56], and minimum message length [57], feature selection attempts to find structures which correctly predict unseen instances, yet are not so complex that they overfit the data [58]. Within the general classification process, previously shown in Figure 1, feature selection occurs just prior to induction between the training data and the classifier (Figure 6).

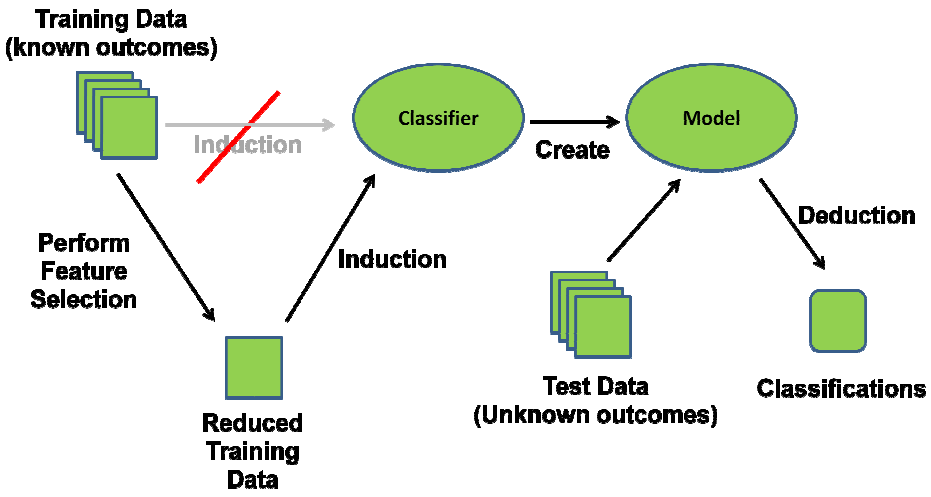

Figure 6: Feature Selection within general classification process

Furthermore, by extracting irrelevant and redundant information from a given dataset, significant computing time can be saved and models that generalize better to unseen points can be built [59]. Formally, feature selection can be defined as follows [7]:

Definition 1 (feature selection) Let Y be the original set of features, with cardinality n. Let d represent the desired number of features in the selected subset X, $X \subseteq Y$. Let the feature selection criterion function for the set X be represented by $J(X)$. Without lost of generality, let us consider a higher value of J to indicate a better feature set. Formally, the problem of feature selection is to find a subset $X \subseteq Y$ such that $|X| = d$ and

$$J(X) = \max_{Z \subseteq Y, |Z|=d} J(Z). \qquad \text{(Eq. 11)}$$

The relationship between feature selection and the performance of a learning algorithm, however, depends largely on whether or not the assumption of monotonicity holds, i.e.

$$J(A \cup B) \geq J(A) \qquad \forall\, A, B \subseteq Y. \qquad \text{(Eq. 12)}$$

This assumption implies that additional features always provide additional discriminating power, thus improving the performance of a learning algorithm [60]. It also assumes that the full distribution of the data is known to the learning algorithm. The optimal Bayes rule, for example, is monotonic; adding features cannot decrease the accuracy. In this case, feature selection would not be recommended. In the vast majority of practical and real world scenarios, however, monotonicity does not hold, often because learning algorithms are not given access to the underlying distribution [61]. In addition, noisy features can increase noise in training examples, making it difficult for the learning algorithm to separate useful signal from noise [60].

In fact, many supervised learning algorithms, including decision tree algorithms such as ID3 [25], C4.5 [26], and CART [30], and instance-based algorithms, such as IBL [62] [63], are known to lose prediction accuracy when faced with many features unnecessary for predicting the output, i.e. *irrelevant* features [61]. These are features that contain almost no useful information

for the learning task at hand. For example, students' ID numbers are irrelevant for predicting students' GPAs [18]. Conversely, algorithms such as Naïve-Bayes [64] [65] [66] are robust with respect to irrelevant features, but their performance may degrade quickly if correlated features, i.e. *redundant* features, are added [61]. Such features duplicate much or all of the information contained in one or more other features. For instance, the purchase price of a product and the amount of sales tax paid contain much of the same information [18].

In the context of machine learning, irrelevant and redundant features typically serve no purpose except to increase induction time. In fact, such features may also cause other problems as they can confuse the learning algorithm by helping to obscure the distributions of the small set of truly relevant features for the task at hand [19]. Feature selection targets those very problems as it is a technique that reduces the number of features and removes irrelevant, redundant or noisy data by selecting a subset of the original feature set [4].

Although optimal feature selection is typically intractable [61] and many problems related to it have been shown to be NP-hard [67] [68], some feature selection algorithms have proved to be significantly more stable than others [69] [70]. Furthermore, feature selection has been shown to yield benefits such as facilitating data visualization and data understanding, reducing measurement and storage requirements, reducing training and utilization times, and defying the curse of dimensionality to improve prediction performance [22]. As a result, it can be found in many areas of research such as bioinformatics [71] [72] [73] [74], machine learning [71] [75] [76] [19], text classification/mining [77] [78] [79] [80] [81], case-based reasoning [82], statistics [83] [84] [85], security [79] [86], pattern recognition [7] [72] [87] [88], data mining [59] [89] [90], and various others. In addition, numerous reviews and comparative studies on existing feature selection algorithms have been conducted [4] [91] [92] [93] [94].

3.2 General Feature Selection Process

Most feature selection methods follow a four step process: subset generation, subset evaluation, stopping criterion, and result validation (Figure 7) [91]. Beginning with subset generation, the selected search strategy produces candidate feature subsets. Each subset is then evaluated and compared to others according to a given evaluation criterion. The best subset is kept and this process is repeated until a stopping criterion is reached, at which point, the selected subset is validated using the pre-selected classifier. The subsequent subsections will provide detailed explanations of each step.

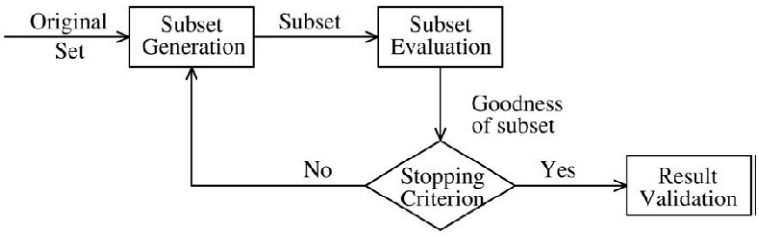

Figure 7: Feature Selection Process [91]

3.2.1 Subset Generation

Subset generation is essentially a heuristic searching problem, with each location in the search space signifying a candidate subset for evaluation. Furthermore, it can largely be determined by answering two basic questions: where to start and how to search [4].

First, a starting point (or points) must be selected. Some algorithms start with an empty set and incrementally add features (i.e. forward), some start with a full set and incrementally remove features (i.e. backward), or some start with both ends and add and remove features simultaneously (i.e. bidirectional). Other algorithms may start with a predetermined number of randomly selected subsets in order to avoid being trapped by local optima [86].

Second, the algorithm must specify its searching strategy. Since a dataset with N features has 2^N unique candidate subsets, exhaustive search is typically not feasible and thus rarely considered or used. So, three main strategies have been previously studied: complete, sequential, and random search [4].

3.2.1.1 Complete Search

Perhaps the biggest advantage of a complete search strategy is that it is guaranteed to find the optimal feature subset based on the evaluation criterion. While an exhaustive search is complete, a strategy does not have to be exhaustive in order to be complete [4]. In fact, algorithms such as beam search [86] and branch-and-bound (BB) [95], can find the optimal subset much more quickly than exhaustive search. However, such algorithms are impractical for problems with large feature sets as the worst case complexity is still exponential. Moreover, methods like BB require the feature selection criterion to be monotonic, although [96] has shown that BB can still work well even in cases where the feature selection criterion is nonmonotonic.

3.2.1.2 Sequential Search

Although sequential strategies tend to be much faster than complete strategies, the loss of completeness can also mean the loss of optimality as it is no longer a guarantee that the optimal solution will be found. These strategies are deterministic and iteratively add or remove features until a stopping criterion is reached. Most sequential search strategies will follow a greedy hill-climbing approach, such as sequential forward selection (SFS) [7] [87], sequential backward selection (SBS) [7] [87], or bidirectional selection [93]. The pseudo-code for SFS and SBS are provided in Figure 8 and Figure 9, respectively. These strategies are the most commonly used

methods for performing feature selection given that they are simple to implement and fast in producing results as the order of the search space is typically $O(N^2)$ or less [4].

Sequential Forward Selection (SFS)

//Step 1 - Initialize candidate subset

$X_0 = \{\emptyset\} \qquad j = 0$

//Step 2 - Find next feature greedily
// based on evaluation metric

$x^* = \arg\max_{x \notin X_j} [J(X_j + x)]$

//Step 3 - Update next feature subset

$X_{j+1} = X_j + x^*$

//Step 4 - Check stopping criterion

If $j < d$

Then $j = j + 1$ and go to Step 2

Else

Return X_j

Figure 8: Pseudo-code for Sequential Forward Selection (SFS) Algorithm

Sequential Backward Selection (SBS)

//Step 1 - Initialize candidate subset

$X_0 = \{Y\} \qquad j = 0$

//Step 2 - Find next feature greedily
// based on evaluation metric

$x^* = \arg\max_{x \in X_j} [J(X_j - x)]$

//Step 3 - Update next feature subset

$X_{j+1} = X_j - x^*$

//Step 4 - Check stopping criterion

If $j < d$

Then $j = j + 1$ and go to Step 2

Else

Return X_j

Figure 9: Pseudo-code for Sequential Backward Selection (SBS) Algorithm

3.2.1.3 Random Search

Random strategies begin with a collection of randomly chosen candidate subsets. Then, the strategy can follow in one of two directions: sequential or stochastic search. Examples such as random-start hill-climbing and simulated annealing follow a deterministic sequential search pattern to generate subsequent candidate solutions, while also injecting randomness throughout the process [86]. Stochastic methods generate new candidate subsets in a completely random manner (i.e. new subsets do not grow or shrink based on deterministic or sequential changes to previous subsets). Examples such as the Las Vegas algorithm [97] and LVW [98] use the aid of randomness to help escape local optima, allowing it to more adequately search for the global

optima [4]. LVW, for instance, uses a probabilistic approach to explore high-order relationships among features. It guarantees the optimal solution(s) at the $(k + 1)$th experiment with probability $l / (2^D - k)$ where D is the number of original features and l is the number of optima [98].

Perhaps the most applied and studied stochastic strategy, is that of a Genetic Algorithm (GA) driven structure [99] [100] [101]. Popularized by Holland and Goldberg, GAs have shown to be effective in searching complex high-dimensional spaces [102] [103]. Moreover, the use of crossover and mutation operators allows GAs to efficiently traverse the search space without getting trapped in local optima. First combined with feature selection by [99], in a GA approach, candidate subsets (i.e. "chromosome") are represented as binary strings of length N, where N is the number of features in the dataset. Each position in the binary string contains a zero or a one representing the absence or the presence of the corresponding feature.

As described by the pseudo-code provided in Figure 10, a predetermined number of chromosomes (i.e. "population") is first generated randomly and then updated through each iteration (i.e. "generation") until a stopping criterion is reached. In every generation, each chromosome is assigned a fitness value based on the predetermined evaluation criterion. This value often determines how likely that chromosome is to survive and breed into the next generation. New chromosomes (i.e. "offspring") are then created in one of two ways: crossover and/or mutation. In crossover, two existing chromosomes are selected as parents and their binary strings are mixed to create new chromosomes. In mutation, random bits ("genes") of a single parent or a newly created child are mutated by simply flipping the binary value, thus creating a new chromosome. Once the new chromosomes have been generated, a new population is selected for the next generation and the process begins all over again. Note that depending on selection strategy, old and new chromosomes may or may not survive each generation.

GA driven Feature Selection

$t = 0$ //initialize generation counter
$X'_t = \{X_0, X_1, \dots, X_\mu\}$ //randomly generate initial μ chromosomes (population)
$F_{t_1} = Eval(X'_t)$ //assign fitness values by evaluating population

While (Not Done) //continue until stopping criterion is reached
{
$P_t = Select\_Parents(X'_t, F_{t_1})$ //select parents for procreation
$O_t = Procreate(P_t)$ //create offspring
$F_{t_2} = Eval(O_t)$ //assign fitness values to offspring
$X'_{t+1} = Select\_Surivors(X'_t, F_{t_1}, O_t, F_{t_2})$ //select next generation
$t = t + 1$ //increment counter
}

Figure 10: Pseudo-code for GA driven Feature Selection Algorithm

3.2.2 Subset Evaluation

As aforementioned, each new candidate subset needs to be evaluated by an evaluation criterion to determine the goodness of the subset. Since a great number of different evaluation techniques exist, it should first be pointed out that although a candidate subset may be optimal or near optimal under one criterion, it may or may not be considered optimal under others. The following three major groups of evaluation criteria exist and will be discussed in subsequent sections: independent, dependent, and hybrids.

3.2.2.1 Independent Criteria

Independent criteria are used to evaluate candidate solutions based on intrinsic characteristics of the features themselves, such as distance, information, dependency, and consistency measures [104] [105] [106] [93]. Distance measures (i.e. separability, divergence, or

discrimination) try to find features that can separate the dataset's class labels as far as possible. For example, in a two class problem, a feature *X* is preferred over feature *Y* if *X* generates a greater difference between the two class conditional probabilities than *Y*. Information measures normally quantify the information that can be gained from each feature. For example, the information gain from feature *X* is determined by the difference between the prior uncertainty and expected posterior uncertainty using *X*. Dependency measures (i.e. correlation or similarity) gauge a feature's ability to predict the value of one variable from the value of another. For instance, feature *X* is chosen over feature *Y* if the association between feature *X* and class *C* is higher than the association between feature *Y* and class *C*. Consistency measures rely on class information feature bias in selecting subsets. These measures try to find a minimum number of features which separate classes as consistently as the full set of features can [4].

Independent criteria are typically used in, and most widely referred to as, filter methods. Coined by John, Kohavi and Pfleger in [58], filter methods have their name because they *filter* out irrelevant features before induction occurs, using a predefined metric. The process uses general characteristics of the training set to select some features and exclude others. As displayed in Figure 11, filter methods evaluate candidate subsets through metrics independent of the induction or learning algorithm and select the *best* subset accordingly. Since filtering methods do not involve the use of a learning algorithm to evaluate candidate sets, they can be combined with any learning algorithm after the filtering is complete. Moreover, they are a computationally effective form of data pre-processing, especially when compared to wrapper methods (which use dependent criteria) [74] [107]. Filter methods rely on abstract metrics of features, which indicate important properties of promising features such as orthogonality, large variance, multi-modality of marginal distributions, high kurtosis, low entropy, etc [60]. As such,

filter methods do not inherit bias from learning algorithms and are also more computationally efficient than wrapper methods [4].

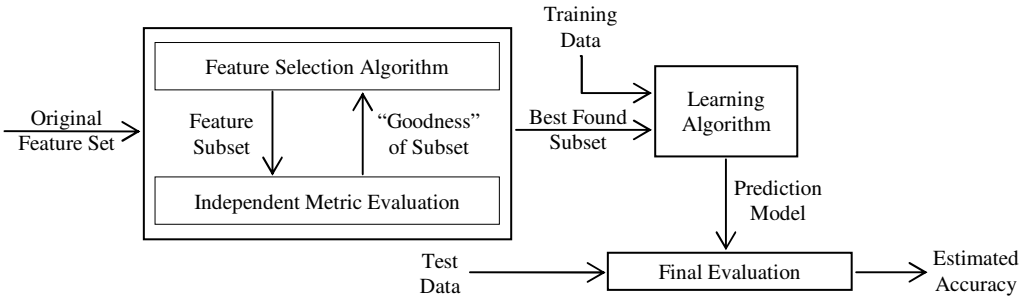

Figure 11: Filter approach to feature selection

Figure 12 describes a generalized form of a filter algorithm, provided by [4]. Given dataset *D*, begin with a given subset S_0 (an empty set, a full set, or any randomly selected subset) and search through the feature space using a particular search strategy. Each generated subset *S* is evaluated by an independent measure *M* and compared with the previous best. If better, it's regarded as the current best subset. The search iterates until a predefined stopping criterion is reached. The algorithm outputs the last current best subset S_{best} as the final feature subset [4].

Filter Algorithm

input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
δ // a stopping criterion
output: S_{best} // an optimal subset

01 **begin**
02 **initialize:** $S_{best} = S_0$;
03 $\gamma_{best} = eval(S_0, D, M)$; // evaluate S_0 by an independent measure M
04 **do begin**
05 $S = generate(D)$; // generate a subset for evaluation
06 $\gamma = eval(S, D, M)$; // evaluate the current subset S by M
07 **if** (γ is better than γ_{best})
08 $\gamma_{best} = \gamma$;
09 $S_{best} = S$;
10 **end until** (δ is reached);
11 **return** S_{best};
12 **end**;

Figure 12: Generalized Filter Algorithm [4]

One of the simplest filtering processes is to evaluate each feature individually based on its correlation with the target function and then select the *k* features with the highest values. This method has been shown to achieve good results in text categorization tasks [108] [109] [110], often used in combination with either a Naïve-Bayesian classifier or a nearest neighbor classifier [107]. Other widely used metrics include Information Gain, Odds Ratio, Chi-Squared [77], Log Probability Ratio [77], FOCUS [111], RELIEF [112] [113], Potential Difference [114], Pearson Correlation Coefficient [115], etc. RELIEF, for instance, assigns a "relevance" weight to each feature, which represents the relevance of the feature to the target concept. It samples instances randomly and updates the relevance values based on the difference between the selected instance and the nearest instances of the same and opposite class ("near-hit" and "near-miss") [61]. The algorithm does require the problem to only contain two classes however. Another example, FOCUS, looks for minimal combinations of attributes that perfectly discriminate among the classes. It begins by looking at each feature individually, goes on to pairs of features, triples, and so forth, stopping only when it finds a combination which generates pure partitions of the training set.

There are some drawbacks to filter methods however. First, most filtering methods require the pre-selection of a set number of features to be chosen. If the number is too high, irrelevant features will be kept and accuracy may suffer. If the number is too low, useful features may not be selected, once again affecting accuracy. Some remedies to this problem include using a hold-out set to test for the best *k*, or to use a hill-climber or some other evolutionary computation algorithm to find an optimal *k*. These solutions, however, will negatively impact what is perhaps the biggest benefit of a filtering algorithm, its relative speed. A second drawback is that filtering algorithms may miss features that would otherwise be useful

to the learning algorithm which will predict unseen instances. Since the algorithm bases its selection purely on feature metrics, it may miss features deemed useful by some learning algorithms but irrelevant in others. This is why wrapper methods are widely known to outperform filter methods (in terms of prediction accuracy) [4] [74].

3.2.2.2 Dependent Criteria

Dependent criteria primarily involve the use of a learning algorithm to assess the goodness of candidate subsets. In most cases, predictive accuracy, or some variation of it, is used as the evaluation criterion. Accordingly, methods that utilize dependent criterion as its subset evaluation criterion are also known as wrapper methods. These methods occur outside the basic learning algorithm, but also use said learning algorithm as a subroutine, rather than just as a post-processor. For this reason, John, Kohavi, and Pfleger [58] refer to them as *wrapper* methods. As displayed in Figure 13, each candidate feature subset is evaluated by running the selected training or validation data through the learning algorithm and using the estimated accuracy of the algorithm as its evaluation metric. As aforementioned, the biggest benefit to using wrapper methods is their tendency to outperform (in terms of prediction accuracy) their filter counterparts. The general argument is that the learning algorithm, which uses the feature subset, should provide a better estimate of accuracy than a separate metric that may have an entirely different bias [107]. Whereas a filter method will use metrics to determine the usefulness of each feature, wrapper methods will use the learning method to determine the usefulness of a candidate set of features for the learning method itself. At the end of the process, not only will the algorithm produce an optimal set of features, but one which will perform most optimally under the aforementioned learning method.

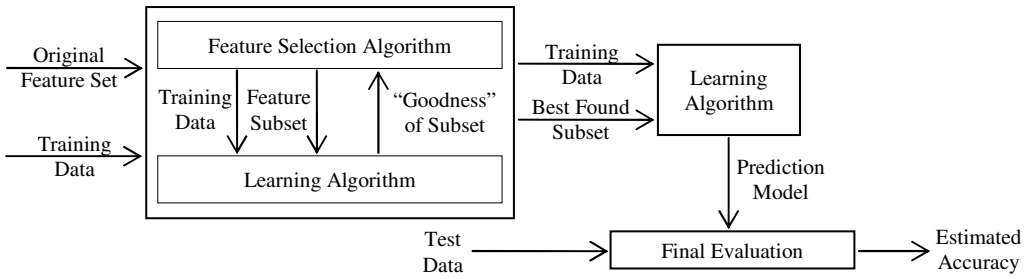

Figure 13: Wrapper approach to feature selection

Although wrapper methods have shown to be more computationally expensive than filter methods due to their repeated calls to the learning algorithm for evaluation, research has shown that wrapper methods tend to give superior performance as feature subsets are better suited to the predetermined learning algorithm [4]. They have been shown to significantly boost predictive performance in classifiers such as C4.5 and Naïve-Bayes [61] and on instance based algorithms such as nearest-neighbor, as they would otherwise always use all available features for classification [116].

Figure 14 provides the pseudo-code for a generalized wrapper algorithm [4], which is quite similar to the generalized filter algorithm (Figure 12 – Section 3.2.2.1) except that it uses a learning algorithm *A* instead of an independent measure *M* for the evaluation of each candidate subset. For each generated subset *S*, the algorithm evaluates its goodness by applying the learning algorithm to the data with subset *S* and evaluating the accuracy. Thus, different learning algorithms may produce different feature selection results. Varying the search strategies via the function generate(*D*) and learning algorithms (*A*) can result in different wrapper algorithms.

Wrapper Algorithm

input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
δ // a stopping criterion

output: S_{best} // an optimal subset

```
01 begin
02     initialize: S_best = S_0;
03     γ_best = eval(S_0, D, A); // evaluate S_0 by a mining algorithm A
04     do begin
05         S = generate(D); // generate a subset for evaluation
06         γ = eval(S, D, A); // evaluate the current subset S by A
07         if (γ is better than γ_best)
08             γ_best = γ;
09             S_best = S;
10     end until (δ is reached);
11     return S_best;
12 end;
```

Figure 14: Generalized Wrapper Algorithm [4]

Just as variations can be found to many wrapper methods, they can also be found in the learning algorithm. Variations of Neural Networks, Bayesian networks, and SVMs are often applied on different wrapper problems [117]. Research done in [90] also discusses overfitting and dynamic search in regards to wrapper methods. Finally, wrapper methods have also been widely used in unsupervised classification, typically using measures such as cluster compactness, scatter separability, or maximum likelihood as evaluation criterion [118] [59].

3.2.2.3 Hybrid Criteria

Hybrid feature selection algorithms combine the use of filter and wrapper methods in an attempt to exploit the benefits of both. By combining both techniques at different stages, hybrids are able to take advantage of the speed of a filter method and the accuracy of a wrapper method [73] [119]. Other implementations may also include the use of an embedded method, such as a decision tree, instead of a wrapper method [120] [121]. Similar to the generalized algorithms previously provided, Figure 15 describes the pseudo-code for a generalized hybrid algorithm [4].

The algorithm first uses a filter method to find the best subset for a given predetermined cardinality and then uses the learning algorithm to select the final best subset among the best subsets across different cardinalities.

Hybrid Algorithm

input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
output: S_{best} // an optimal subset

```
01 begin
02     initialize: S_best = S_0;
03     c_0 = card(S_0); // calculate the cardinality of S_0
04     γ_best = eval(S_0, D, M); // evaluate S_0 by an independent measure M
05     θ_best = eval(S_0, D, A); // evaluate S_0 by a mining algorithm A
06     for c = c_0 + 1 to N begin
07         for i = 0 to N − c begin
08             S = S_best ∪ {F_j}; // generate a subset with cardinality c for evaluation
09             γ = eval(S, D, M); // evaluate the current subset S by M
10             if (γ is better than γ_best)
11                 γ_best = γ;
12                 S'_best = S;
13         end;
14         θ = eval(S'_best, D, A); // evaluate S'_best by A
15         if (θ is better than θ_best);
16             S_best = S'_best;
17             θ_best = θ;
18         else;
19             break and return S_best;
20     end;
21     return S_best;
22 end;
```

Figure 15: Generalized Hybrid Algorithm [4]

The algorithm begins with a given subset S_0 (typically an empty set in sequential forward selection) and iterates to find the best subsets at each increased cardinality. In each round for a best subset with cardinality c, it searches through all possible subsets of cardinality $c + 1$ by adding one feature from the remaining features. Each newly generated subset S with cardinality $c + 1$ is evaluated by an independent measure M and compared with the previous best. If S is better, it becomes the current best subset S'_{best} at level $c + 1$. At the end of each iteration, a learning algorithm A is applied on S'_{best} at level $c + 1$ and the quality of the mined result is compared with that from the best subset at level c. If S'_{best} is better, the algorithm continues to

find the best subset at the next level; otherwise, it stops and outputs the current best subset as the final best subset. The quality of results from a learning algorithm provides a natural stopping criterion in this model [4].

As a hybrid method, the method presented in [117] begins by pre-processing the dataset by using two different filter methods, F-score and Information Gain. The separate feature subsets are then combined and processed through a Sequential Floating Wrapper method, which yields the final feature subset. A Support Vector Machine (SVM) then utilizes the resulting feature subset to compute the classification accuracy.

3.2.3 Stopping Criterion

A preselected stopping criterion determines when the feature selection process needs to stop. There is often little variation in the stopping criterion used for most feature selection methods. The subsequent list, provided by [4], details a few of the most frequently used ones.

1. The search completes
2. A given bound is reached, such as a minimum number of features or a maximum number of iterations
3. Subsequent addition or deletion of any feature does not produce and better subset
4. A sufficiently good subset is found (e.g., the classification error rate for a candidate subset is less than the allowable error rate for a particular task).

3.2.4 Result Validation

Given the case that prior knowledge about the data is available (e.g. synthetic data), then a simple way to validate the selected subset is by comparing it to the known optimal subset. Knowledge about irrelevant and redundant features can also aid in validation as those features

are not expected to be selected. In most applications, however, such prior knowledge is not available. In such settings, the validation task typically falls to the comparison of the learner's error rate on the full set of features versus the error rate on the selected set of features [4] [93].

3.3 Embedded Methods

Embedded methods follow a slightly different feature selection pattern than the general process discussed in Section 3.2, so they are presented here separately. Such methods perform feature selection as part of the learning algorithm itself. That is, within the process of the learning algorithm, the algorithm decides which attributes to use and which to ignore [18]. Decision Trees are perhaps the most well known example of embedded methods [25] [26] [30].

Much like wrapper methods, embedded methods interact directly with a specific learning algorithm. In other words, the feature selection algorithm is *embedded* into the classifier model itself rather than using the model to evaluate candidate feature sets. Moreover, these methods have the advantages that they interact directly with the classifier while also being less computationally expensive than wrapper methods [74]. A generalized form of this technique is not provided here as embedded methods may be highly different in implementation depending on the learning algorithms encompassing them (e.g. decision tree vs. SVM).

Mao, Mohiuddin, and Jain, for instance, use a multilayer feedforward Neural Network with back propagation to simultaneously develop both an optimal feature set and an optimum classifier [122]. They define and utilize a "node saliency" measure to prune the least salient nodes thus reducing the complexity of the network after it has been trained.

Recursive partitioning methods such as decision trees [25] [26] [30], employ a greedy search through the space of decision trees, at each stage using an evaluation function to select the

feature which has the best ability to discriminate among the classes. They partition the data based on this feature and repeat the process on each subset, extending the tree downward until no further discrimination is possible. Other embedded methods include Separate-and-Conquer for decision lists [123] and Support Vector Machines of Recursive Feature Elimination [71].

3.4 General Algorithmic Framework

Given the previously provided general procedure for feature selection algorithms, it is now possible to discuss an algorithmic framework. Accordingly, this chapter will present a general feature selection taxonomy (Figure 16) and a categorical table (Table 5) which portrays many of the existent feature selection algorithms. Moreover, although the categorical table of algorithms is by no means exhaustive, it does provide a vast and broad representation of proposed and studied feature selection algorithms.

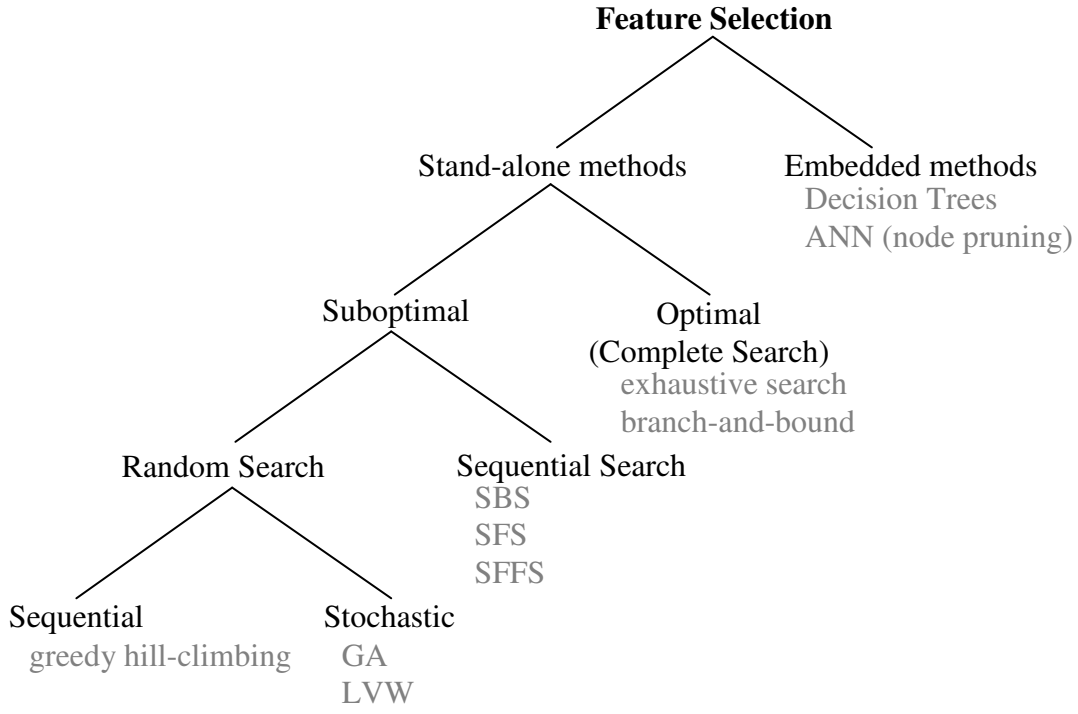

Figure 16: Taxonomy of Feature Selection Algorithms

As mentioned, a broad taxonomy of feature selection algorithms is presented in Figure 16. As shown, feature selection is first divided into stand-alone and embedded methods. Stand-alone methods are then split according to their respective guarantees of optimality (optimal and suboptimal). Suboptimal methods are further divided into those using sequential search and ones using random search. A final distinction is then made of random methods, dividing them into methods that search in a stochastic manner and methods that begin randomly, but then follow a sequential pattern for search. Also note that all leaf nodes include at least one example of an existent method related to its category. It should be stated that although the taxonomy does not mention subset evaluation methods such as filter, wrapper, and hybrid methods, it does not do so because the majority of these methods are structured so that they may be combined with most stand-alone methods.

Subsequently, Table 5 provides an in depth categorical table of existing feature selection methods [4]. The table is divided by two dimensions (Search Strategies and Evaluation Criteria), as these are the two most dominating factors in feature selection algorithms. Within Search Strategies, algorithms are categorized as Complete, Sequential, and Random. Under Evaluation Criteria, algorithms are categorized into Filter, Wrapper, and Hybrid. Filter algorithms are further divided into Distance, Information, Dependency, and Consistency. Although most of the algorithms listed in this section are not implemented in this research it is noteworthy to mention that the CEFS architecture proposed in this work is structured in such a way that most, if not all, of the listed algorithms can utilize it and benefit from it. This is an important aspect which speaks to the wide extensibility of the proposed methodology and architecture of CEFS.

Table 5: Categorical Framework of Feature Selection Algorithms [4]

			Search Strategies		
			Complete	Sequential	Random
Evaluation Criteria	Filter	Distance	B&B [95] BFF [124]	RELIEF [112] RELIEFF [113] RELIEFS [125] SFS [126] Segen's [127]	
		Information	MDLM [128]	DTM [129] Koller's [19] SFG [93] FCBF [130]	
		Dependency	Bobrowski's [131]	CFS [106] RRESET [132] POE+ACC [133] DVMM [134]	
		Consistency	Focus [111] ABB [135] MIFESI [136] Schlimmer's [137]	Set Cover [138]	LVI [93] QBB [93] LVF [139]
	Wrapper	Predictive Accuracy	BS [86] AMB&B [140] FSLC [141] FSBC [142]	SFS-SPLASH 13 WSFG [143] WSBG [143] BDS [86] PQSS [86] RC [144] SS [145] Queiros' [146]	SA [86] RGSS [86] LVW [98] RMHC-PF [147] GA [100] [101] RVE [148] ELSA [60]
	Hybrid	Filter + Wrapper		BBHFS [119] Xing [73]	

3.5 Common Misconceptions

Techniques such as dimensionality reduction, feature extraction, feature construction, feature weighting, feature creation and others are often misused as synonymous to feature selection. It should be pointed out that feature selection strictly selects a subset of the existing feature set, without the creation of new features [61] [149] [150]. However, to better understand feature selection, a brief explanation of some of the aforementioned techniques is noteworthy.

The first two techniques, dimensionality reduction and feature extraction, go hand in hand as various algorithms are often interchangeably characterized under both. Tan *et al* comments that dimensionality reduction commonly uses techniques from linear algebra to project the data from a high-dimensional space into a lower-dimensional space [18]. Similarly, Jain *et al* describes feature extraction algorithms as algorithms which create *new features* based on transformations or combinations of the original feature set [7]. For example, Principal Component Analysis (PCA) finds new attributes which are linear combinations of the original attributes, are orthogonal to each other, and capture the maximum amount of variation in the data [18]. In other words, PCA is an orthogonal transformation of the coordinate system, which is obtained by projection onto the *principal components*, or *features*, of the data. Since a small number of principal components are often sufficient, a reduction in dimensionality occurs [151]. Further reviews of PCA literature can also be found at [152] and [153].

Consider another example; suppose you have a set of photographs, where each photograph is to be classified as to whether or not it contains a human face. By using a feature extraction algorithm, pixels can be transformed to provide higher-level features, such as edges and areas highly correlated with the presence of human faces. Other examples include applying a Fourier transform to times series data to identify underlying frequencies in order to detect certain periodic patterns [18] and using Independent Component Analysis (ICA) in conjunction with Support Vector Machines (SVM) in order to improve prediction accuracy in series forecasting [154].

Other techniques such as feature weighting [18] and feature construction [22] also provide similar distinctions, but for the brevity of this research, will not be discussed at this time. In summary, feature selection is not to be confused with other techniques which go beyond the

original feature set, creating new features; that is, feature selection algorithms always yield a direct subset of the original feature set.

3.6 Relevance and Optimality of Features

Relevance, as it pertains to feature selection, is a bit of a loaded term. Defining it is not trivial nor is it widely agreed upon. In fact, numerous authors have provided a number of different definitions for relevance. A brief summary of some of these definitions will be provided in the subsequent paragraphs. Detailed explanations and further definitions of relevance can be found in works such as [61] by Kohavi and John, [107] by Blum and Langley, [72] by Nilsson *et al*, and most recently in [155] by Bell and Wang.

Although the word relevance is often used casually and without formal definition in most feature selection studies, its definition is as important as its use. At the heart of this matter lies the question of relevance vs. usefulness. Selecting only the most relevant features will often produce suboptimal results, especially if the features are redundant. Conversely, a subset of useful features may leave out many redundant, yet relevant, features [22]. In other words, the relevance of a feature does not imply that it should be in the optimal feature subset, while irrelevance does not imply it should not be in the optimal feature subset [61]. For clarification purposes, provided next are some formal definitions acquired from [61], [72], and [107] which may aid in understanding the role of relevance in feature selection.

Definition 2 (conditional independence) A variable X_i is conditionally independent of a variable Y given (conditioned on) the set of variables $S \subset X$ iff it holds that

$$P\big(p(Y|X_i, S) = p(Y|S)\big) = 1. \qquad \text{(Eq. 13)}$$

This is denoted $Y \perp X_i|S$.

Conditional independence is a measure of *irrelevance*, but it is difficult to use as an operational definition since it depends on the conditioning set S [72]. The definitions of strong and weak relevance, defined next, will help refine what it means for a feature to be considered irrelevant.

Definition 3 (strong relevance) A feature X_i is strongly relevant iff there exists some x_i, y, and s_i for which $p(X_i = x_i, S_i = s_i) > 0$ such that

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y \mid S_i = s_i). \quad \text{(Eq. 14)}$$

Definition 4 (weak relevance) A feature X_i is weakly relevant iff it is not strongly relevant and there exists a subset of features S_i' of S_i for which there exists some x_i, y, and s_i' with $p(X_i = x_i, S_i' = s_i') > 0$ such that

$$p(Y = y \mid X_i = x_i, S_i' = s_i') \neq p(Y = y \mid S_i' = s_i'). \quad \text{(Eq. 15)}$$

In other words, a feature X is strongly relevant if the removal of X alone will yield a decrease in the performance of an optimal classifier. Conversely, a feature X is weakly relevant if it is not strongly relevant and there exists a subset of features, S, such that the performance of a classifier on S is worse than the performance on $S \cup \{X\}$. Thus, a feature is *relevant* if it is either weakly or strongly relevant, otherwise it is *irrelevant* [61]. The importance lies not only with its definition, but research completed in [156] has also shown that algorithms such as C4.5 tend to generate deeper decision trees with lower performances when weakly relevant features are not deleted. Figure 17 provides a useful visual representation of the feature subset space divided into irrelevant, weakly relevant and strongly relevant feature subsets [61].

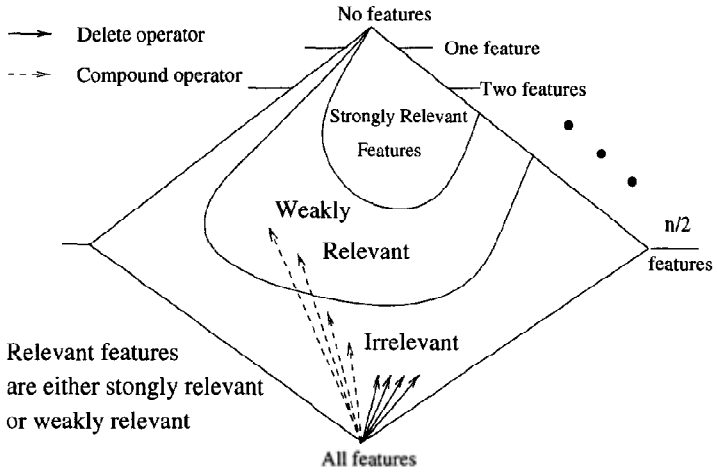

Figure 17: Feature Subset Space [61]

A Bayes classifier, for instance, must use *all* strongly relevant features and possibly some weakly relevant features (but recall that such classifiers are monotonic and have access to the underlying distribution). Classifiers induced from data, however, are likely to be suboptimal as they have no access to the same distribution. Accordingly, such induction algorithms may benefit from the omission of features, including strongly relevant ones [61]. Thus, it is important to define what it means for a feature subset to be optimal.

Definition 5 (optimal feature subset) Given a learning algorithm L and a dataset D, an optimal feature set, S^*, is a subset of features such that the predictive accuracy of D is highest under L.

As previously mentioned, however, relevant features do not imply that they are optimal, and conversely, optimal features do not imply that they are relevant. This is shown through Examples 1 and 2, acquired from [61].

Example 1 (relevance does *not* imply optimality) Let all possible instances be $\{0, 1\}^3$, that is, three Boolean features, say X_1, X_2, X_3. Let the distribution of instances be uniform, and assume the target concept is $f\ (X_1, X_2, X_3) = (X_1 \wedge X_2) \vee X_3$. Under any reasonable definition of relevance, all features are relevant to this target function. If the hypothesis space is the space of

monomials, i.e., conjunctions of literals, the only optimal feature subset is $\{X_3\}$. Moreover, the accuracy of the monomial X_3 is 87.5%, the highest accuracy achievable within this space and adding another feature to the monomial will simply decrease its accuracy. Therefore, this example shows that relevance, even strong relevance, does not imply that a feature should be in an optimal feature subset.

Example 2 (optimality does *not* imply relevance) Assume feature X_i is always set to the value of 1. Under all definitions of relevance described above, this feature is irrelevant. Now consider a limited Perceptron classifier that has an associated weight with each feature and then classifies instances based upon whether the linear combination is greater than zero. Also assume that the threshold = 0. Given that X_i is always set to 1, the limited Perceptron is equivalent in representation power to the regular Perceptron. However, removal of all irrelevant features would remove that crucial feature. Since cases such as this are believed to be rare in practice, it still holds true that irrelevant features should generally be removed.

Recall, however, that an objective measure still needs to be selected in order to implement and deploy a feature selection algorithm. So, for the scope of this research, the usefulness of a feature set will provide the necessary objective function for subset comparison and selection. As such, the following definition derived from [157] provides this criteria.

Definition 6 (incremental usefulness) Given a sample of data D, a learning algorithm L, and a feature set S, feature x_i is incrementally useful to L with respect to S if the accuracy of the hypothesis that L produces using the feature set $\{x_i\} \cup S$ is better than the accuracy achieved using just the feature set S.

Accordingly, the definition of incremental usefulness will be the focal point for determining whether or not a feature should be included in the optimal feature subset. This is further apparent given the fact that the proposed algorithm later described in this research will employ a wrapper method, which tailors directly to the learning algorithm used to evaluate each feature subset. Therefore, features will be assessed according to their usefulness with respect to the learning algorithm rather than their overall relevance.

As a final illustration on the issue of relevance vs. usefulness, refer to an example provided in [61]. One of the artificial datasets (*m*-of-*n*-3-7-10) represents a symmetric target function, implying that all features should be ranked equally by any filtering method. However, Naïve-Bayes improves if a single feature (any one of them) is removed. Consequently, that is the precise motivation behind the selection of a wrapper method over a filter method in the implementation of the proposed architecture in this research (Chapter 4). Note, however, that relevance according to these definitions does not imply membership in the optimal feature subset, and that irrelevance does not imply that a feature cannot be in the optimal feature subset [61]. The underlying purpose among any of this is to design the architecture in such a way that it is allowed to choose usefulness over relevance.

3.7 Feature Selection in Unsupervised Learning

Although the use of feature selection in unsupervised learning will not be implemented in this research, a brief discussion of the subject is important given the amount of attention it has received [4] [158] [118] [159] [160]. Unsupervised learning, more specifically clustering [161] [162], groups instances based on the information describing the instance or based on their relationships. The goal is to obtain groups with instances similar or related to one another and different from instances in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better the clustering [158].

Wrapper methods used in conjunction with clustering, evaluate the goodness of a feature subset by the quality of the clusters resulted from applying the clustering algorithm on the candidate feature subset. A number of heuristic measures exist for estimating the quality of clustering results, such as cluster compactness, scatter separability, and maximum likelihood [4].

Further work on developing dependent criteria in feature selection for clustering can also be found in [59], [118], and [159]. Since filter methods don't depend on the use of a separate learning algorithm for evaluation, the methodology for using filter feature selection in unsupervised learning does not differ from that of supervised learning. Given the infancy of this research area, advances continue to occur.

3.8 Feature Selection in Supervised Learning

It has been shown that CART [30], ID3 [25], C4.5 [26], and nearest neighbor [44] algorithms suffer in the presence of irrelevant features. Naive Bayesian classifiers [163] [65], which assume independence of features given the instance label, also degrade when using correlated and redundant features [60]. Even learners such as SVMs, which are considered more robust in high-dimensional settings, can improve classification performance when combined with feature selection techniques [164] [80]. Feature selection is also of particular interest when using nonparametric classification methods such as *k* nearest neighbor [165], Parzen's windows [166], or more generally methods based on geometrical models that have a reputation for having high computational and storage costs [167].

Feature selection algorithms have also been widely combined with evolutionary computation algorithms in supervised leanring tasks. The research conducted in [168], for example, combines a Genetic Algorithm and a *k*-Nearest Neighbor method to identify genes which can jointly discriminate between different classes of cancer. This and many other applications of feature selection on supervised learning tasks, such as text categorization [108] or Parkinsonian sleep analysis [169], are extensively available but outside the scope of this particular project.

3.9 Ensemble Feature Selection

Feature selection methods discussed up to this point employ the use of a single classifier. An ensemble system, on the other hand, is composed of a set of multiple classifiers and performs classification by selecting from the predictions made by each of the classifiers [18]. Since wide research has shown that ensemble systems are often more accurate than any of the individual classifiers of the system alone [9] [10] [11], it is only natural that ensemble systems and feature selection would be combined at some point.

An effective way of generating a diverse, yet accurate, ensemble of base classifiers is to use ensemble feature selection [47]. To recall, theoretical and empirical research has shown that an efficient ensemble should consist not only of high accuracy classifiers, but classifiers which also err in different parts of the input space [9]. Providing different feature subsets allow base classifiers to make their classification errors in different subareas of the instance space. While feature selection algorithms attempt to find an optimal feature subset for the learning algorithm, ensemble feature selection has the additional goal of finding a set of feature subsets which will promote disagreement among the base classifiers [47] [170]. This is perhaps the most important point in understanding the motivation and goals of the research presented in this work. The approach proposed in Chapter 4 will attempt to provide the aforementioned disagreement among the base classifiers, but with a set of highly optimal class-specific feature subsets. The results are believed to be higher prediction accuracy and better domain understandability.

Various successful attempts have been made at implementing ensemble feature selection. In [171], Ho proposes a technique called Random Subspace Method (RSM), which randomly selects a predetermined number of features from the original feature set. This is repeated n times to create n feature subsets that are used to generate the n base classifiers used in the ensemble.

Ho goes on to show that RSM ensembles can reach effective results presumably because the lack of accuracy in the ensemble members is compensated for by the diversity created by the RSM [171]. In another implementation, Optiz begins by using RSM to generate an initial population of feature subsets, which he then uses and optimizes in his Genetic Algorithm (GA). Optiz uses GAs in conjunction with Neural Networks (NN) as a wrapper method to find optimal feature subsets, which he then uses as training data to build an ensemble model of NNs. Optiz comments that the initial population, acquired through RSM, was surprisingly good and produced better ensembles on average than the popular and power ensemble approaches of bagging and boosting [170]. Other ensemble feature selection studies have included implementations such as a Hill-Climber and Bayesian Classifiers with cross-validation integration [47], GA and *k* Nearest Neighbor classifiers [172] [173], sequential-search-based strategies and Bayesian Classifiers [174], and comparative ensemble feature selection studies such as [175].

Perhaps the most related work to the architecture proposed in this research is the work done by Vale *et al* in [176]. In that study, the authors implement a filter method with an emphasis on class-based feature selection to generate the feature subsets used by each base classifier. The use of a filter method is perhaps the most evident shortcoming of the implementation as wrapper methods have often shown to provide more accurate results. In addition, the authors implement a combination based ensemble integration technique, although the specific details of the combination technique are not provided. Depending on the specific combination technique, further improvement can be seen here as well.

Chapter 4

Class-specific Ensemble Feature Selection (CEFS)

The approach presented and explored in this research, Class-specific Ensemble Feature Selection (CEFS), provides a novel methodology in which to perform feature selection. Specifically, CEFS takes advantage of the notion that features found to be resilient for classifying each classification separately, may be different than the features found to be resilient for the overall dataset. This difference is crucial as there exists features that are optimal for classifying a specific classification, but which may not have been included in the overall best feature set. This loss of information becomes vital in classification tasks.

By performing feature selection in a class-specific manner, the algorithm becomes more robust, more effective, and more informative. Moreover, one of the key aspects of CEFS is its extensibility and flexibility in regards to machine learners and feature selection algorithms. In other words, CEFS is designed in such a way that most, if not all, machine learners and feature selection algorithms can be coupled with it.

The CEFS algorithm will focus mainly on improving prediction accuracy in classification problems. As an added benefit, the class-specific design will select features optimal to each separate classification, thereby providing more information and understanding in regards to the most useful features to each classification and to the model. To better understand the theory and design behind CEFS, the remainder of this chapter will provide the theoretical, architectural, and algorithmic frameworks for CEFS.

4.1 Theoretical Framework

Let $Y = \{Y_1, Y_2, \dots, Y_n\}$ represent the original set of features, with cardinality *n*, and let $I = \{I_1, I_2, \dots, I_m\}$ represent the original training set of instances, with cardinality *m*. Let $t = \{t_1, t_2, \dots, t_m\}$ represent the set of known target outcomes (i.e., classifications) for *I*. Then, $I_j = \{i_1, i_2, \dots, i_n, t_j\}$ represents the j^{th} instance in *I*, as an assignment of values projected onto *Y* and with target outcome t_j. Moreover, let *d* represent the desired number of features in a candidate subset *X*, such that $X \subseteq Y$. Now, let the feature selection evaluation criterion for the set *X* be denoted by *J(X)* and be measured with respect to *I*. Without loss of generality, consider a *higher* value of *J* to indicate a better feature set. Then, traditional feature selection seeks to find a subset $X \subseteq Y$, such that $|X| = d$ and

$$J(X) = \max_{Z \subseteq Y, |Z| = d} J(Z).$$

Now, let $t' = \{t'_1, t'_2, \dots, t'_k\}$ represent the set of *distinct* known target outcomes for *I*, where $t' \subseteq t$ and $k \leq m$. Furthermore, $\forall t'_k \in t'$ let $d_{t'_k}$ represent the desired number of features in the corresponding subset $X_{t'_k}$, such that $X_{t'_k} \subseteq Y$. Then, let the feature selection evaluation criterion for $X_{t'_k}$ be denoted by $J'\left(X_{t'_k}\right)$, but be measured w.r.t. to $I_{t'_k} = \left\{I_{t'_k,1}, I_{t'_k,2}, \dots, I_{t'_k,p}\right\}$ with cardinality *p*, where $I_{t'_k} \subseteq I$, $I_{t'_k,j} = \left\{i_{t'_k,j,1}, i_{t'_k,j,2}, \dots, i_{t'_k,j,n}, t_{t'_k,j}\right\}$, and $I_{t'_k,j} \in I_{t'_k} \Leftrightarrow t_{t'_k,j} = t'_k$. Without loss of generality, consider a *higher* value of J' to indicate a better feature set. Then, class-specific feature selection seeks to find a subset $X_{t'_k} \subseteq Y$, such that $\left|X_{t'_k}\right| = d_{t'_k}$ and

$$J'\left(X_{t'_k}\right) = \max_{Z' \subseteq Y, |Z'| = d_{t'_k}} J'(Z'), \forall t'_k \in t'.$$

Now, let $C(A)$ represent a learning classifier C using feature set A. Moreover, let $G\big(C(A)\big)$ represent the generalization evaluation criterion of C using feature set A. Without loss

of generality, consider a *higher* value of G to indicate better generalization for feature set A under classifier C. Also, let $E = \left\{C\left(X_{t_k'}\right), C\left(X_{t_k'}\right), \ldots, C\left(X_{t_k'}\right)\right\}$ represent a class-specific ensemble learning machine which predicts according to $C\left(X_{t_k'}\right)$ when the classifiers all agree and $C(X)$ when there is disagreement. Research has already shown that generally $G(X) \geq G(Y)$ and $|X| < |Y|$. The goal of this research is to show that generally $G(E) \geq G(X)$, and that for $X_{t'} = \left\{X_{t_1'} \cup X_{t_2'} \cup \ldots \cup X_{t_k'}\right\}$, $X_{t'} \not\subseteq X$ and $X_{t'} \not\supseteq X$.

4.2 Architectural Framework

Presented next is the procedural design of CEFS (Figure 18). First, assume dataset *D*, which contains *n* different classifications. CEFS will use an all-purpose subset generation method such as Sequential Backward Selection (SBS) [82] or a Genetic Algorithm (GA) [170] to produce candidate subsets. These subsets will then be evaluated using a dependent (i.e., wrapper) subset evaluation criterion to evaluate each candidate subset. Note that independent methods of evaluation are also applicable if chosen for use. This process is repeated until an optimal class-specific feature subset if found for each of the *n* distinct classifications in *D*.

In other words, the feature selection algorithm will be run *n* times, each time searching for the subset of features which will maximize the classification accuracy, not of the entire dataset *D*, but of one of the *n* classifications, thus producing *n* optimal *class-specific* subsets. This is done by evaluating the prediction accuracy of the classifier on a single classification at a time, instead of the prediction accuracy of the classifier on the entire test set (which groups all classifications together). To clarify, the training set and test set do not ever change, they still contain all of the instances as though the algorithm is going to predict as normal. The facet that

changes is the evaluation criterion. By using only the prediction accuracy of a single classification at a time, the algorithm is able to find an optimal set of class-specific features.

The *n* optimal class-specific feature subsets are coupled with the same type of classifier, and all are combined to form an ensemble machine. New and unseen instances are then sent to the ensemble machine for classification. If all class-specific classifiers agree on the classification of an instance, then that instance is classified accordingly. If any classifier disagrees with the remaining classifiers, the algorithm classifies the instance using the overall best feature subset for the entire dataset.

The notion behind this approach is that class-specific feature sets have superior discriminating power in classifying instances when in agreement, but have inferior discriminating power when in disagreement. Hence, a feature set that can generalize well to the entire dataset is used in such instances.

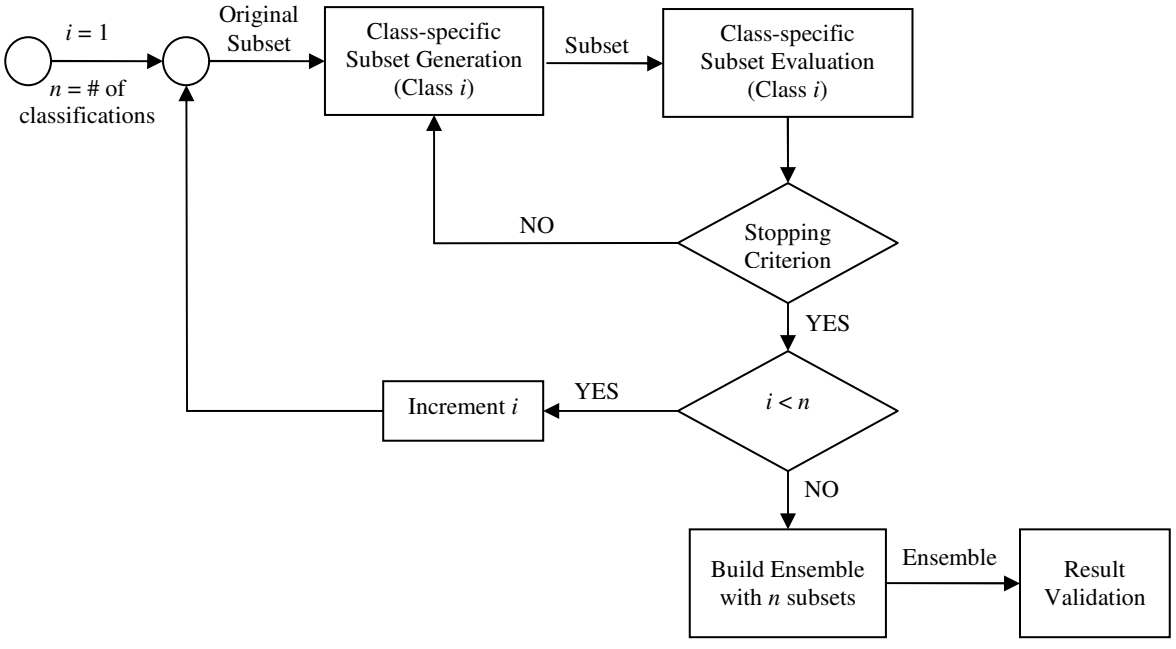

Figure 18: CEFS Procedure

4.3 Algorithmic Framework

It should first be noted that CEFS itself is not a feature selection algorithm, but rather procedural framework in which to conduct feature selection in classification tasks. Thus, it is classifier and feature selection algorithm independent. So, although a broad range of machine learning and feature selection algorithms can be utilized in a class-specific architecture, this section will discuss the algorithms which are implemented to test the hypothesis presented in this research.

Three different techniques are tested and used to generate candidate subsets for CEFS: Sequential Backward Selection (SBS), Sequential Forward Selection (SFS), and Genetic Algorithm (GA). The first two are greedy, sequential search algorithms, and are chosen because they follow a strictly deterministic pattern, so results are easily reproducible. However, for sufficiently large datasets, SBS and SFS are unfeasible in regards to run time. So, a third algorithm, a GA, is implemented to handle such datasets. GAs begin with a random sampling of the feature space and typically proceed in a stochastic manner, attempting to converge on an optimal solution. The implementation of a GA is not added solely for its scalability, however. It is of interest to this research to test the aforementioned CEFS architecture under two sets of structurally different subset generation techniques (greedy/deterministic vs. random/stochastic).

Additionally, two classifiers are implemented in this research to evaluate candidate feature subsets: a Naïve Bayesian Classifier (NBC) and a Discrete *k*-Nearest Neighbor Algorithm (*k*NN). Both classifiers are described in detail in Chapter 2 of this work. The implementation of these algorithms is well suited for this research as lazy learners tend to benefit greatly from feature selection [177]. Additionally, this implementation of CEFS evaluates candidate subsets using a wrapper method. Specifically, the subset evaluation criterion is the leave-one-out (LOO)

validation error from the classifier using the training set. In other words, suppose dataset D contains m instances, then the LOO validation error is the average error across m runs, where in each run, one of the m instances is removed from the training set to become the validation set, and then returned to the training set to be part of the training data for the next run. It is important to note that the classifier is retrained for each of the m runs, preventing the classifier model from having prior knowledge of the instance in the validation set. This approach has the advantage of utilizing as much data as possible for training (i.e., $m - 1$ instances). In addition, test sets are mutually exclusive and they effectively cover the entire dataset [18].

Once again, it is of the utmost importance to understand that CEFS is classifier and feature selection algorithm *independent*. Moreover, that the emphasis of this research is not on the robustness or efficiency of the feature selection algorithm or of the classifier, but rather on the notion that any chosen combination of the two can perform better when implemented in a class-specific architecture than when implemented in a traditional feature selection architecture. One of the reasons that SBS, SFS, NBC, and *k*NN are chosen is because they are all deterministic algorithms that provide an easier setting for reproducible results. Furthermore, *k*NNs are equipped to handle both discrete and continuous input data, allowing for a more thorough and diverse collection of datasets to be tested. Conversely, the GA is chosen so high-dimensional datasets could be handled and so that a stochastic method could also be tested under CEFS.

Chapter 5

Implementation

In order to gauge the usefulness and effectiveness of the CEFS framework and to test the aforementioned hypothesis in this text, an implementation of a Class-specific Ensemble Feature Selection architecture is presented here. As mentioned in Section 4.3, this implementation is tested using three different feature selection subset generators (SBS, SFS, and GA) and two different classifiers (NBC and *k*NN). Moreover, the implementation is tested over 10 datasets, varying in the number of instances, features, and classifications. It must be noted, however, that the NBC is only combined with the SBS algorithm and only tested over three of the ten datasets. This is due to the fact that the NBC/SBS combination is used mainly as a preliminary experiment to ascertain the potential performance benefits of CEFS and to begin exploring the difference in optimal feature sets between CEFS and traditional feature selection.

One of the other goals of this chapter is to show a proof of concept in regards to class-specific feature selection. That is, if the CEFS architecture can outperform a traditional feature selection architecture on different learners and under different selection algorithms, then it can be generalized that similar performance trends can be seen under any learner and any feature selection algorithm, making the CEFS architecture, algorithm independent.

So to achieve these goals, the remainder of the chapter will be arranged as follows. First, a brief reiteration of the proposed hypothesis is provided. Next, experimental data, tools, and procedure, algorithms, algorithmic specifications and algorithmic constraints are discussed.

Finally, the chapter concludes by presenting and detailing results regarding the performance and domain understandility of this experiment. Also, note that throughout the revealing of the experimental results, the aforementioned hypothesis will be discussed.

5.1 Proposed Hypothesis

As the implementation presented in this chapter will seek to test the hypothesis proposed earlier in Chapter 1, it is worthwhile to first revisit the aforementioned hypothesis. Recall that the following four part hypothesis is suggested:

1. There exists a recombination technique such that, when applied to a Class-specific Ensemble Feature Selection (CEFS) architecture, it will allow the system to outperform a traditional feature selection architecture in terms of prediction accuracy.

2. The same CEFS architecture will scale up by continuing to outperform the traditional feature selection architecture when the number of features are increased, as well as, when the number of class labels are increased.

3. The union of the set of relevant class-specific features may not be inclusive of the set of relevant features found traditionally for the entire dataset.

4. Conversely, the set of relevant features found traditionally for the entire dataset may not be inclusive to the union of the set of relevant features for each class label.

5.2 Experiment

The experiment conducted here implements a CEFS architecture using the aforementioned feature selection and classification algorithms and compares it against a traditional feature selection architecture utilizing the same algorithms. The subsequent sections describe the details surrounding this experiment.

5.2.1 Data

A wide variety of datasets are tested in this experiment. These datasets range in the number of instances (i.e., depth), number of features (i.e., width), number of classifications, and in whether or not they are missing values. Moreover, some datasets include strictly discrete input data, some strictly continuous input data, and some contain mixed input data. The notion behind the selection of datasets is to provide a collection that represents many varying dataset characteristics, thereby testing the algorithm under several possible scenarios. Table 6 contains the descriptive information specific to each dataset. A few additional comments need also be stated regarding the preparation of each dataset:

1. Each dataset is randomly sampled, based on a predetermined size, to create a training set and a test set. This task is completed 15 times for each dataset, thus creating 15 sets of randomly sampled data. The training sets are balanced such that they contain an equal number of instances from each classification[4]. For example, the Congressional Voting training set contains 268 instances, which can be classified as either "republican" or "democrat". This means that 134 instances have a classification of "republican" and 134 have a classification of "democrat". This is especially important since some of the algorithms implemented here have a strong statistical emphasis that could be negatively and unfairly affected by an unbalanced training set.

2. A leave-one-out validation scheme (as described in Section 4.3) is used during training to evaluate candidate k values for the kNN algorithm and to evaluate candidate feature subsets under each classifier.

3. Datasets with missing values were typically handled by replacing the missing values with the median feature value (if feature is discrete) or the mean feature value (if the feature is continuous). Further information specific to each dataset is provided in subsequent sections.

4. Most continuous datasets were normalized. In other words, each value in a feature was divided by the maximum absolute value in that feature. This normalizes all values between -1 and 1, if negative values exist, and between 0 and 1, if only positive values exist. Further information is also provided in subsequent sections.

[4] The Arrhythmia dataset is the only one containing an unbalanced training set, as the number of instances in one of the three classifications was simply too small for proper training.

Table 6: Dataset characteristics

Dataset Name	Total Number of Instances	Number of Features	Number of Class labels	Instances in Training	Instances in Testing	Missing Values
Breast Cancer	286	9	2	136	150	No
Wine	178	13	3	114	64	No
Credit Approval	690	15	2	490	200	Yes
Congressional Voting	435	16	2	268	167	Yes
Alabama County	340	16	2	188	152	Yes
Musk v1	476	166	2	248	228	No
Arrhythmia	452	279	3	235	217	Yes
Madaleon	2600	500	2	194	2404	No
Colon Cancer	62	2000	2	36	26	No
Lymphoma	42	4026	2	34	8	Yes

5.2.1.1 Breast Cancer Dataset

The Breast Cancer dataset is acquired from the UCI Machine Learning Repository [24]. It was originally created by Matjaz Zwitter and Milan Soklic of the Institute of Oncology at the University Medical Center, in Ljubljana, Yugoslavia. It contains 286 instances with two possible classifications: "recurrence" (85 instances) and "no-recurrence" (201 instances). Furthermore, it contains nine nominal features: age bracket, menopause, tumor-size bracket, inv-nodes, node-caps, deg-malig, breast side, breast-quad, and irradiat. Although it contains missing values, the dataset is left as is and classified as such.

5.2.1.2 Wine Dataset

The Wine dataset is acquired from the UCI Machine Learning Repository [24], however, it originates from [178]. It contains 178 instances with three possible classifications: "class 1" (59 instances), "class 2" (71 instances), and "class 3" (48 instances). Moreover, it contains 13

continuous attributes: alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, and proline. The dataset does not contain missing values.

5.2.1.3 Credit Approval Dataset

The Credit Approval dataset is also acquired from the UCI Machine Learning Repository [24], however, the original source is proprietary and confidential. It contains 690 instances with two possible classifications: "+" (307 instances) and "-" (383 instances). It also spans 15 features, six that are continuous and nine that are discrete. Feature names are also proprietary. The missing values in discrete features are replaced by the median value in the feature, whereas the missing values in continuous features are replaced by the mean value in the feature. All continuous features are normalized as previously mentioned in this chapter.

5.2.1.4 Congressional Voting Dataset

The Congressional Voting Records dataset is acquired from the UCI Machine Learning Repository [24] and originates from [179]. It contains 435 instances separated by two classifications: "republican" (168 instances) and "democrat" (267 instances). It also contains the following 16 Boolean valued (yes or no) features: handicapped-infants, water-project-cost-sharing, adoption-of-the-budget-resolution, physician-fee-freeze, El-Salvador-aid, religious-groups-in-schools, anti-satellite-test-ban, aid-to-Nicaraguan-contras, mx-missile, immigration, synfuels-corporation-cutback, education-spending, superfund-right-to-sue, crime, duty-free-exports, and export-administration-act-South-Africa. Missing values are denoted by "?" and are not replaced by anything.

5.2.1.5 Alabama County Jails Dataset

The Alabama County Jails dataset is a proprietary dataset acquired from a Regional County Corrections Department in Alabama. It is first introduced and described in [42]. The data obtained from the Alabama County Corrections Program consists 340 instances spanning 18 discrete attributes, which include the offender's personal information and current and previous charges and convictions. The end result, or predicted outcome, is determined by whether or not the individual completes the program (represented by a one signifying completion – 222 instances, or a negative one signifying failure to complete the program – 118 instances). Missing values are left as is and classified accordingly.

5.2.1.6 Musk (version 1) Dataset

The Musk v1 dataset is obtained from the UCI Repository [24], although it originates from the AI Group at Arris Pharmaceutical Corp., in San Francisco, CA. The dataset contains 476 conformations of molecules and it is divided into two classifications: 0 => "non-musk" (269 instances) or 1 => "musk" (207 instances). The instances are described by 166 features, all continuous and normalized as previously mentioned. No values are missing in this dataset.

5.2.1.7 Arrhythmia Dataset

The Arrhythmia dataset is acquired from the UCI Repository [24], but originally owned by Guvenir, Acar, and Muderrisoglu, and reported first on [180]. It contains 279 features, of which 206 are continuous and 73 are discrete. Although the original dataset is comprised of 16 different classifications, the classifications are organized such that they can be condensed to three. Class 01 refers to 'normal' ECG, classes 02 to 15 refer to different classes of arrhythmia,

and class 16 refers to the rest of unclassified instances. So, the adjusted dataset contains three classifications: “N” (245 instances), “A” (185 instances), and “U” (22 instances). Moreover, missing values in discrete features are replaced by the median value in the feature, whereas the missing values in continuous features are replaced by the mean value in the feature. Continuous features are then normalized as previously described. It is also noteworthy to mention that this is the only dataset without a balanced training set. Since there exists only 22 instances classified as “U”, a balanced training set would prove inappropriate for training as the number of instances would be far too few. So, the training set is comprised of 111 instances classified as “N”, 111 classified as “A”, and 13 classified as “U”. The remaining instances are placed in the test set.

5.2.1.8 Madelon Dataset

Although the Madelon dataset is also acquired from the UCI Repository [24], it stems from the Feature Selection Challenge at the 2003 Neural Information Processing Systems Conference. The data is synthetic and generated by conference organizers. The dataset contains 2600 instances with binary classification: “-1” (1300 instances) and “1” (1300 instances). All 500 features are continuous and normalized, and there are no missing values.

5.2.1.9 Colon Cancer Dataset

The Colon Cancer dataset originates from [13] as a well known microarray dataset. It contains the expression of 2000 genes (i.e., features) across the 62 tissues (i.e., instances). The identity (i.e., classification) of each instance is represented by either a -1 => “no cancer” (40 instances) or a 1 => “cancer” (22 instances). No values are reported missing and all 2000 features are continuous, and thus normalized according to the previously mentioned process.

5.2.1.10 Lymphoma Dataset

The B-Cell Lymphoma dataset stems from [181] and can be acquired from *http://llmpp.nih.gov/lymphoma*. It is comprised of gene expression data utilized to diagnose Diffuse large B-cell lymphoma (DLBCL). The disease is an aggressive malignancy of mature B lymphocytes, with an annual incidence of over 25,000 cases, accounting for roughly 40% of cases of non-Hodgkin's lymphoma. The dataset contains 42 instances which can be classified as "GC_B-Like" (21 instances) or "Activated_B-like" (21 instances). Moreover, instances are described across 4026 continuous features. The original data is first "filtered" and log-transformed (base 2) [181]. Then, the data is standardized so that the mean and standard deviation are 0 and 1, respectively [168]. Missing values are replaced with the feature's mean value to keep column's mean and standard deviation intact.

5.2.2 Tools

The datasets used in this experiment are stored and accessed using SQL Maestro for MySQL. The CEFS algorithm is written using Java JDK 1.6 as the programming language of choice and jGRASP as the IDE.

5.2.3 Procedure

This section will present the procedural organization of this experiment beginning from a high-level and ending with low-level details. As aforementioned, the available classifiers for this experiment are NBC and *k*NN, and the available feature selection subset generators are SBS, SFS, and GA. The experimental process then begins by separating each dataset into 15 randomly sampled sets of training and testing data, as described in Section 5.2.1. As the first

phase of testing, three of the ten datasets (Breast Cancer, Credit Approval, and Congressional Voting) are deployed using the NBC/SBS combination. This is due, in part, to the reasons previously mentioned in the beginning of this chapter, but also because these three datasets contain strictly discrete input values, thus bypassing the need for discretization for the NBC.

Given the positive results attained from the first phase of testing, five of the ten datasets are then tested under *k*NN/SBS and *k*NN/SFS combinations. The five datasets include the aforementioned three used with NBC/SBS plus the Wine and Alabama County Jails datasets. These datasets are added because they contain discrete and continuous input data, and because the Wine dataset contains more than two distinct classifications.

Following further promising results from the second phase of testing, the remaining five datasets (Musk v1, Arrhythmia, Madelon, Colon Cancer, and Lymphoma) are added to the previous five and are all tested under the *k*NN/GA combination. Note that these five remaining datasets are not tested under SBS or SFS frameworks because the feature cardinality of each dataset makes runtime simply unfeasible. Conversely, the addition of these five datasets allows CEFS to be tested under medium and high-dimensional data, thereby testing the scalability of the framework.

In terms of lower level processing tasks, each run begins by selecting the dataset and the classifier. Once this is complete, the classifier is trained and a classification model is typically built. In the case of a NBC or a *k*NN, no formal model is built as they are both lazy learners. Instead, a *k*NN will take this time to find an optimal *k* for the classifier, using a leave-one-out validation scheme to measure performance. The available *k* values are discussed in the Algorithmic Constraints section of this chapter. Once the classifier is ready, the baseline accuracy is measured by applying the test set to the classifier.

The feature selection subset generator is then chosen. Using the traditional feature selection architecture, the overall best feature set is then found. Another measure of accuracy is then taken, but this time, using the overall best feature set. The class-specific ensemble machine then begins to be built.

First in the ensemble is the allocation of the same classifier used above for each ensemble module. Just as before, each module also searches for an optimal k if using a kNN classifier. However, the performance of each k is measured by the leave-one-out *class-specific* accuracy instead of the leave-one-out overall accuracy of the training set. Once the classifiers are ready, each ensemble module then uses the same feature selection subset generator to find an optimal *class-specific* feature subset. Just as before, this means measuring the performance of each candidate subset by calculating the accuracy of the *class-specific* instances only. As soon as all class-specific subsets have been found, the system is ready to classify the test set.

Each instance in the test set is then sent to the ensemble machine. Each ensemble module classifies the instance separately and sends its classification to the decision phase. If all modules classify the instance the same way, the ensemble machine uses that as its classification. If any module disagrees with the others, the test instance is sent back out to the classifier using the traditionally acquired overall best feature subset for the tie-breaker. The instance is then classified according to the latter classifier. The three test set accuracies are then printed and compared, along with the overall best feature subset and the class-specific best feature subsets.

5.2.4 Procedure Validation

Two steps are taken to validate the procedure and algorithms mentioned in the previous section. First, each algorithmic combination implemented in this experiment is traced, step by

step, using the Play Tennis dataset (Table 2). This is possible through the use of the java debugger (also implemented in jGRASP), and due to the fact that the Play Tennis dataset is sufficiently small to warrant manual calculations. Accordingly, all calculations are first processed by hand and then validated through the IDE's debugger.

The second step also requires the use of the IDE's debugger. This step randomly selects instances from the training and test sets and once again computes and compares the manual calculations to the ones provided by the algorithm at runtime. Although this step is conceptually similar to the first step, recall that the Play Tennis dataset is strictly discrete, so this step allows for instances which are strictly continuous or of a mixed combination to also be validated.

5.2.5 Algorithms

As previously detailed in Section 4.3, two classifiers (NBC and *k*NN) and three feature selection subset generators (SBS, SFS, and GA) are implemented in this work. Although the NBC, SBS, and SFS algorithms have been sufficiently described to allow reproducible results, the other two algorithms, *k*NN and GA, need further description in regards to specific implementation details in order to warrant the similar results.

5.2.5.1 Algorithm Procedural Specifications

As commonly seen in most *k* Nearest Neighbor implementations, the algorithm implemented here utilizes the Euclidean distance to measure likeness between instances. Specifically, the distance between two points for a continuous feature is computed as described in Section 2.4.4 of this work. Conversely, the distance between two points for a discrete feature is computed as either 0 (if the values are the same) or 1 (if the values are different). Moreover,

the normalization or standardization of continuous features ensures that the calculated distance in those features also ranges between 0 and 1. Finally, it should be noted that if the classifier encounters a tie in the k nearest classifications, the final prediction is selected as the classification of the closest instance to the test/validation instance.

Several notes need to also be made in regards to the specifications of the GA. First, the initial randomly sampled population is generated based on percentages of the overall number of features. In other words, 1/5 of the initial population contains roughly 10% of the features, 1/5 contains 20%, then 40%, 60%, and 80%, respectively. This creates a higher amount of diversity in the initial population as the number of features ranges broadly. Second, candidate feature subsets with a higher leave-one-out validation accuracy (LOOVA) are deemed to be better (i.e., more fit). Moreover, if two candidate subsets yield the same LOOVA, then the one with the smaller number of features is deemed to be better.

Parent selection is implemented using binary tournament selection, where two different candidate parents are randomly selected and the more fit of the two becomes one of the parents chosen for procreation. This occurs μ times, where μ represents the number of candidate subsets in the population. Offspring are then created using crossover and mutation based on the probabilities detailed in the subsequent section. Finally, a generational selection approach is implemented, where the newly created offspring replace the parents. It should also be noted that the highest ranking parent can survive each generation based on the probability for elitism also detailed in the next section.

5.2.5.2 Algorithm Constraints

Two of the algorithms implemented in this experiment, *k*NN and GA, require constraints to be set or optimized before the algorithm is ready to be deployed. In the case of the *k*NN, a *k* value is optimized for the overall classifier and a different one is optimized for each class-specific ensemble module. During the preliminary stages of this research, the potential list of *k* values were as follows:

> *k*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 25, 0.25**n*, 0.5**n*, and 0.75**n*
> (where *n* is the number of instances in the dataset and such that $k < n$)

However, it was observed that some of the class-specific ensemble modules were selecting *k* such that the classifier simply became a recaller, with no regards for precision. In other words, the class-specific ensemble modules exhibiting this trend classified every instance as its class-specific classification, highly overfitting its training data and poorly generalizing to test data. To adjust for these occurrences, *k* values greater than 10 were removed from the list of potential *k* for class-specific modules.

Genetic Algorithms (GA) also require the optimization of constraints in order to perform at its peak capability. Most importantly in a GA, or in any Evolutionary Computation algorithm, is that candidate solutions converge toward a global optimum instead of getting trapped in local optimum or just jumping around the search space with little to no improvement. So, as a starting point, the following constraints were acquired from [101]:

> Population size (i.e., number of instances): 50
> Number of generations (i.e., number of runs): 20
> Probability of crossover: 0.6
> Probability of mutation: 0.001
> Probability of selection of highest ranked individual (i.e., elitism): 0.6

However, the above settings did not allow for successful convergence in the majority of datasets. So, the constraints were modified until convergence was observed. For all datasets, except Colon Cancer and Lymphoma, the new constraints were as follows:

Population size (i.e., number of instances): 20
Number of generations (i.e., number of runs): 40
Probability of crossover: 0.6
Probability of mutation: 0.001
Probability of selection of highest ranked individual (i.e., elitism): 0.8

It was observed that the algorithm needed less diversity in the initial population and more time to explore the search space to converge on a near optimal solution. Moreover, the higher elitism rate yielded greater selection pressure which also aided in convergence.

Given the significantly larger search space of the Colon Cancer and Lymphoma datasets, the population size is doubled and the elitism rate is raised to 0.9. Although the number of features in both datasets would normally make this unfeasible in terms of time, the small number of instances in the dataset allows for it to be implemented.

5.2.6 Results

The results in this section provide a basis for comparative analysis in two areas: performance and domain understandability. The results are also utilized to test the hypothesis previously presented in this text. Individual run results are provided in Appendices A and B.

5.2.6.1 Performance

The central performance metric utilized to compare architectures in this study is the average prediction accuracy obtained from the 15 randomly sampled test sets of each dataset. The average values are presented in subsequent tables and figures in this section. Refer to

Appendix A for a full list of accuracy values for each individual run. The first set of measures are provided in Table 7 and depicted in Figure 19. They contain the average accuracies using the NBC/SBS combination for classification. For these, and the remaining results in this section, a higher accuracy represents better performance. Note that CEFS outperforms traditional feature selection in all three datasets. Although this trend continues throughout the remaining results, these results already show proof that part 1 of the hypothesis can be accepted. Specifically, an effective recombination technique has been found which allows the CEFS architecture to outperform a traditional feature selection architecture in terms of prediction accuracy.

Table 7: Tabularized comparison of feature selection architectures based on dataset accuracy averages (under NBC/SBS algorithms)

Dataset	No Feature Selection	Traditional Feature Selection	CEFS
BREAST CANCER	66.44	68.89	***69.20***
CREDIT APPROVAL	***85.33***	83.77	84.23
CONGRESSIONAL VOTING	89.74	94.49	***94.61***

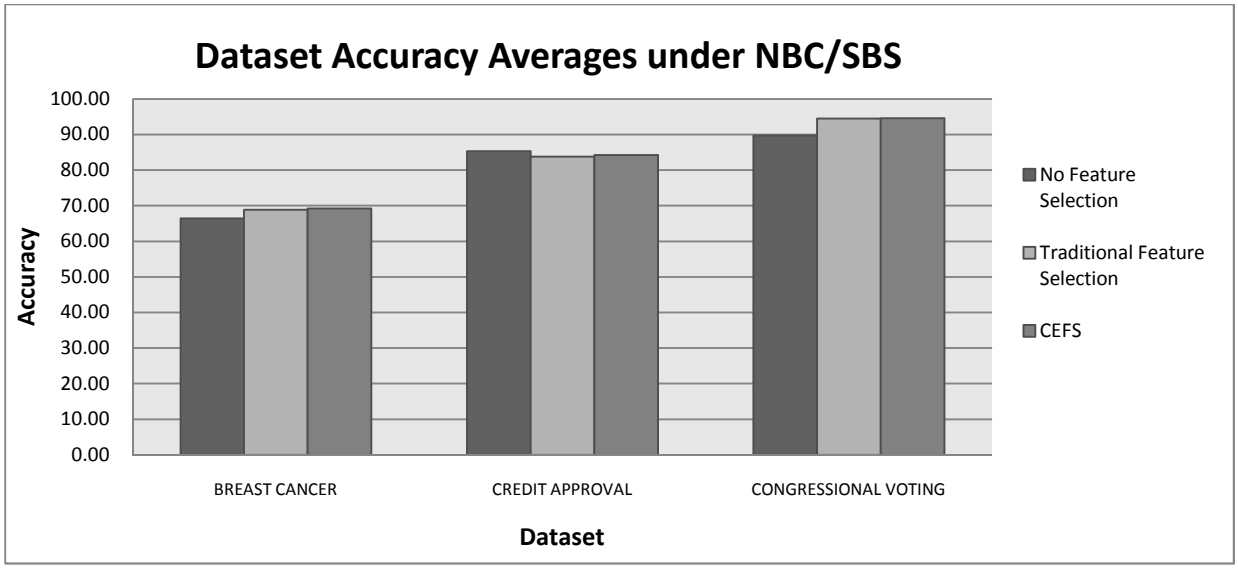

Figure 19: Graphical comparison of feature selection architectures based on dataset accuracy averages (under NBC/SBS algorithms)

Also, note that the accuracy of the Credit Approval dataset under the architecture without feature selection is actually higher than either architecture containing feature selection (Table 7 and Figure 19). Although this is not common and happens rarely in this study, it is a clear reminder that there is no silver bullet in machine learning and that there are certain cases where feature selection can actually hinder performance. As will be repeatedly shown in this study, however, this is much more the exception than the norm.

Presented next in Table 8 and Table 9, and in Figure 20 and Figure 21, are the results from the combinations of a *k*NN/SBS and a *k*NN/SFS. Much like the previous results, the CEFS architecture continues to outperform the traditional feature selection architecture in all datasets.

Table 8: Tabularized comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/SBS algorithms)

Dataset	No Feature Selection	Traditional Feature Selection	CEFS
BREAST CANCER	77.20	78.44	***81.64***
WINE	***94.06***	93.44	93.75
CREDIT APPROVAL	74.40	84.47	***85.50***
CONGRESSIONAL VOTING	91.30	94.01	***94.53***
MONTGOMERY COUNTY	49.52	62.37	***63.42***

Table 9: Tabularized comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/SFS algorithms)

Dataset	No Feature Selection	Traditional Feature Selection	CEFS
BREAST CANCER	77.20	80.89	***83.64***
WINE	***94.06***	91.56	92.92
CREDIT APPROVAL	74.40	84.23	***85.20***
CONGRESSIONAL VOTING	91.30	94.13	***94.49***
MONTGOMERY COUNTY	49.52	66.36	***67.50***

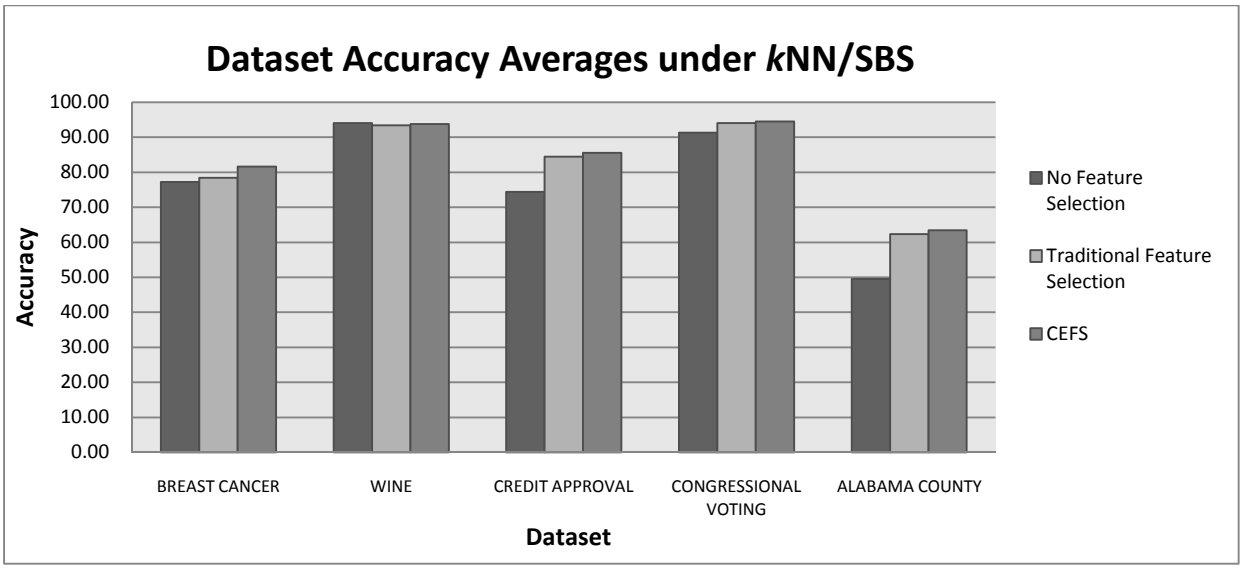

Figure 20: Graphical comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/SBS algorithms)

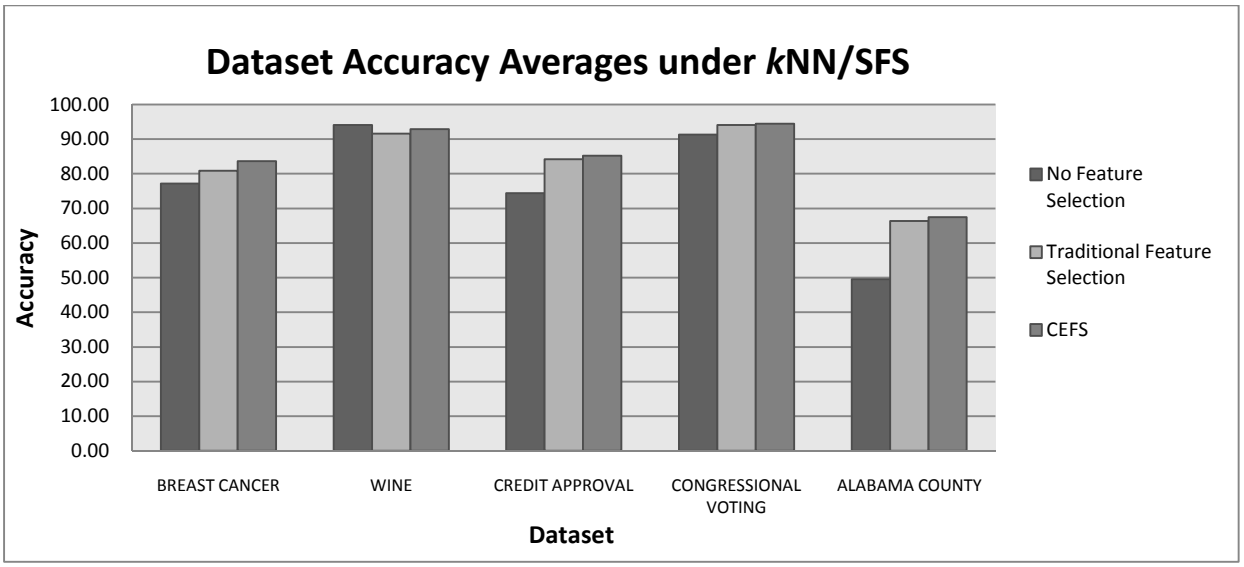

Figure 21: Graphical comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/SFS algorithms)

Next, the results from the *k*NN/GA combination are presented in Table 10 and in Figure 22. The overall performance improvement trend seen in the previous algorithmic combinations is also seen here, as CEFS outperforms traditional feature selection and no feature selection in all

10 datasets. Additionally, the accuracy improvements seen on the five large datasets and the two multi-classification datasets suggest that part #2 of the hypothesis can also be accepted. In other words, CEFS may successfully scale up using the framework presented here.

Table 10: Tabularized comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/GA algorithms)

Dataset	No Feature Selection	Traditional Feature Selection	CEFS
BREAST CANCER	77.20	79.96	***81.24***
WINE	94.06	94.69	***95.42***
CREDIT APPROVAL	74.40	85.20	***85.50***
CONGRESSIONAL VOTING	91.30	93.93	***94.09***
ALABAMA COUNTY	49.52	62.37	***64.17***
MUSK V1	78.77	80.06	***80.18***
ARRHYTHMIA	67.10	68.36	***69.59***
MADELON	61.32	62.40	***62.82***
COLON CANCER	65.90	67.69	***68.97***
LYMPHOMA	75.83	75.00	***77.50***

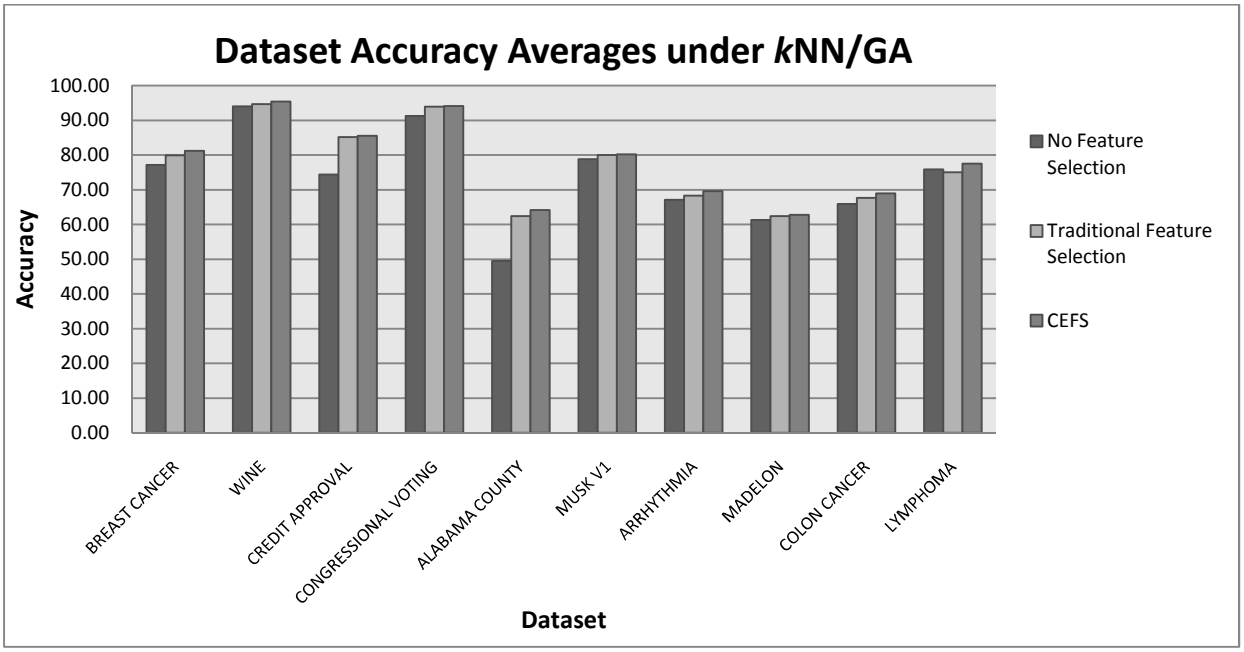

Figure 22: Graphical comparison of feature selection architectures based on dataset accuracy averages (under *k*NN/GA algorithms)

As shown by the results in this section, choosing an effective recombination technique for the ensemble system allows CEFS to become a better performing architecture than existent architectures. Moreover, the performance improvements on datasets ranging from 9 features to over 4000 features show that CEFS can scale up. These findings support the validity of the first two parts of the hypothesis presented in this research. The next section will explore the remaining two parts.

5.2.6.2 Domain Understandability

One of the great advantages of feature selection is the improved data domain understandability afforded from removing irrelevant and redundant features. This key benefit allows domain experts to shift their focus strictly to the most resilient and discriminating features while also reducing model complexity. However, parts 3 and 4 of the hypothesis suggest that traditional feature selection methods may actually miss crucially relevant data, when all classifications are considered at once. To test parts 3 and 4, the best found subset was recorded after each run and for each dataset. The detailed list of these subsets is found in Appendix B.

So, consider the best found subsets for run #1 of the *k*NN/SFS algorithms, using the Congressional Voting dataset, and tabularized in Table 11. The feature subset data from this run alone provides the necessary information to accept parts 3 and 4 of the hypothesis. As shown, the best overall feature set includes features 4, 7, and 12. Conversely, the union of the class-specific best feature sets includes features 4, 5, 7, and 11. Since feature 11 is included in the union of the class-specific sets but not in the overall best set, it follows that the union of the class-specific subsets is not inclusive of the overall best set, thereby proving part 3 of the hypothesis. On the other hand, feature 12 is included in the best overall set, but not in the union

of the class-specific sets, thus showing that the overall best set is not inclusive of the union of the class-specific best sets and therefore proving part 4 of the hypothesis.

Table 11: Best found subsets for run #1 of *k*NN/SFS (using Congressional Voting dataset)

Class label	F 1	F 2	F 3	F 4	F 5	F 6	F 7	F 8	F 9	F 10	F 11	F 12	F 13	F 14	F 15	F 16
Overall	0	0	0	X	0	0	X	0	0	0	0	X	0	0	0	0
republican	0	0	0	X	0	0	X	0	0	0	0	0	0	0	0	0
democrat	0	0	0	X	X	0	0	0	0	0	X	0	0	0	0	0

If a feature is included in the overall best feature set, it should intuitively follow that the feature is relevant to predicting at least one of the classifications in the dataset. The proof and acceptance of part 4 of the hypothesis clearly disproves this notion. On the other hand, a feature relevant to a single classification should intuitively be relevant enough to appear in the overall best feature set. Once again, this notion is rejected as part 3 of the hypothesis is accepted.

The question then becomes of whether this improves or diminishes the understandability of the data domain. The answer is that compelling arguments can be made for both cases. On one hand, the union of the best class-specific subsets will generate a higher number of features as features relevant for one classification may be significantly different than the features relevant to another. This may in turn increase model complexity and decrease understandability.

On the other hand, by performing feature selection in a class-specific manner, features can be isolated to individual classifications, thus highly increasing and improving classification understandability, a characteristic not available on traditional feature selection architectures. Moreover, certain classification tasks may wish to focus on one classification over another, a task that can be implemented in a CEFS architecture, but not in a traditional feature selection architecture.

Chapter 6

Conclusion

As discussed in Chapter 1: Introduction, the significant growth in data storage and the widening disparity between data processing and data acquisition have produced ample reasons to employ feature selection techniques when performing machine learning tasks. By effectively removing irrelevant and redundant features from the dataset, such techniques provide essential benefits such as improved learner accuracy, improved domain understandability, decreased model complexity, decreased training time, decreased required storage, etc. As the potential for increased benefit is significant, the research subsequently provides in depth discussions on Classification and Feature Selection in Chapters 2 and 3, respectively.

Given the clear motivation and objectives presented in Chapters 1 – 3, a novel class-specific ensemble feature selection architecture is proposed and comprehensively presented in Chapter 4. This architecture builds an ensemble of models, each with feature subsets optimized with respect to a distinct classification and under a specific base classifier. The proposed method seeks to outperform (in terms of prediction accuracy) a traditional feature selection architecture by providing an approach which brings sought after diversity and disagreement to the ensemble model while supplying the same model with feature subsets containing highly useful features to the base classifiers themselves.

So, to test the effectiveness of the aforementioned architecture and the hypothesis surrounding the potential benefits of such a design, a comprehensive experiment is constructed.

This experiment is presented in detail in Chapter 5 and consists of two classifiers, three feature selection subset generators, and ten datasets. The results obtained from the experiment solidify the potential benefits of a class-specific feature selection architecture and allow for the acceptance of the four part hypothesis proposed at the beginning of this study. In summary, a strong case is made for the implementation of a class-specific architecture when performing feature selection tasks.

Plenty of work remains to be completed however. One of the main drawbacks to a class-specific architecture is the ease in which an ensemble module can overfit the training data. This problem is exacerbated by the fact that the module can sometimes become a pure recaller, with no regards for precision (essentially classifying every instance it sees as the class-specific classification). Given the greedy nature of choosing the best performing class-specific subset, 100% recall means 100% prediction for that particular classification, which can clearly be a problem. So, the class-specific subset evaluation criterion must be balanced in the future, as precision must also be taken into account.

It should also be noted that although the CEFS architecture outperformed on average the traditional feature selection architecture, it most often did not do so with statistical significance. Specifically, p-values ranged from 0.04 to 0.91 using two-tailed t-tests. This was especially true as the search space of features scaled up. In other words, the stability of the architecture decreased as the number of features increased. Creating a more stable architecture would likely positively affect the overall performance, thus providing statistically significant results. Finally, it is important to continue to extend CEFS to other classifiers and other feature selection algorithms so that the architecture can be tested on a wider base of techniques and so that further benefits can be explored.

References

[1] G. Piatetsky-Shapiro, "Knowledge Discovery in Databases: 10 years later," *SIGKDD Explorations*, vol. 1, no. 2, pp. 59-61, Jan. 2000.

[2] Thomson Reuters. (2009, May) What Makes Information Intelligent?. [Online]. http://thomsonreuters.com/content/corporate/articles/What_makes_information_intel

[3] R. C. Craddock, P. Holtzheimer III, X. Hu, and H. Mayberg, "Disease state prediction from resting state functional connectivity," *Magnetic Resonance in Medicine*, vol. 62, no. 6, pp. 1619-1628, Oct. 2009.

[4] H. Liu and L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491-502, Apr. 2005.

[5] J. Alpert and N. Hajaj. (2008, Jul.) The Official Google Blog: We knew the web was big.... [Online]. http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

[6] C. Soares, L. Montgomery, K. Rouse, and J. E. Gilbert, "Automating Classification using General Regression Neural Networks," in *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, San Diego, CA, 2008, pp. 508-513.

[7] A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153-158, 1997.

[8] C. Soares, P. Williams, J. E. Gilbert, and G. Dozier, "A Class-Specific Ensemble Feature Selection Approach for Classification Problems," in *Proceedings of the 48th Annual Southeast Regional Conference*, Oxford, MS, April 15-17, 2010.

[9] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, no. 1, 2, pp. 105-139, 1999.

[10] T. G. Dietterich, "Ensemble Learning Methods," in *Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 2001.

[11] L. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993-1001, 1990.

[12] T. R. Golub, et al., "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science*, vol. 286, no. 5439, pp. 531-537, 1999.

[13] U. Alon, et al., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *PNAS*, vol. 96, no. 12, pp. 6745-6750, 1999.

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Pearson Education, 2003.

[15] P. Langley and H. A. Simon, "Applications of machine learning and rule induction," *Communications of the ACM*, vol. 38, no. 11, pp. 55-64, 1995.

[16] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Santa Fe Institute, Santa Fe, CA, Technical Report SFI-TR-95-02-010, 1995.

[17] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[18] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, 2006.

[19] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proceedings of the 13th International Conference on Machine Learning*, San Francisco, CA, 1996, pp. 284-292.

[20] J. Sinkkonen and W. S. Sarle. (2002, Oct.) Neural Network FAQ, part 2 of 7: Learning. [Online]. ftp://ftp.sas.com/pub/neural/FAQ2.html

[21] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, 1st ed. Chapman and Hall/CRC Press, 1986.

[22] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, 2003.

[23] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," in *Aide-Memoire of a Lecture at AMS Conference on Math Challenges of the 21st Century*, 2000.

[24] A. Asuncion and D. J. Newman. (2007) UCI Machine Learning Repository. [Online]. http://archive.ics.uci.edu/ml/

[25] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[26] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[27] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *International Conference on Machine Learning*, 2004.

[28] D. L. Olson and D. Delen, *Advanced Data Mining Techniques*. Springer, 2010.

[29] Wikipedia. (2010, Jun.) Precision and Recall. [Online]. http://en.wikipedia.org/wiki/Precision_and_recall

[30] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth International Group, 1984.

[31] E. Hunt, J. Martin, and P. Stone, *Experiments in Induction*. New York: Academic Press, 1966.

[32] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249-268, Jul. 2007.

[33] S. K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345-389, Dec. 1998.

[34] S. R. Safavin and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660-674, 1991.

[35] D. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*. Hoboken, NJ: John Wiley & Sons, 2005.

[36] W. Cohen, "Fast Effective Rule Induction," in *Proceedings of the International Conference on Machine Learning*, 1995, pp. 115-123.

[37] J. Furnkranz, "Separate-and-Conquer Rule Learning," *Artificial Intelligence Review*, vol. 13, pp. 3-54, 1999.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318-362.

[39] M. N. Siddique and M. O. Tokhi, "Training Neural Networks: Backpropagation vs. Genetic Algorithms," *IEEE International Joint Conference on Neural Networks*, vol. 4, pp. 2673-2678, 2001.

[40] G. Zhang, "Neural Networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 30, no. 4, pp. 451-462, 2000.

[41] L. Jiang, H. Zhang, and Z. Cai, "A Novel Bayes Model: Hidden Naive Bayes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1361-1371, Oct. 2009.

[42] C. Soares, L. Montgomery, C. Hamilton, and J. E. Gilbert, "Improving Accuracy in the Montgomery County Corrections Program Using Case-Based Reasoning," in *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, San Diego, CA, December 11 - 13, 2008, pp. 318-323.

[43] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of Improving K-Nearest Neighbor for Classification," in *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 2007, pp. 679-683.

[44] T. Cover and P. Hart, "Nearest Neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, 1967.

[45] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, Jun. 1998.

[46] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge: Cambridge University Press, 2000.

[47] A. Tsymbal, S. Puuronen, and D. Patterson, "Ensemble feature selection with the simple Bayesian classification," *Information Fusion*, vol. 4, pp. 87-100, 2003.

[48] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[49] C. Merz, "Dynamical selection of learning algorithms," in *Learnin from Data, Artificial Intelligence and Statistics*, D. Fisher and H. .-J. Lenz, Eds. NY: Springer-Verlag, 1996.

[50] M. Koppel and S. P. Engelson, "Integrating multiple classifiers by finding their areas of expertise," in *AAAI-96 Workshop On Integrating Multiple Learning Models*, 1996, pp. 53-58.

[51] A. Krogh and J. Vedelsby, "Neural Network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems*, D. Touretzky and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, vol. 7, pp. 231-238.

[52] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, vol. 9, no. 3, 4, pp. 385-404, 1996.

[53] C. Brodley and T. Lane, "Creating and exploiting coverage and diversity," in *AAAI-96 Workshop on Integrating Multiple Learned Models*, 1996, pp. 8-14.

[54] L. Asker and R. Maclin, "Ensembles as a sequence of classifiers," in *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, San Francisco, CA, 1997, pp. 860-865.

[55] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Information Processing Letters*, no. 24, pp. 377-380, 1987.

[56] J. Rissanen, "Stochastic complexity and modeling," *The Annals of Statistics*, no. 14, pp. 1080-1100, 1986.

[57] C. Wallace and P. Freeman, "Estimation and inference by compact coding," *Journal of the Royal Statistical Society*, no. 49, pp. 240-265, 1987.

[58] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, 1994, pp. 121-129.

[59] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in unsupervised learning via evolutionary search," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 365-369.

[60] Y. Kim, "Feature Selection in Supervised Learning and Unsupervised Learning via Evolutionary Search," University of Iowa Thesis, 2001.

[61] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence: Special Issue on relevance*, vol. 97, no. 1 - 2, pp. 273-324, 1997.

[62] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[63] D. W. Aha, D. Kibler, and M. K. Albert, "Instance Based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37-66, 1991.

[64] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992, pp. 223-228.

[65] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[66] I. J. Good, *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press, 1965.

[67] A. L. Blum and R. L. Rivest, "Training a 3-Node Neural Networks is NP-Complete," *Neural Networks*, vol. 5, pp. 117-127, 1992.

[68] E. Amaldi and V. Kann, "On the approximation of minimizing non zero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, pp. 237-260, 1998.

[69] A. Kalousis, J. Prados, and M. Hilario, "Stability of Feature Selection Algorithms," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, Houston, TX, 2005.

[70] A. Kalousis, J. Prados, and M. Hilario, "Stability of feature selection algorithms: a study on high-dimensional spaces," *Knowledge and Information Systems*, vol. 12, no. 1, pp. 95-116, 2006.

[71] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1 - 3, pp. 389-422, 2002.

[72] R. Nilsson, J. M. Pena, J. Bjorkegren, and J. Tegner, "Consistent Feature Selection for Pattern Recognition," *Journal of Machine Learning Research*, vol. 8, pp. 589-612, 2007.

[73] E. Xing, M. Jordan, and R. Karp, "Feature Selection for High-Dimensional Genomic Microarray Data," in *Proceedings of the 15th International Conference on Machine Learning*, 2001, pp. 601-608.

[74] F. K. Ahmad, N. M. Norwawi, S. Deris, and N. H. Othman, "A review of feature selection techniques via gene expression profiles," *International Symposium on Information Technology*, vol. 2, pp. 1-7, Aug. 2008.

[75] P. Langley, "Selection of relevant features in Machine Learning," in *AAAI Fall Symposium on Relevance*, 1994, pp. 140-144.

[76] R. Caruana and D. Freitag, "Greedy attribute selection," in *Machine Learning: Proceedings of the Eleventh International Conference*, 1994, pp. 28-36.

[77] G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289-1305, 2003.

[78] H. Lu, et al., "The Effects of Domain Knowledge Relations on Domain Text Classification," in *Proceedings of the 27th Chinese Control Conference*, Kumming, Yunnan, China, 2008, pp. 460-463.

[79] Q. Chen, "Feature Selection for the Topic-Based Mixture Model in Factored Classification," in *Proceedings of the 2006 International Conference on Computational Intelligence and Security*, 2006, pp. 39-44.

[80] E. Gabrilovich and S. Markovitch, "Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5," in *Proceedings of the 21st International Conference on Machine Learning*, 2004, pp. 41-48.

[81] A. Dasgupta, P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney, "Feature Selection Methods for Text Classification," in *Proceedings of the 13th SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, 2007, pp. 230-239.

[82] D. W. Aha and R. L. Bankert, "Feature selection for the case-based classification of cloud types: An empirical comparison," in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, 1994, pp. 106-112.

[83] M. Ben-Bassat, *Use of distance measures, information measures and error bounds in feature evaluation*, 2nd ed., P. R. Krishnaiah and L. N. Kanal, Eds. North-Holland Publishing Company, 1982.

[84] A. Miller, *Subset Selection in Regression*, 2nd ed. Chapman & Hall/CRC, 2002.

[85] N. R. Draper and H. Smith, *Applied Regression Analysis*, 2nd ed. John Wiley and Sons, 1981.

[86] J. Doak, "An evaluation of feature selection methods and their application to computer security," University of California at Davis, Technical Report CSE-92-18, 1992.

[87] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119-1125, 1994.

[88] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Patter Recognition*, vol. 33, pp. 25-41, 2000.

[89] Y. Kim, W. N. Street, and F. Menczer, "An evolutionary multi-objective local selection algorithm for customer targeting," in *Proceedings of Congress on Evolutionary Computation*, 2001, pp. 759-766.

[90] R. Kohavi and D. Sommerfield, "Feature subset selection using the wrapper model: Overfitting and dynamic search space topology," in *The First International Conference on Knowledge Discovery and Data Mining*, 1995, pp. 192-197.

[91] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 1 - 4, pp. 131-156, 1997.

[92] D. W. Aha and R. L. Bankert, "A comparative evaluation of sequential feature selection algorithms," in *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1995, pp. 1-7.

[93] H. Liu and H. Motoda, *Feature selection for Knowledge Discovery and Data Mining*. Springer, 1998.

[94] A. Arauzo-Azofra and J. M. Benitez, "Empirical Study of Feature Selection Methods in Classification," in *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, 2008, pp. 584-589.

[95] P. M. Narendra and K. Fukunaga, "A Branch and Bound Algorithm for Feature Subset Selection," *IEEE Transactions on Computers*, vol. 26, no. 9, pp. 917-922, Sep. 1977.

[96] Y. Hamamoto, S. Uchimura, Y. Matsunra, T. Kanaoka, and S. Tomita, "Evaluation of the Branch and Bound Algorithm for Feature Selection," *Pattern Recognition Letters*, vol. 11, pp. 453-456, Jul. 1990.

[97] G. Brassard and P. Bratley, *Fundamentals of Algorithms*. New Jersey: Prentice Hall, 1996.

[98] H. Liu and R. Setiono, "Feature Selection and Classification - a probabilistic wrapper approach," in *Proceedings of the 9th International Conference on Industrail and Engineering Applications of AI and Expert Systems*, Fukuoka, Japan, 1996, pp. 419-424.

[99] W. Siedlecki and J. Sklansky, "A Note on Genetic Algorithms for Large-Scale Feature Selection," *Pattern Recognition Letters*, vol. 10, pp. 335-347, Nov. 1989.

[100] H. Vafaie and I. F. Imam, "Feature Selection Methods: Genetic Algorithms vs. Greedy-Like Search," in *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, 1994.

[101] J. Yang and V. Honavar, "Feature Subset Selection Using a Genetic Algorithm," in *Feature Extraction, Construction and Selection: A Data Mining Perspective*, 1998, pp. 117-136.

[102] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[103] J. H. Holland, *Adaptation in natural and Artificial Systems*. Ann Arbor, IL: The University of Michigan Press, 1975.

[104] H. Almuallim and T. G. Dietterich, "Learning Boolean Concepts in the Presence of Many Irrelevant Features," *Artificial Intelligence*, vol. 69, no. 1-2, pp. 279-305, 1994.

[105] M. Ben-Bassat, "Pattern Recognition and Reduction of Dimensionality," in *Handbook of Statistics-II*, P. R. Krishnaiah and L. N. Kanal, Eds. North Holland, 1982, pp. 773-791.

[106] M. A. Hall, "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 359-366.

[107] A. L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," *Artificial Intelligence*, vol. 97, pp. 245-271, 1997.

[108] C. Apte, F. J. Damerau, and S. M. Weiss, "Automated Learning of Decision Rules for Text Categorization," *ACM Transactions on Information Systems*, vol. 12, no. 3, p. 233, 1994.

[109] D. D. Lewis, "Representation and learning in information retrival.," Department of Computer Science, University of Massachusetts, Amherst, Doctoral dissertation and Technical Report UM-CS-1991-093, 1992.

[110] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of Speech and Natural Language Workshop*, San Francisco, CA, 1992, pp. 212-217.

[111] H. Almuallim and T. G. Dietterich, "Learning with many irrelevant features," in *Proceedings of the Ninth National Conference on Artificial Intelligence*, San Jose, CA, 1991, pp. 547-552.

[112] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Tenth National Conference on Artificial Intelligence*, 1992, pp. 129-134.

[113] I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," *Machine Learning: ECML-94*, vol. 784/1994, pp. 171-182, 1994.

[114] G. Liu, L. Dong, S. Yuan, Y. Liu, and Y. Li, "New Feature Selection Algorithm based on Potential Difference," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, Harbin, China, 2007, pp. 566-570.

[115] S. B. Cho and H. H. Won, "Machine Learning in DNA Microarray Analysis for Cancer Classification," in *Proceedings of the First Asia-Pacific Bioinformatics Conference*, 2003.

[116] P. Langeley and S. Sage, "Oblivious Decision Trees and Abstract Cases," in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, Seattle, WA, 1994, pp. 113-117.

[117] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, "A Hybrid Feature Selection Mechanism," in *Eighth International Conference on Intelligent Systems Design and Applications*, 2008, pp. 271-276.

[118] M. Dash and H. Liu, "Feature Selection for Clustering," in *Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining*, 2000, pp. 110-121.

[119] S. Das, "Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection," in *Proceedings of the 18th International Conference on Machine Learning*, 2001, pp. 74-81.

[120] D. Lei, Y. Xiaochun, and X. Jun, "Optimizing Traffic Classification Using Hybrid Feature Selection," in *Proceedings of the Ninth International Conference on Web-Age Information Management*, 2008, pp. 520-525.

[121] M. Baglioni, B. Furletti, and F. Turini, "DrC4.5: Improving C4.5 by means of prior knowledge," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, CA, 2005, pp. 474-481.

[122] J. Mao, K. Mohiuddin, and A. K. Jain, "Parsimonious Network Design and Feature Selection Through Node Pruning," in *Proceedings of the International Conference on Pattern Recognition*, Jerusalem, 1994, pp. 622-624.

[123] G. Pagallo and D. Haussler, "Boolean feature discovery in empirical learning," *Machine Learning*, vol. 5, pp. 71-99, 1990.

[124] L. Xu, P. Yan, and T. Chang, "Best First Strategy for Feature Selection," in *Proceedings of the Ninth International Conference on Pattern Recognition*, 1988, pp. 706-708.

[125] H. Liu, H. Motoda, and L. Yu, "Feature Selection with Selective Sampling," in *Proceedings of the 19th International Conference on Machine Learning*, 2002, pp. 395-402.

[126] P. Pudil and J. Novovicova, "Novel Methods for Subset Selection with Respect to Problem Knowledge," in *Feature Extraction, Construction and Selection: A Data Mining Perspective*, H. Liu and H. Motoda, Eds. Springer, 2001, pp. 101-116.

[127] J. Segen, "Feature Selection and Constructive Inference," in *Proceedings of the Seventh International Conference on Pattern Recognition*, 1984, pp. 1344-1346.

[128] J. Sheinvald, B. Dom, and W. Niblack, "A Modeling Approach to Feature Selection," in *Proceedings of the 10th International Conference on Pattern Recognition*, 1990, pp. 535-539.

[129] C. Cardie, "Using Decision Trees to Improve Case-Based Learning," in *Proceedings of the 10th International Conference on Machine Learning*, 1993, pp. 25-32.

[130] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 856-863.

[131] L. Bobrowski, "Feature Selection Based on Some Homogeneity Coefficient," in *Proceedings of the Ninth International Conference on Pattern Recognition*, 1988, pp. 544-546.

[132] M. Modrzejewski, "Feature Selection Using Rough Sets Theory," in *Proceedings of the European Conference on Machine Learning*, 1993, pp. 213-226.

[133] A. N. Mucciardi and E. E. Gose, "A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition," *IEEE Transactions on Computers*, vol. 20, pp. 1023-1031, 1971.

[134] N. Slonim, G. Bejerano, S. Fine, and N. Tishbym, "Discriminative Feature Selection via Multiclass Variable Memory Markov Model," in *Proceedings of the 19th International Conference on Machine Learning*, 2002, pp. 578-585.

[135] H. Liu, H. Motoda, and M. Dash, "A Monotonic Measure for Optimal Feature Selection," in *Proceedings of the 10th European Conference on Machine Learning*, 1998, pp. 101-106.

[136] A. L. Oliveira and A. S. Vincentelli, "Constructive Induction Using a Non-Greedy Strategy for Feature Selection," in *Proceedings of the Ninth International Conference on Machine Learning*, 1992, pp. 355-360.

[137] J. C. Schlimmer, "Efficiently Inducing Determinations: A Completeand Systematic Search Algorithm that Uses Optimal Pruning," in *Proceedings of the 10th International Conference on Machine Learning*, 1993, pp. 284-290.

[138] M. Dash, "Feature Selection via Set Cover," in *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop*, 1997, pp. 165-171.

[139] H. Liu and R. Setiono, "A Probabilistic Approach to Feature Selection - A Filter Solution," in *Proceedings of the 13th International Conference on Machine Learning*, 1996, pp. 319-327.

[140] I. Foroutan and J. Sklansky, "Feature Selection for Automatic Classification of Non-Gaussian Data," *IEEE Transactions of Systems, Man, and Cybernatics*, vol. 17, no. 2, pp. 187-198, 1987.

[141] M. Ichino and J. Sklansky, "Feature Selection for Linear Classifier," in *Proceedings of the Seventh International Conference on Pattern Recognition*, 1984, pp. 124-127.

[142] M. Ichino and J. Sklansky, "Optimum Feature Selection by Zero-One Programming," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, no. 5, pp. 737-746, 1984.

[143] P. A. Devijver and J. Kittler, *attern Recognition: A Statistical Approach*. Prentice Hall International, 1982.

[144] P. Domingos, "Context Sensitive Feature Selection for Lazy Learners," *AI Revolution*, vol. 14, pp. 227-253, 1997.

[145] A. W. Moore and M. S. Lee, "Efficient Algorithms for Minimizing Cross Validation Error," in *Proceedings of the 11th International Conference on Machine Learning*, 1994, pp. 190-198.

[146] C. E. Queiros and E. S. Gelsema, "On Feature Selection," in *Proceedings of Seventh International Conference on Pattern Recognition*, 1984, pp. 128-130.

[147] D. B. Skalak, "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms," in *Proceedings of the 11th International Conference on Machine Learning*, 1994, pp. 293-301.

[148] D. J. Stracuzzi and P. E. Utgoff, "Randomized Variable Elimination," in *Proceedings of the 19th International Conference on Machine Learning*, 2002, pp. 594-601.

[149] J. Kittler, *Feature Selection and Extraction*. Academic Press, 1986.

[150] L. Rendell and R. Seshu, "Learning hard concepts through constructive induction: framework and rationale," *Computational Intelligence*, vol. 6, no. 4, pp. 247-270, Nov. 1990.

[151] B. Scholkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[152] I. T. Jolliffe, *Principal Component Analysis*. New York, NY: Springer-Verlag, 1986.

[153] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks*. New York: Wiley, 1996.

[154] L. J. Cao and W. K. Chong, "Feature Extraction in Support Vector Machine: A comparison of PCA, KPCA, and ICA," in *Proceedings of the 9th International Conference on Neural Information Processing*, 2002, pp. 1001-1005.

[155] D. A. Bell and H. Wang, "A formalism for relevance and its application in feature subset selection," *Machine Learning*, vol. 41, pp. 175-195, 2000.

[156] S. B. Thrun, et al., "The MONK's Problems: A Performance Comparison of Different Learning Algorithms," Carnegie Mellon University, Technical Report CMU-CS-91-197, 1991.

[157] R. A. Caruana and D. Freitag, "How useful is relevance?," in *Working Notes of the AAAI Fall Symposium on Relevance*, New Orleans, LA, 1994, pp. 25-29.

[158] M. Steinbach, L. Ertoz, and V. Kumar, "The challenges of clustering high-dimensional data," *New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition*, 2003.

[159] J. G. Dy and C. E. Brodley, "Feature Subset Selection and Order Identification for Unsupervised Learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 247-254.

[160] L. Talavera, "Feature Selection as a Preprocessing Step for Hierarchical Clustering," in *Proceedings of the International Conference on Machine Learning*, 1999, pp. 389-397.

[161] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, 1988.

[162] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

[163] P. Langeley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the 10th National Conference on Artificial Intelligence*, 1992, pp. 223-228.

[164] J. Weston, et al., "Feature Selection for SVMs," AT&T, 2001.

[165] P. E. Hart, "The Condensed Nearest Neighbor Rule," *IEEE Transactions on Information Theory*, pp. 515-516, 1968.

[166] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Annals of Mathematics and Statistics*, no. 33, pp. 1065-1076, 1962.

[167] M. Sebban, D. A. Zighed, and S. Di Palma, "Selection and Statistical Validation of Features and Prototypes," in *Principles of Data Mining and Knowledge Discovery*. Prague, Czech Republic: Springer, 1999, pp. 184-192.

[168] L. Li, C. Weinberg, T. Darden, and L. Pedersen, "Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method," *Bioinformatics*, vol. 17, no. 12, pp. 1131-1142, 2001.

[169] J. Fairley, G. Georgoulas, and G. Vachtsevanos, "Sequential Feature Selection Methods for Parkinsonian Human Sleep Analysis," in *Proceedings of the 17th Conference on Control and Automation*, Thessaloniki, Greece, 2009, pp. 1468-1473.

[170] D. W. Opitz, "Feature Selection for Ensembles," in *Proceedings of 16th National Conference on Artificial Intelligence*, 1999, pp. 379-384.

[171] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 832-844, 1998.

[172] C. Guerra-Salcedo and D. Whitley, "Genetic Approach for Feature Selection for Ensemble Creation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, FL, 1999, pp. 236-243.

[173] C. Guerra-Salcedo and D. Whitley, "Feature Selection Mechanisms for Ensemble Creation: A Genetic Search Perspective,," Data Mining with Evolutionary Algorithms: Research Directions. Papers from the AAAI Workshop., 1999.

[174] A. Tsymbal, P. Cunningham, M. Pechenizkiy, and S. Puuronen, "Search Strategies for Ensemble Feature Selection in Medical Diagnostics," in *Proceedings of the 16th IEEE Symposium on Computer-Based Medical Systems*, 2003, pp. 124-129.

[175] L. E. A. Santana, D. F. d. Oliveira, A. M. P. Canuto, and M. C. P. d. Souto, "A Comparative Analysis of Feature Selection Methods for Ensembles with Different Combination Methods," in .

[176] K. M. O. Vale, F. G. Dias, A. M. P. Canuto, and M. C. P. Souto, "A Class-Based Feature Selection Method for Ensemble Systems," in *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, 2008, pp. 596-601.

[177] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection," in *Proceedings of the 11th European Conference on Machine Learning*, 2000, pp. 109-116.

[178] M. Forina, R. Leardi, C. Armanino, and S. Lanteri. (1988) PARVUS: An extendable package of programs for data exploration, classification and correlation. Software Package.

[179] C. Q. Inc., *Congressional Quarterly Almanac, 98th Congress, 2nd session 1984*, Volume XL ed. Washington, D.C., 1985.

[180] H. A. Guvenir, B. Acar, G. Demiroz, and A. Cekin, "A Supervised Machine Learning Algorithm for Arrhythmia Analysis," in *Proceedings of the Computers in Cardiology Conference*, Lund, Sweden, 1997.

[181] A. Alizadeh, et al., "Distinct types of diffuse large B-cell lymphoma identifed by gene expression profling," *Nature*, vol. 403, pp. 503-511, Feb. 2000.

Appendices

Appendix A: Performance during Individual Runs

Note: the table headings for Appendix A are as follows:

- NBC – Naïve Bayesian Classifier
- *k*NN – *k* Nearest Neighbor algorithm
- SBS – Sequential Backward Selection
- SFS – Sequential Forward Selection
- GA – Genetic Algorithm
- No FS – Use of classifier alone, with no feature selection
- Trad. FS – Use of classifier with traditional feature selection architecture
- CEFS – Use of classifier with CEFS architecture
- Diff – Performance Difference between traditional feature selection architecture and CEFS architecture.

Lymphoma Dataset

LYMPHOMA (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	62.5	62.5	75	12.5
2	62.5	75	62.5	-12.5
3	75	87.5	87.5	0
4	87.5	100	100	0
5	62.5	75	87.5	12.5
6	87.5	87.5	87.5	0
7	75	62.5	50	-12.5
8	87.5	75	87.5	12.5
9	75	75	75	0
10	87.5	87.5	87.5	0
11	75	62.5	62.5	0
12	62.5	75	75	0
13	75	62.5	75	12.5
14	87.5	62.5	75	12.5
15	75	75	75	0
Avg	75.833	75	77.5	2.5

Colon Cancer Dataset

COLON CANCER (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	53.846	57.692	61.538	3.846
2	61.538	69.231	69.231	0
3	61.538	73.077	73.077	0
4	76.923	69.231	73.077	3.846
5	57.692	61.538	61.538	0
6	73.077	76.923	76.923	0
7	61.538	57.692	61.538	3.846
8	73.077	76.923	73.077	-3.846
9	76.923	73.077	73.077	0
10	61.538	65.385	69.231	3.846
11	69.231	69.231	65.385	-3.846
12	61.538	69.231	76.923	7.692
13	57.692	57.692	61.538	3.846
14	76.923	73.077	73.077	0
15	65.385	65.385	65.385	0
Avg	65.897	67.692	68.974	1.282

Madelon Dataset

MADELON (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	60.973	62.677	64.256	1.579
2	59.85	61.347	62.594	1.247
3	61.056	58.811	60.973	2.162
4	62.968	62.053	63.342	1.289
5	60.058	62.677	64.713	2.036
6	60.432	59.933	61.513	1.58
7	60.64	60.515	62.677	2.162
8	61.388	62.51	62.427	-0.083
9	60.848	66.874	63.425	-3.449
10	62.469	61.264	61.139	-0.125
11	59.518	60.682	61.305	0.623
12	61.222	63.549	63.092	-0.457
13	62.843	63.716	61.929	-1.787
14	62.552	66.002	64.381	-1.621
15	62.926	63.342	64.589	1.247
Avg	61.316	62.397	62.824	0.426867

Arrhythmia Dataset

ARRHYTHMIA (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	68.203	64.977	66.359	1.382
2	68.203	73.272	74.194	0.922
3	65.899	66.359	70.968	4.609
4	67.742	69.585	69.585	0
5	68.203	67.281	67.742	0.461
6	66.82	65.438	66.359	0.921
7	63.134	63.134	64.977	1.843
8	64.977	68.203	68.203	0
9	70.507	67.742	70.507	2.765
10	64.977	70.046	70.968	0.922
11	64.055	61.751	64.977	3.226
12	70.968	72.811	73.733	0.922
13	68.664	73.733	75.115	1.382
14	70.507	71.429	70.507	-0.922
15	63.594	69.585	69.585	0
Avg	67.097	68.356	69.585	1.228867

Musk (version 1) Dataset

MUSK V1 (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	82.018	81.579	80.263	-1.316
2	77.193	77.193	75	-2.193
3	80.263	82.456	80.702	-1.754
4	81.14	79.825	84.211	4.386
5	75.877	79.386	77.193	-2.193
6	79.386	84.211	85.526	1.315
7	78.509	82.018	82.018	0
8	76.316	79.825	81.14	1.315
9	73.684	77.193	79.385	2.192
10	78.947	84.211	82.456	-1.755
11	75	78.07	77.632	-0.438
12	77.193	75	75.439	0.439
13	83.772	80.702	83.333	2.631
14	79.825	78.07	77.193	-0.877
15	82.456	81.14	81.14	0
Avg	78.772	80.059	80.175	0.1168

Montgomery County Jails Dataset

MONTGOMERY COUNTY (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	50.658	44.737	55.263	10.526
2	46.053	84.211	71.711	-12.5
3	50	65.132	65.132	0
4	50.658	36.184	54.605	18.421
5	50	84.211	84.211	0
6	50	19.079	35.526	16.447
7	42.105	84.211	83.553	-0.658
8	40.789	36.842	48.684	11.842
9	50	65.789	54.605	-11.184
10	50	76.316	75	-1.316
11	56.579	73.684	73.684	0
12	56.579	59.211	66.447	7.236
13	44.079	57.895	43.421	-14.474
14	46.053	63.816	66.447	2.631
15	59.211	84.211	84.211	0
Avg	49.518	62.369	64.167	1.798067

MONTGOMERY COUNTY (kNN - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	50.658	64.474	69.737	5.263
2	46.053	69.737	66.447	-3.29
3	50	65.132	55.921	-9.211
4	50.658	65.132	75	9.868
5	50	84.211	84.211	0
6	50	55.263	57.895	2.632
7	42.105	71.711	68.421	-3.29
8	40.789	36.842	43.421	6.579
9	50	50.658	42.105	-8.553
10	50	59.868	60.526	0.658
11	56.579	57.237	68.421	11.184
12	56.579	59.211	59.211	0
13	44.079	48.684	50	1.316
14	46.053	63.158	67.105	3.947
15	59.211	84.211	82.895	-1.316
Avg	49.518	62.369	63.421	1.052467

MONTGOMERY COUNTY (kNN - SFS)				
Run	No FS	Trad. FS	CEFS	Diff
1	50.658	73.684	79.605	5.921
2	46.053	84.211	82.237	-1.974
3	50	65.132	65.132	0
4	50.658	57.895	68.421	10.526
5	50	84.211	84.211	0
6	50	84.211	78.947	-5.264
7	42.105	84.211	84.211	0
8	40.789	36.842	41.447	4.605
9	50	49.342	43.421	-5.921
10	50	73.026	71.053	-1.973
11	56.579	73.684	73.684	0
12	56.579	59.211	59.211	0
13	44.079	23.684	32.237	8.553
14	46.053	63.158	65.789	2.631
15	59.211	82.895	82.895	0
Avg	49.518	66.360	67.500	1.140267

Congressional Voting Dataset

CONGRESSIONAL VOTING (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	89.222	92.814	94.012	1.198
2	90.419	91.617	91.617	0
3	88.623	94.012	94.611	0.599
4	92.216	94.012	95.21	1.198
5	92.216	95.808	94.611	-1.197
6	91.018	92.216	91.018	-1.198
7	91.018	94.012	94.012	0
8	93.413	94.611	95.21	0.599
9	92.814	94.012	95.808	1.796
10	95.21	97.605	95.808	-1.797
11	92.216	95.21	94.012	-1.198
12	90.419	93.413	94.012	0.599
13	92.216	92.814	94.611	1.797
14	91.617	94.611	94.611	0
15	86.826	92.216	92.216	0
Avg	91.298	93.932	94.092	0.159733

CONGRESSIONAL VOTING (kNN - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	89.222	92.216	94.611	2.395
2	90.419	94.012	94.611	0.599
3	88.623	94.012	94.611	0.599
4	92.216	95.21	95.808	0.598
5	92.216	94.611	94.611	0
6	91.018	90.419	91.018	0.599
7	91.018	94.012	94.012	0
8	93.413	95.21	95.21	0
9	92.814	95.808	97.006	1.198
10	95.21	96.407	95.808	-0.599
11	92.216	92.216	94.012	1.796
12	90.419	94.611	94.012	-0.599
13	92.216	92.216	93.413	1.197
14	91.617	95.808	95.808	0
15	86.826	93.413	93.413	0
Avg	91.298	94.012	94.531	0.518867

CONGRESSIONAL VOTING (kNN - SFS)				
Run	No FS	Trad. FS	CEFS	Diff
1	89.222	94.012	94.611	0.599
2	90.419	95.21	94.611	-0.599
3	88.623	94.012	94.012	0
4	92.216	95.21	95.21	0
5	92.216	94.611	94.611	0
6	91.018	89.82	89.82	0
7	91.018	93.413	95.21	1.797
8	93.413	95.21	95.21	0
9	92.814	95.808	95.808	0
10	95.21	98.204	98.204	0
11	92.216	95.21	94.611	-0.599
12	90.419	94.611	94.012	-0.599
13	92.216	91.617	94.012	2.395
14	91.617	93.413	94.012	0.599
15	86.826	91.617	93.413	1.796
Avg	91.298	94.132	94.491	0.359267

CONGRESSIONAL VOTING (NBC - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	88.024	94.611	94.611	0
2	89.222	94.012	94.012	0
3	88.024	94.012	94.611	0.599
4	89.82	95.21	95.21	0
5	90.419	93.413	93.413	0
6	88.623	94.611	95.808	1.197
7	87.425	94.611	94.611	0
8	91.617	95.808	95.808	0
9	91.617	96.407	96.407	0
10	92.814	97.006	97.006	0
11	94.012	93.413	93.413	0
12	89.82	94.012	94.012	0
13	87.425	94.012	93.413	-0.599
14	89.82	93.413	94.012	0.599
15	87.425	92.814	92.814	0
Avg	89.740	94.491	94.611	0.119733

Credit Approval Dataset

CREDIT APPROVAL (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	76	84	83	-1
2	74	85	84.5	-0.5
3	79.5	88.5	89	0.5
4	72.5	92	90.5	-1.5
5	77	80.5	80	-0.5
6	69.5	88.5	86.5	-2
7	67.5	82.5	85	2.5
8	75.5	83.5	89	5.5
9	75.5	86.5	85.5	-1
10	75.5	86	87	1
11	75.5	80.5	81	0.5
12	75.5	86	86	0
13	73.5	85.5	84.5	-1
14	73.5	83	82.5	-0.5
15	75.5	86	88.5	2.5
Avg	74.4	85.2	85.5	0.3

CREDIT APPROVAL (kNN - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	76	85	85	0
2	74	84.5	83	-1.5
3	79.5	83.5	87	3.5
4	72.5	87.5	86	-1.5
5	77	83	84	1
6	69.5	79.5	83	3.5
7	67.5	81.5	84	2.5
8	75.5	89	88	-1
9	75.5	82.5	85.5	3
10	75.5	84.5	86.5	2
11	75.5	84.5	85.5	1
12	75.5	86	84.5	-1.5
13	73.5	84	85	1
14	73.5	84.5	87	2.5
15	75.5	87.5	88.5	1
Avg	74.4	84.46667	85.5	1.033333

CREDIT APPROVAL (kNN - SFS)				
Run	No FS	Trad. FS	CEFS	Diff
1	76	86.5	85.5	-1
2	74	79.5	83	3.5
3	79.5	88.5	89.5	1
4	72.5	87.5	89	1.5
5	77	81	83	2
6	69.5	84	84.5	0.5
7	67.5	82	82.5	0.5
8	75.5	89.5	88	-1.5
9	75.5	83	83	0
10	75.5	82.5	83	0.5
11	75.5	80	82	2
12	75.5	86	86.5	0.5
13	73.5	84.5	85.5	1
14	73.5	85	87.5	2.5
15	75.5	84	85.5	1.5
Avg	74.4	84.233	85.2	0.966667

CREDIT APPROVAL (NBC - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	85	85.5	85	-0.5
2	83	80.5	83	2.5
3	85	83.5	83.5	0
4	89	88	89	1
5	83	81.5	81.5	0
6	87	88.5	88.5	0
7	81.5	80.5	80.5	0
8	87	88	88.5	0.5
9	85	84.5	84.5	0
10	87	82	83.5	1.5
11	84	81.5	81.5	0
12	87.5	85.5	85.5	0
13	85.5	84	83.5	-0.5
14	85.5	80.5	81	0.5
15	85	82.5	84.5	2
Avg	85.333	83.767	84.233	0.466667

Wine Dataset

WINE (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	96.875	93.75	96.875	3.125
2	92.188	95.313	95.313	0
3	92.188	95.313	93.75	-1.563
4	96.875	93.75	95.313	1.563
5	90.625	90.625	90.625	0
6	96.875	93.75	93.75	0
7	93.75	96.875	98.438	1.563
8	95.313	89.063	90.625	1.562
9	95.313	95.313	95.313	0
10	95.313	98.438	100	1.562
11	90.625	93.75	93.75	0
12	92.188	98.438	98.438	0
13	92.188	89.063	92.188	3.125
14	95.313	98.438	98.438	0
15	95.313	98.438	98.438	0
Avg	94.063	94.688	95.417	0.729133

WINE (kNN - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	96.875	96.875	96.875	0
2	92.188	93.75	95.313	1.563
3	92.188	95.313	95.313	0
4	96.875	93.75	93.75	0
5	90.625	92.188	92.188	0
6	96.875	87.5	87.5	0
7	93.75	96.875	96.875	0
8	95.313	96.875	96.875	0
9	95.313	95.313	95.313	0
10	95.313	98.438	96.875	-1.563
11	90.625	89.063	89.063	0
12	92.188	87.5	89.063	1.563
13	92.188	90.625	92.188	1.563
14	95.313	92.188	93.75	1.562
15	95.313	95.313	95.313	0
Avg	94.063	93.438	93.750	0.312533

WINE (kNN - SFS)				
Run	No FS	Trad. FS	CEFS	Diff
1	96.875	92.188	92.188	0
2	92.188	81.25	92.188	10.938
3	92.188	90.625	90.625	0
4	96.875	89.063	90.625	1.562
5	90.625	89.063	92.188	3.125
6	96.875	93.75	95.313	1.563
7	93.75	96.875	100	3.125
8	95.313	89.063	90.625	1.562
9	95.313	92.188	93.75	1.562
10	95.313	95.313	95.313	0
11	90.625	93.75	90.625	-3.125
12	92.188	92.188	92.188	0
13	92.188	95.313	95.313	0
14	95.313	87.5	87.5	0
15	95.313	95.313	95.313	0
Avg	94.063	91.563	92.917	1.354133

Breast Cancer Dataset

BREAST CANCER (kNN - GA)				
Run	No FS	Trad. FS	CEFS	Diff
1	80.667	80.667	81.333	0.666
2	70.667	84	86.667	2.667
3	77.333	74.667	84	9.333
4	82	79.333	65.333	-14
5	84.667	82	83.333	1.333
6	78	81.333	81.333	0
7	82.667	82.667	79.333	-3.334
8	79.333	88.667	88.667	0
9	74.667	85.333	88.667	3.334
10	78.667	75.333	84	8.667
11	71.333	72	72.667	0.667
12	80	76.667	83.333	6.666
13	65.333	80.667	80.667	0
14	78	80	81.333	1.333
15	74.667	76	78	2
Avg	77.20007	79.9556	81.2444	1.2888

BREAST CANCER (kNN - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	80.667	74.667	76	1.333
2	70.667	82	87.333	5.333
3	77.333	77.333	87.333	10
4	82	78.667	80	1.333
5	84.667	88	88.667	0.667
6	78	82	80	-2
7	82.667	81.333	88	6.667
8	79.333	88.667	85.333	-3.334
9	74.667	78	80	2
10	78.667	75.333	88	12.667
11	71.333	72	71.333	-0.667
12	80	86.667	87.333	0.666
13	65.333	65.333	72.667	7.334
14	78	71.333	74.667	3.334
15	74.667	75.333	78	2.667
Avg	77.200	78.444	81.644	3.2

BREAST CANCER (kNN - SFS)				
Run	No FS	Trad. FS	CEFS	Diff
1	80.667	79.333	78.667	-0.666
2	70.667	84	86.667	2.667
3	77.333	77.333	85.333	8
4	82	88.667	80.667	-8
5	84.667	84.667	84.667	0
6	78	81.333	83.333	2
7	82.667	82.667	84	1.333
8	79.333	88.667	88.667	0
9	74.667	78	88.667	10.667
10	78.667	75.333	88	12.667
11	71.333	80	81.333	1.333
12	80	78	82.667	4.667
13	65.333	80.667	81.333	0.666
14	78	80	78	-2
15	74.667	74.667	82.667	8
Avg	77.200	80.889	83.645	2.7556

BREAST CANCER (NBC - SBS)				
Run	No FS	Trad. FS	CEFS	Diff
1	67.333	68	68.667	0.667
2	70	62	62.667	0.667
3	61.333	71.333	74	2.667
4	70.667	73.333	74.667	1.334
5	72.667	71.333	73.333	2
6	68.667	78	70.667	-7.333
7	64.667	75.333	73.333	-2
8	71.333	70.667	71.333	0.666
9	62	72	69.333	-2.667
10	67.333	65.333	66	0.667
11	67.333	68	69.333	1.333
12	64	65.333	65.333	0
13	57.333	63.333	70	6.667
14	62.667	64.667	64.667	0
15	69.333	64.667	64.667	0
Avg	66.444	68.889	69.200	0.3112

Appendix B: Best found overall and class-specific feature subsets per individual runs

Note 1: a zero value represents a feature that was not used, whereas, a non-zero value represents the index of a used feature.

Note 2: the best found subsets for the five larger datasets (Musk v1, Arrhythmia, Madelon, Colon Cancer, and Lymphoma) are not provided here as they are extremely large.

Breast Cancer Dataset
(NBC/SBS)
Overall: 0 0 0 0 5 6 0 0 0
no-recurrence-events: 0 2 0 0 5 0 0 0 0
recurrence-events: 1 2 3 0 0 6 0 0 9

Overall: 0 0 3 4 0 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 1 0 3 0 0 6 0 0 0

Overall: 0 0 0 0 0 6 0 8 0
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 0 0 3 0 0 6 0 8 9

Overall: 0 2 0 4 0 6 0 0 9
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 1 0 3 4 0 6 0 0 9

Overall: 0 0 0 4 0 0 0 8 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 3 0 0 6 0 8 9

Overall: 0 0 0 0 0 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 0 4 5 6 0 0 0

Overall: 0 2 0 0 0 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 2 3 0 5 0 0 0 9

Overall: 0 2 0 4 0 6 0 0 9
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 0 0 3 0 5 6 0 0 9

Overall: 0 2 0 0 0 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 1 2 0 0 0 6 0 0 0

Overall: 0 0 0 0 5 6 7 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 1 2 3 0 0 6 0 0 9

Overall: 0 0 0 0 5 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 1 0 3 0 5 6 7 8 0

Overall: 0 0 0 4 5 6 0 0 0
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 4 5 6 0 0 0

Overall: 0 0 0 4 5 6 0 8 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 3 4 0 0 0 0 9

Overall: 0 0 3 0 5 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 3 4 0 6 0 0 0

Overall: 0 2 0 0 0 6 0 0 9
no-recurrence-events: 0 0 0 0 0 0 0 0 9
recurrence-events: 0 0 0 4 0 6 0 8 9

(*k*NN/SBS)
Overall: 1 2 3 0 0 6 7 8 9
no-recurrence-events: 1 2 3 4 0 6 7 8 9
recurrence-events: 0 2 3 0 5 6 7 0 0

Overall: 0 0 0 0 5 0 0 0 0
no-recurrence-events: 0 0 0 4 0 0 7 8 0
recurrence-events: 0 2 3 0 0 6 0 8 0

Overall: 1 0 0 0 5 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 7 0 9
recurrence-events: 0 2 0 4 5 6 0 8 9

Overall: 0 0 0 0 5 6 7 0 0
no-recurrence-events: 1 0 3 4 5 0 0 0 0
recurrence-events: 0 0 0 0 5 6 0 0 0

Overall: 1 0 3 4 5 6 7 8 9
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0
no-recurrence-events: 0 0 0 4 0 0 7 8 9
recurrence-events: 0 0 3 0 0 0 7 0 0

Overall: 1 2 0 0 5 6 7 0 0
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 0 2 3 0 5 6 0 8 9

Overall: 0 0 0 0 0 6 0 0 0
no-recurrence-events: 1 2 0 4 0 6 7 8 9
recurrence-events: 0 0 0 0 0 6 0 8 9

Overall: 1 2 3 4 5 6 0 8 9
no-recurrence-events: 0 0 0 4 0 0 0 0 0
recurrence-events: 1 0 3 0 5 6 0 8 0

Overall: 0 2 0 0 0 6 0 0 0
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 1 2 3 0 5 6 7 8 9
no-recurrence-events: 1 2 3 4 5 6 7 8 9
recurrence-events: 1 0 3 0 5 0 0 0 0

Overall: 1 0 0 4 0 0 0 0 0
no-recurrence-events: 0 0 0 0 0 0 0 0 9
recurrence-events: 0 2 3 4 5 6 7 0 0

Overall: 0 2 3 0 5 6 7 0 9
no-recurrence-events: 0 0 0 4 5 0 0 8 0
recurrence-events: 1 2 3 4 0 6 7 0 9

Overall: 1 2 0 0 5 6 7 0 0
no-recurrence-events: 1 0 0 4 5 0 7 8 0
recurrence-events: 0 2 3 0 0 6 7 8 0

Overall: 0 0 0 4 5 6 0 8 9
no-recurrence-events: 0 0 0 4 0 0 0 8 0
recurrence-events: 0 0 0 0 0 6 0 0 0

(*k*NN/SFS)
Overall: 1 0 3 4 5 6 7 8 0
no-recurrence-events: 0 0 0 0 5 6 0 0 0
recurrence-events: 0 0 0 4 5 0 0 0 9

Overall: 0 0 0 4 0 0 0 0 0
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 0 0 6 0 0 0

Overall: 0 2 0 0 5 6 7 8 0
no-recurrence-events: 0 0 0 4 0 0 0 8 0
recurrence-events: 1 0 0 0 0 0 0 8 0

Overall: 0 0 0 4 0 6 0 0 0
no-recurrence-events: 0 2 0 4 0 6 7 8 9
recurrence-events: 1 0 0 0 0 0 0 0 0

Overall: 1 2 3 4 5 6 7 8 9
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 4 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 0
no-recurrence-events: 1 2 0 4 5 0 7 0 9
recurrence-events: 0 0 0 0 0 6 0 0 0

Overall: 1 2 3 4 5 6 7 8 9
no-recurrence-events: 0 2 0 4 5 6 0 0 9
recurrence-events: 0 0 0 0 0 6 0 0 0

Overall: 0 0 0 0 5 0 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 0 0 5 0 0 0 9

Overall: 1 0 3 0 5 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 0 0 0 0 0 0 9

Overall: 0 2 0 0 0 6 0 0 0
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0
no-recurrence-events: 1 2 3 4 5 6 7 8 9
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 0 0 3 4 0 0 0 0 0
no-recurrence-events: 0 0 0 0 0 0 7 0 0
recurrence-events: 0 0 0 0 0 0 0 0 9

Overall: 0 2 0 0 5 0 0 0 0
no-recurrence-events: 1 2 0 4 5 0 0 0 0
recurrence-events: 1 2 3 4 5 6 7 0 9

Overall: 0 0 3 0 5 0 0 0 0
no-recurrence-events: 1 0 0 0 5 0 0 0 0
recurrence-events: 0 0 3 0 5 0 0 0 0

Overall: 0 0 0 4 5 6 7 8 9
no-recurrence-events: 1 0 3 4 5 0 0 0 9
recurrence-events: 0 0 0 0 5 6 0 0 0

(*k*NN/GA)
Overall: 1 2 3 4 5 6 7 8 9
no-recurrence-events: 1 2 0 4 5 6 0 8 9
recurrence-events: 0 0 0 4 5 0 0 0 9

Overall: 0 0 0 4 0 0 0 0 0
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 0 0 0 6 0 0 0

Overall: 1 0 3 4 5 6 0 8 9
no-recurrence-events: 1 0 0 0 5 0 0 0 0
recurrence-events: 0 0 3 0 0 0 7 0 9

Overall: 1 0 0 0 5 6 0 0 0
no-recurrence-events: 0 2 0 0 0 0 0 0 0
recurrence-events: 0 0 0 0 5 6 0 0 0

Overall: 0 0 3 0 5 6 7 8 9
no-recurrence-events: 0 0 0 0 5 0 0 8 9
recurrence-events: 0 0 0 4 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 2 3 0 0 6 0 0 0

Overall: 1 2 0 4 0 6 0 0 9
no-recurrence-events: 0 0 0 4 0 6 7 0 9
recurrence-events: 0 0 0 0 0 6 0 0 0

Overall: 0 0 0 0 5 0 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 0 0 5 0 0 0 9

Overall: 0 0 0 4 5 6 0 0 0
no-recurrence-events: 0 0 0 0 5 0 0 0 0
recurrence-events: 0 0 0 0 0 0 0 0 9

Overall: 0 0 0 4 5 6 0 0 0
no-recurrence-events: 1 0 0 4 5 0 7 8 9
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 1 2 3 0 5 6 7 8 9
no-recurrence-events: 1 2 3 4 5 6 7 8 9
recurrence-events: 0 0 0 0 5 0 0 0 0

Overall: 1 0 3 4 5 6 7 0 0
no-recurrence-events: 0 0 0 0 0 0 7 0 0
recurrence-events: 0 0 0 0 0 0 0 0 9

Overall: 0 2 0 0 5 0 0 0 0
no-recurrence-events: 0 2 0 4 5 6 0 8 9
recurrence-events: 1 0 0 0 5 0 0 0 0

Overall: 0 0 3 0 5 0 0 0 0
no-recurrence-events: 1 0 0 4 0 0 0 0 0
recurrence-events: 0 0 3 0 5 0 0 0 0

Overall: 0 0 3 4 5 6 0 8 9
no-recurrence-events: 1 0 3 4 5 0 0 0 9
recurrence-events: 0 0 0 0 5 0 7 8 9

Wine Dataset
(*k*NN/SBS)
Overall: 1 2 3 4 0 0 0 0 0 0 10 11 12 13
1: 1 2 3 4 0 6 0 0 0 0 0 0 13
2: 0 2 3 0 0 0 0 0 0 10 0 0 13
3: 0 0 0 0 0 0 7 0 0 0 11 0 0

Overall: 1 0 0 0 0 0 7 0 0 10 0 12 13
1: 1 0 0 0 0 0 7 0 0 0 0 0 0
2: 1 2 0 4 5 0 7 0 0 10 0 12 13
3: 1 0 0 0 5 0 7 0 0 0 0 0 0

Overall: 0 2 0 0 0 0 7 0 0 10 11 12 13
1: 1 2 3 4 0 0 7 0 0 0 0 0 0
2: 0 0 0 0 5 0 0 0 0 10 0 0 0
3: 0 2 0 0 0 0 0 0 0 0 0 11 12 0

Overall: 1 2 3 4 0 0 7 0 0 10 0 0 13
1: 1 2 3 4 0 6 0 0 0 0 0 0 13
2: 1 2 3 4 0 0 7 8 0 10 0 12 13
3: 1 0 3 0 0 0 7 0 9 0 0 0 0

Overall: 0 2 0 4 5 0 7 8 9 10 0 0 13
1: 1 0 0 0 0 6 0 0 0 0 0 0 0
2: 0 0 0 4 5 0 7 0 0 0 0 0 13
3: 0 0 0 0 0 0 7 0 0 10 0 0 0

Overall: 1 0 3 4 0 6 0 0 0 0 0 12 13
1: 1 0 3 0 0 0 7 0 0 0 0 0 13
2: 1 0 3 4 0 6 0 0 0 0 0 12 13
3: 1 0 3 0 0 0 7 0 0 0 0 0 0

Overall: 1 2 3 0 0 0 0 0 0 10 0 12 13
1: 1 2 0 0 0 6 0 0 0 0 0 0 13
2: 0 2 3 0 0 0 0 0 0 10 0 0 13
3: 0 2 0 0 0 0 7 0 0 10 0 0 0

Overall: 1 2 0 0 0 0 0 0 0 0 11 12 13
1: 1 0 0 0 0 0 7 0 0 0 0 0 0
2: 1 0 0 0 0 0 0 0 0 10 0 12 13
3: 0 0 3 4 0 0 0 8 9 10 0 0 0

Overall: 0 2 3 4 0 0 7 8 9 10 0 12 13
1: 1 2 3 0 0 0 7 0 0 0 0 0 0
2: 0 0 0 0 0 0 0 0 0 10 0 0 13
3: 0 2 0 0 0 0 7 8 0 0 0 0 0

Overall: 0 2 3 4 0 0 7 0 9 10 0 12 13
1: 1 2 3 4 0 0 7 0 0 0 0 0 0
2: 0 2 3 0 0 0 7 0 9 10 0 0 13
3: 0 0 3 4 0 0 7 0 0 0 0 0 0

Overall: 0 2 3 0 5 0 7 8 0 10 11 0 13
1: 1 0 0 4 5 0 7 0 0 0 0 0 0
2: 1 2 3 0 5 0 7 0 0 10 0 0 13
3: 1 0 0 0 5 0 7 0 0 0 0 0 0

Overall: 0 0 3 0 5 6 7 0 9 0 0 0 13
1: 1 2 3 4 0 0 7 0 0 0 0 0 0
2: 1 2 3 0 5 6 7 0 0 10 0 12 13
3: 1 0 0 0 5 0 7 0 0 0 0 0 0

Overall: 0 2 3 4 5 0 7 8 9 10 11 0 13
1: 1 0 0 0 0 0 7 0 0 0 0 0 0
2: 1 2 3 0 5 0 7 8 0 10 0 0 13
3: 0 0 3 0 0 0 7 0 0 0 11 0 0

Overall: 1 2 3 0 0 0 0 0 9 10 0 12 13
1: 1 0 0 4 5 0 7 0 0 0 0 0 0
2: 1 0 0 4 5 0 0 0 9 10 0 0 0
3: 0 0 0 0 0 0 7 0 0 10 0 0 0

Overall: 0 2 0 0 0 6 0 0 9 10 0 0 13
1: 1 0 0 4 5 0 7 0 0 0 0 0 0
2: 1 0 0 0 5 0 0 0 0 10 0 0 0
3: 1 2 0 0 5 0 7 8 0 0 0 0 0

(*k*NN/SFS)
Overall: 1 0 3 4 0 0 7 0 0 10 11 0 0
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 2 0 0 0 0 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 0 0 0 7 0 0 0 0 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 2 3 0 0 0 0 0 0 10 0 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 0 0 6 7 0 0 10 0 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 0 5 0 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 0 5 6 7 0 0 10 0 12 0
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 0 0 6 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 2 3 0 5 0 7 0 0 0 11 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 0 3 0 5 0 7 0 0 0 11 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 3 4 0 0 7 0 0 0 11 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 2 0 0 5 0 7 0 0 10 11 12 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 4 5 6 7 0 0 10 11 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 4 0 0 0 0 0 10 11 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 0 0 0 0 0 0 7 8 0 10 0 0 0
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 0 0 6 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 0 0 0 7 0 0 10 0 0 0
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 0 3 0 0 0 0 0 0 10 0 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 2 3 4 5 6 7 0 9 10 0 12 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 0 0 0 5 0 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 3 0 5 0 7 0 0 10 11 12 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 2 3 0 5 0 7 0 0 10 0 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 0 0 0 0 5 0 7 0 0 10 0 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 4 0 6 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 4 5 0 7 0 0 10 11 12 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 0 0 0 0 0 0 0 0 0 10 0 0 13
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 0 0 0 0 0 7 8 0 10 0 12 0
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 0 0 4 5 0 0 0 9 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

Overall: 1 2 3 4 0 0 7 8 9 10 11 0 13
1: 1 2 3 4 5 6 7 8 9 10 11 12 13
2: 1 0 0 0 5 0 0 0 0 10 0 0 0
3: 1 2 3 4 5 6 7 8 9 10 11 12 13

(*k*NN/GA)
Overall: 1 0 3 4 0 0 0 0 9 10 0 12 13
1: 0 2 3 4 0 0 0 8 9 10 0 0 13
2: 1 2 0 4 0 6 0 8 0 10 0 12 13
3: 0 0 0 0 0 0 0 0 9 10 0 12 0

Overall: 1 2 0 4 0 0 7 0 9 10 11 12 13
1: 1 2 0 0 0 6 0 0 9 10 0 0 0
2: 1 2 0 4 5 0 7 0 0 10 0 12 13
3: 0 0 0 0 0 0 7 0 9 0 0 12 13

Overall: 0 2 0 0 0 0 7 0 0 10 11 12 13
1: 1 0 3 0 0 0 0 0 0 0 11 12 13
2: 1 0 0 0 5 0 7 0 0 10 11 0 13
3: 0 0 3 0 0 6 7 0 0 0 0 0 0

Overall: 1 2 0 0 0 0 7 8 9 10 0 12 13
1: 1 2 0 0 5 6 7 0 0 0 0 0 13
2: 1 2 0 4 0 0 0 8 0 10 11 12 13
3: 0 0 0 0 5 6 7 0 9 0 11 0 0

Overall: 1 0 0 4 0 6 7 0 9 10 0 12 13
1: 0 2 0 4 5 6 0 0 0 10 11 0 0
2: 0 0 0 4 5 6 7 0 0 10 0 0 13
3: 1 0 0 0 0 0 7 0 9 10 0 12 13

Overall: 0 2 3 0 5 0 7 0 0 10 11 12 13
1: 0 0 3 0 0 0 7 8 0 0 0 0 13
2: 1 2 3 4 0 0 0 0 9 10 11 0 13
3: 1 0 3 0 5 0 7 0 9 0 0 12 0

Overall: 1 0 3 4 0 0 7 0 0 10 11 0 13
1: 0 0 0 0 0 6 0 0 0 10 0 0 13
2: 1 0 0 4 5 0 7 0 0 10 0 12 13
3: 0 2 0 4 5 0 7 0 9 0 0 0 13

Overall: 1 0 3 0 0 6 7 0 0 10 0 12 0
1: 1 2 0 0 5 0 7 0 0 10 11 0 0
2: 0 0 0 4 0 0 7 0 0 10 0 0 0
3: 0 0 0 0 0 0 7 0 9 0 0 12 13

Overall: 1 2 0 0 5 6 0 0 9 10 11 12 13
1: 0 0 3 0 0 0 7 8 0 10 0 0 0
2: 1 2 3 0 5 6 0 0 0 10 11 12 13
3: 0 0 0 0 5 0 7 0 9 0 0 0 13

Overall: 0 2 0 4 5 0 7 0 9 10 0 12 13
1: 1 2 0 0 5 0 7 8 0 0 0 0 0
2: 1 0 0 0 0 0 7 8 0 10 0 0 13
3: 0 0 3 0 5 0 0 0 9 0 11 12 0

Overall: 1 0 3 0 0 0 0 0 0 10 11 12 13
1: 1 0 0 0 0 0 0 0 9 10 0 12 13
2: 1 0 3 4 5 0 0 0 0 10 11 12 13
3: 0 0 3 0 0 6 7 0 0 0 0 0 0

Overall: 0 2 0 4 5 0 7 0 9 10 0 12 13
1: 0 0 0 0 5 6 7 0 0 10 0 0 13
2: 0 0 0 0 5 0 0 0 0 10 0 0 13
3: 0 2 0 0 0 0 7 0 0 0 11 12 13

Overall: 0 2 0 4 5 6 7 8 0 10 0 12 13
1: 1 0 3 0 0 6 0 0 0 0 0 0 13
2: 0 2 0 4 5 0 7 0 0 10 0 12 13
3: 0 0 0 0 0 0 7 8 0 10 0 12 0

Overall: 1 0 0 4 5 0 7 0 0 10 11 12 13
1: 1 0 0 4 5 6 0 0 9 10 0 0 0
2: 1 0 0 0 5 0 0 0 9 10 0 0 13
3: 1 0 0 4 0 6 7 0 0 0 0 12 0

Overall: 1 0 3 4 0 6 0 0 9 10 11 12 13
1: 1 0 0 4 5 0 7 0 0 10 11 0 0
2: 1 0 0 4 5 6 7 0 0 0 0 0 13
3: 0 0 0 0 0 6 0 0 9 10 0 12 0

Credit Approval Dataset
(NBC/SBS)
Overall: 0 0 0 4 5 0 7 0 9 10 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 2 0 4 0 0 0 0 9 10 11 0 0 0 0

Overall: 0 0 0 0 0 0 7 0 9 0 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 2 0 0 0 0 0 0 9 10 11 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 1 0 0 4 5 6 7 8 9 10 11 0 13 14 15

Overall: 0 0 0 4 0 6 7 0 9 0 11 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 2 0 0 0 0 0 0 9 10 11 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 15

Overall: 0 0 0 4 0 0 7 0 9 10 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 0

Overall: 0 0 0 4 0 0 7 0 9 10 0 12 13 0 15
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 15

Overall: 0 0 0 4 0 6 0 0 9 0 11 0 13 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 15

Overall: 0 0 0 4 0 0 0 0 9 10 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 0

Overall: 0 0 0 4 5 0 0 0 9 10 0 0 13 0 15
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 2 0 4 0 6 7 8 9 10 11 0 0 0 15

Overall: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 1 0 3 4 5 6 7 8 9 10 11 12 13 14 15

Overall: 1 0 0 4 0 0 0 0 9 10 0 0 13 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 1 2 3 4 0 6 7 8 9 10 11 0 0 14 15

Overall: 0 0 0 4 0 0 7 0 9 10 0 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 2 0 0 0 0 0 0 9 10 11 0 13 0 0

Overall: 1 0 0 4 0 0 0 0 9 0 11 0 0 0 0
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 4 5 6 0 8 9 10 11 0 0 14 15

Overall: 1 0 0 0 0 0 0 0 9 10 0 0 0 0 15
+: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-: 0 0 0 0 0 0 0 0 9 10 11 0 0 0 0

(*k*NN/SBS)
Overall: 0 0 0 0 0 0 0 0 9 10 0 12 0 14 0
+: 0 2 0 4 0 0 7 8 9 0 0 0 0 14 0
-: 0 0 3 4 0 6 0 8 9 10 11 12 0 0 0

Overall: 1 0 0 4 0 6 7 0 9 10 0 0 0 0 0
+: 1 0 0 4 0 6 0 0 9 0 0 0 0 14 0
-: 1 0 3 4 0 6 0 8 9 10 11 0 0 0 0

Overall: 1 0 0 0 0 0 0 0 9 10 0 0 0 0 0
+: 0 0 0 4 0 0 0 8 9 10 0 12 0 14 0
-: 0 0 3 4 5 0 7 0 9 0 11 12 13 0 0

Overall: 1 0 0 4 0 0 0 0 9 10 0 12 0 0 0
+: 1 0 3 0 0 0 7 0 9 0 0 0 0 0 0
-: 1 0 3 0 0 6 7 8 9 0 11 0 0 0 15

Overall: 0 0 0 4 5 0 7 0 9 10 0 12 13 14 0
+: 1 2 0 4 5 0 0 8 9 0 0 0 0 0 0
-: 1 0 3 4 0 0 7 0 9 0 11 0 0 0 0

Overall: 1 0 3 0 0 0 7 8 9 10 0 12 0 14 0
+: 0 0 0 4 0 0 7 0 9 0 0 12 0 14 0
-: 1 2 3 4 0 0 0 8 9 0 11 0 0 0 0

Overall: 1 0 0 4 0 6 7 0 9 0 0 0 0 14 0
+: 1 2 0 4 0 6 0 8 9 0 0 12 0 14 0
-: 0 0 3 4 0 6 7 0 9 0 11 12 0 0 0

Overall: 0 0 0 4 0 6 7 0 9 10 0 12 0 14 0
+: 0 0 3 0 0 6 7 0 9 10 0 12 0 0 0
-: 1 0 0 4 5 6 7 8 9 10 11 12 0 14 0

Overall: 1 0 0 4 0 6 7 0 9 10 0 12 0 0 0
+: 0 2 0 4 0 6 7 0 9 10 0 12 13 14 0
-: 1 2 3 4 0 0 7 8 9 0 11 12 13 0 0

Overall: 1 0 0 4 5 6 7 0 9 10 0 12 13 0 0
+: 0 2 0 4 0 0 7 0 9 0 0 0 0 14 0
-: 0 0 3 4 0 0 0 0 9 10 11 12 0 0 0

Overall: 0 0 0 4 0 0 7 0 9 10 0 0 0 14 0
+: 0 2 0 4 0 6 0 0 9 10 0 12 0 0 0
-: 1 0 3 4 5 6 0 0 9 10 11 12 13 0 0

Overall: 0 0 0 4 0 6 0 0 9 10 0 0 13 14 0
+: 0 0 0 4 0 6 7 0 9 0 0 0 0 14 0
-: 0 0 3 4 0 0 7 8 0 0 11 12 0 0 0

Overall: 0 2 0 4 0 6 7 0 9 10 0 0 13 14 0
+: 0 2 0 0 0 6 0 0 9 0 0 0 0 14 0
-: 0 0 3 0 0 0 0 8 9 0 11 12 0 0 0

Overall: 0 0 0 4 0 6 0 0 9 10 0 0 13 14 0
+: 0 0 0 0 0 0 7 0 9 0 0 0 0 0 0
-: 0 0 3 0 0 0 0 8 9 0 11 12 0 0 0

Overall: 1 0 0 4 0 6 0 0 9 10 0 0 13 0 0
+: 0 0 0 4 0 6 7 0 9 10 0 12 13 14 0
-: 1 0 3 4 0 6 0 8 9 0 11 12 0 0 0

(*k*NN/SFS)
Overall: 1 2 0 0 0 6 0 0 9 10 0 0 13 14 0
+: 0 2 3 4 0 6 0 0 9 10 0 0 0 0 0
-: 0 2 0 0 0 6 0 0 9 10 11 0 13 0 0

Overall: 1 2 0 0 0 6 7 0 9 0 0 0 13 0 0
+: 0 0 0 0 0 0 7 0 9 0 0 0 0 14 0
-: 0 0 3 0 0 0 0 8 0 0 11 12 0 0 0

Overall: 0 2 0 4 0 6 7 0 9 10 0 0 13 14 0
+: 0 0 0 0 0 0 7 8 9 0 0 12 0 14 0
-: 1 2 3 0 0 0 0 0 0 0 11 12 0 0 0

Overall: 1 2 0 0 0 6 0 0 9 10 0 0 0 0 0
+: 0 2 3 4 5 0 7 0 9 0 0 12 13 14 0
-: 0 2 0 0 0 6 7 0 9 10 0 0 13 0 0

Overall: 1 2 0 4 0 6 0 0 9 0 0 12 0 14 0
+: 0 2 3 4 0 0 7 0 9 0 0 0 13 14 0
-: 1 2 3 4 5 6 7 0 9 10 11 12 13 14 0

Overall: 0 0 0 0 0 0 7 0 9 10 0 0 13 0 0
+: 0 0 3 4 0 0 0 0 9 0 0 12 13 14 0
-: 1 2 0 0 0 6 0 0 0 10 11 12 0 0 0

Overall: 1 2 0 4 0 6 7 0 9 0 0 12 13 14 0
+: 0 2 3 4 0 0 7 0 9 0 0 0 13 14 0
-: 0 2 3 4 0 0 0 0 0 0 11 12 0 0 15

Overall: 1 2 0 4 5 6 7 0 9 10 0 12 0 14 0
+: 0 0 3 0 0 0 7 0 9 0 0 0 13 14 0
-: 0 2 3 4 5 6 0 0 9 10 11 12 13 0 0

Overall: 1 2 0 4 0 6 0 0 9 0 0 0 0 14 0
+: 0 0 0 0 0 6 7 0 9 0 0 0 0 14 0
-: 0 2 3 4 0 0 0 8 0 0 11 12 0 0 0

Overall: 0 2 0 4 0 6 0 0 9 0 0 0 0 0 0
+: 0 2 0 4 5 0 7 0 9 10 0 0 13 14 0
-: 0 2 0 4 0 6 7 0 9 10 0 0 0 0 0

Overall: 0 0 3 4 0 0 7 0 9 0 0 0 13 0 0
+: 0 2 3 0 0 0 7 0 9 10 0 0 0 14 0
-: 0 2 0 4 0 6 0 0 9 10 0 0 13 0 0

Overall: 0 0 0 4 0 6 0 0 9 10 0 0 13 14 0
+: 0 0 0 0 0 6 0 0 9 0 0 12 13 14 0
-: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0

Overall: 1 2 0 4 0 6 7 0 9 10 0 12 13 14 0
+: 0 0 3 0 0 6 0 0 9 0 0 0 0 0 0
-: 0 2 0 4 5 6 7 0 9 10 0 0 13 14 0

Overall: 0 2 0 4 0 6 0 0 9 10 0 0 13 14 0
+: 0 0 0 0 0 0 7 0 9 10 0 0 13 0 0
-: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0

Overall: 0 0 3 4 0 0 7 0 9 0 0 0 13 14 0
+: 0 0 0 0 0 0 7 0 9 0 0 0 0 14 0
-: 1 2 3 4 5 0 7 0 9 10 11 12 13 14 0

(*k*NN/GA)
Overall: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
+: 1 2 0 0 5 6 7 8 9 0 0 0 13 14 0
-: 0 2 3 4 5 0 0 8 9 0 11 12 0 0 0

Overall: 0 2 0 4 0 0 0 0 9 10 0 0 0 0 0
+: 0 0 0 0 0 0 7 0 9 0 0 0 0 14 0
-: 1 2 3 4 0 6 0 8 9 0 11 0 13 0 0

Overall: 1 2 0 4 0 6 0 0 9 10 0 12 13 0 0
+: 0 2 3 0 5 0 7 0 9 0 0 0 13 14 0
-: 1 2 3 0 0 0 0 0 9 0 11 12 13 14 0

Overall: 0 0 0 0 0 0 7 0 9 10 0 0 0 14 0
+: 0 0 0 4 5 6 7 0 9 10 0 12 0 14 0
-: 0 0 3 0 5 0 7 8 9 10 11 0 13 0 15

Overall: 1 0 0 0 5 6 0 0 9 0 0 12 13 14 0
+: 0 0 3 4 0 0 7 0 9 0 0 12 0 14 0
-: 0 0 3 0 5 0 7 8 0 0 11 0 0 0 0

Overall: 0 2 0 4 0 6 7 0 9 10 0 12 0 0 0
+: 0 0 3 0 5 0 0 0 9 0 0 12 13 14 0
-: 0 0 3 4 0 0 0 8 9 0 11 12 0 14 0

Overall: 0 2 0 0 0 0 7 0 9 10 0 0 0 0 0
+: 0 2 0 0 5 0 7 8 9 0 0 0 0 14 0
-: 0 0 3 0 5 6 0 0 9 10 11 12 13 0 0

Overall: 1 0 0 0 0 0 7 8 9 10 0 12 0 0 0
+: 0 0 3 0 0 0 7 0 9 0 0 0 13 14 0
-: 1 0 0 4 5 6 7 0 9 10 0 12 13 14 0

Overall: 0 0 0 4 5 6 7 0 9 10 0 0 0 14 0
+: 0 2 0 4 0 6 0 0 9 10 11 12 0 14 0
-: 0 0 3 0 0 0 0 8 9 0 11 12 13 0 0

Overall: 1 0 0 4 0 6 7 0 9 10 0 0 13 14 0
+: 0 2 0 4 0 0 7 0 9 10 0 0 13 14 0
-: 0 0 3 4 5 0 7 0 9 0 11 12 0 14 0

Overall: 0 0 3 0 5 0 0 0 9 0 0 0 0 0 0
+: 1 0 0 0 5 0 7 8 9 0 0 0 0 0 0
-: 0 0 3 4 0 0 0 8 9 0 11 12 13 14 0

Overall: 0 0 0 4 0 6 0 0 9 10 0 0 13 14 0
+: 1 0 0 4 5 6 0 0 9 0 0 12 13 14 0
-: 0 0 3 0 0 0 7 8 9 0 11 12 13 0 0

Overall: 0 0 0 0 0 0 0 0 9 10 0 0 0 14 0
+: 0 2 0 0 0 6 7 0 9 0 0 0 0 14 0
-: 1 0 3 0 5 6 7 8 9 0 11 0 0 0 0

Overall: 0 0 3 0 0 0 0 0 9 0 0 0 13 0 0
+: 1 2 0 4 5 6 0 0 9 0 0 0 0 14 0
-: 0 0 3 0 5 0 0 8 9 10 11 12 0 0 0

Overall: 0 2 0 0 0 6 7 0 9 10 0 12 13 14 0
+: 1 0 0 0 0 6 0 0 9 0 0 0 13 14 0
-: 0 0 3 0 5 6 7 8 9 0 11 12 0 0 0

Congressional Voting Dataset
(NBC/SBS)
Overall: 0 0 0 4 0 0 0 0 0 0 11 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 0 12 0 0 0 0

Overall: 0 2 0 4 0 0 0 0 9 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 11 0 0 0 0 0

Overall: 1 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
republican: 0 0 0 4 0 6 0 0 0 0 0 0 0 0 0 0
democrat: 1 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 9 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 1 0 0 4 5 0 0 0 0 10 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 1 0 0 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 15 0

Overall: 0 0 3 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 0 8 0 0 0 12 0 0 0 0

Overall: 1 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 1 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 7 0 0 0 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 6 0 0 0 0 0 12 0 0 0 0

Overall: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 1 0 0 4 0 0 0 0 0 0 11 12 0 0 0 0

(*k*NN/SBS)
Overall: 0 0 0 4 0 0 0 0 0 0 11 12 13 0 0 0
republican: 0 0 0 4 0 0 7 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 2 3 4 0 0 7 0 0 0 0 0 0 0 0 0
republican: 1 0 3 4 0 6 0 0 9 0 11 12 13 14 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 8 0 0 0 0 0 14 0 0
republican: 1 0 0 4 0 0 0 8 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 6 0 8 9 10 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 9 0 11 0 0 0 0 0
republican: 0 0 3 4 0 0 0 0 0 0 0 0 0 14 0 0
democrat: 1 0 0 4 5 0 0 8 0 0 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 7 0 0 0 0 0 0 0 0 0

Overall: 1 2 3 4 0 0 0 8 9 0 11 12 0 14 0 0
republican: 1 0 0 4 0 0 0 8 9 0 0 0 13 14 0 0
democrat: 1 2 0 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 3 4 0 0 0 8 0 0 11 0 13 14 15 0
republican: 0 2 3 4 5 0 7 0 0 10 11 12 0 14 0 0
democrat: 0 0 3 4 0 0 7 0 0 10 11 0 0 14 0 0

Overall: 0 2 3 4 0 6 0 0 9 10 11 0 0 0 15 0
republican: 1 0 0 4 0 0 0 8 9 0 0 0 0 14 0 0
democrat: 0 0 3 4 5 6 7 0 0 10 0 12 0 0 15 0

Overall: 0 0 3 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 7 0 9 0 11 0 0 0 0 0

Overall: 0 2 3 4 0 6 7 0 9 0 11 12 13 14 0 0
republican: 0 2 3 4 0 0 7 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 7 0 0 0 11 0 0 0 0 0

Overall: 0 0 3 4 0 0 7 0 0 0 0 12 0 14 0 0
republican: 0 2 0 4 0 0 7 0 0 0 0 0 13 0 0 0
democrat: 0 2 3 4 0 0 7 0 0 0 0 0 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
republican: 0 0 0 4 0 0 0 8 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 11 0 0 0 0 0

Overall: 1 0 3 4 5 6 0 0 9 10 11 0 13 14 0 16
republican: 1 0 0 4 0 6 0 0 9 10 11 12 13 14 0 0
democrat: 0 2 3 4 5 0 0 8 0 0 11 0 0 0 0 0

Overall: 0 0 3 4 0 0 0 0 0 10 11 0 13 0 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 5 0 0 0 0 0 11 12 13 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 14 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 1 2 0 4 0 0 0 0 0 0 11 12 0 0 0 0

(*k*NN/SFS)
Overall: 0 0 0 4 0 0 7 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 7 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 0 3 4 0 0 7 0 0 10 11 0 0 0 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 9 0 0 0 0 0 0 16
republican: 0 0 0 4 0 0 0 0 0 10 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 7 0 0 0 11 12 0 0 15 0

Overall: 0 2 0 4 0 0 0 0 0 0 11 12 0 0 0 0
republican: 1 2 3 4 0 0 0 0 0 0 0 0 0 14 0 0
democrat: 0 0 0 4 5 0 0 0 0 0 11 0 13 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 2 3 4 0 0 0 0 0 0 0 0 0 0 15 16
democrat: 0 0 3 4 0 0 0 0 9 0 0 0 13 0 0 0

Overall: 1 0 0 4 5 6 0 0 0 10 0 12 0 14 0 0
republican: 1 0 0 4 5 0 0 8 0 0 0 0 0 14 0 16
democrat: 0 0 0 4 0 0 0 0 0 10 0 0 0 0 0 0

Overall: 1 2 3 4 0 6 0 0 0 0 11 0 0 0 0 0
republican: 1 0 0 4 0 6 7 0 9 0 11 0 0 14 0 0
democrat: 0 0 3 4 5 6 7 8 0 10 11 0 0 14 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 0
democrat: 0 2 3 4 5 6 7 0 9 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 16
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 16
democrat: 0 2 3 4 0 0 0 0 9 0 11 0 13 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 0 0 16
republican: 0 0 0 4 0 0 0 0 0 10 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 7 0 0 10 11 0 0 0 15 0

Overall: 0 0 3 4 0 6 7 8 0 10 0 0 0 14 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 2 3 4 0 0 0 0 0 10 0 0 0 0 0 0

Overall: 1 2 3 4 0 6 0 0 9 10 11 12 13 14 0 16
republican: 0 2 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 3 4 0 0 0 0 0 0 0 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 0 0 0 11 0 0 0 0 0
republican: 1 2 3 4 5 6 7 0 0 0 11 12 0 14 0 0
democrat: 1 0 0 4 5 0 0 0 9 0 11 12 0 0 0 0

Overall: 1 0 0 4 0 0 0 0 0 0 0 0 0 0 0 16
republican: 0 2 3 4 0 6 7 0 9 0 0 0 0 14 0 0
democrat: 0 0 3 4 0 6 7 0 0 0 0 12 0 0 0 0

(*k*NN/GA)
Overall: 0 2 0 4 0 0 0 0 9 0 0 0 0 14 0 0
republican: 0 0 0 4 5 0 0 8 0 10 0 0 13 14 0 0
democrat: 1 0 0 4 0 6 7 0 0 10 11 12 0 0 0 0

Overall: 0 2 3 4 5 0 7 0 0 0 0 0 13 14 0 0
republican: 1 2 3 4 0 6 0 0 9 10 11 12 13 14 0 0
democrat: 0 2 3 4 5 0 0 0 0 0 11 0 13 14 15 0

Overall: 0 0 0 4 0 0 7 0 0 0 11 12 13 0 0 0
republican: 0 0 0 4 0 0 0 0 0 9 10 0 12 0 0 0 16
democrat: 0 0 3 4 0 6 0 0 9 10 0 12 0 0 0 0

Overall: 0 0 0 4 5 6 0 0 0 0 0 0 0 14 0 0
republican: 0 2 0 4 0 0 7 8 0 0 11 12 0 14 0 0
democrat: 0 0 0 4 0 0 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 0 0 4 0 6 0 0 9 0 0 12 0 0 0 0
republican: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
democrat: 0 0 0 4 0 0 0 0 0 0 0 12 13 14 0 0

Overall: 0 2 3 4 0 0 0 0 0 0 0 0 0 0 15 16
republican: 1 0 3 4 0 0 0 0 0 10 0 0 0 14 15 0
democrat: 1 0 3 4 5 0 0 0 0 10 11 12 0 0 0 0

Overall: 0 0 0 4 0 6 0 0 0 0 0 12 13 0 0 16
republican: 0 2 3 4 0 0 0 0 9 0 0 0 0 0 15 16
democrat: 0 0 3 4 5 0 7 8 9 10 11 0 13 14 15 0

Overall: 0 0 3 4 5 0 7 8 0 10 11 0 0 14 0 0
republican: 0 0 0 4 0 0 7 0 0 0 0 12 0 0 0 16
democrat: 0 0 3 4 5 6 0 8 0 10 0 0 0 14 15 0

Overall: 1 0 3 4 5 6 7 0 0 0 11 12 0 0 15 16
republican: 1 2 0 4 5 0 7 0 0 0 0 12 13 14 0 0
democrat: 0 0 3 4 0 0 7 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 8 9 10 0 0 0 0 0 0
republican: 1 2 3 4 0 6 7 8 9 10 11 12 0 0 0 0
democrat: 0 0 3 4 5 6 7 0 0 0 11 12 0 14 15 0

Overall: 0 0 0 4 0 0 0 0 0 0 0 12 0 14 15 16
republican: 0 2 0 4 0 6 7 0 0 0 0 0 13 14 0 0
democrat: 0 2 3 4 0 0 7 0 0 0 0 0 0 0 0 0

Overall: 0 2 3 4 0 0 0 0 0 10 0 0 0 0 0 0
republican: 0 2 3 4 0 0 7 0 0 10 0 0 13 14 15 16
democrat: 0 0 0 4 0 0 0 8 0 0 0 12 0 0 0 0

Overall: 0 0 3 4 0 6 0 0 9 10 0 12 0 0 0 0
republican: 0 0 0 4 0 0 7 0 0 0 0 0 13 14 0 0
democrat: 1 0 3 4 5 0 7 0 0 10 11 0 0 0 15 16

Overall: 0 0 0 4 0 0 0 0 9 0 11 0 13 0 0 0
republican: 0 0 0 4 0 6 0 0 0 10 11 0 0 0 0 0
democrat: 0 0 0 4 5 6 0 0 0 0 11 12 0 0 0 0

Overall: 0 0 0 4 0 0 0 8 9 0 11 12 13 14 0 0
republican: 0 0 0 4 0 6 7 0 9 0 0 12 13 14 15 16
democrat: 0 0 3 4 0 0 0 0 0 0 11 12 0 0 0 0

Montgomery County Jails Dataset
(*k*NN/SBS)
Overall: 1 0 0 0 5 6 0 0 0 0 11 12 0 0 0 0
-1: 0 2 3 0 5 0 0 0 0 0 11 0 0 0 15 16
1: 0 0 0 4 5 6 0 8 0 0 0 12 0 14 0 0

Overall: 0 0 0 4 5 6 0 0 0 0 11 0 13 0 0 0
-1: 1 0 0 4 5 6 7 0 0 0 11 0 13 0 15 0
1: 0 0 0 4 5 6 7 8 9 10 0 0 13 14 15 16

Overall: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
-1: 0 2 3 0 5 0 0 0 0 0 11 0 13 0 15 16
1: 0 0 0 4 5 6 0 8 0 0 11 12 0 0 0 0

Overall: 0 2 0 4 5 6 7 0 0 10 11 12 13 14 0 16
-1: 0 2 0 0 5 6 0 0 9 0 11 0 0 0 15 16
1: 0 0 0 4 0 0 0 0 0 0 0 0 0 0 14 0 0

Overall: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0
-1: 1 2 0 4 0 0 0 0 0 0 11 0 0 0 0 0
1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0

Overall: 1 2 0 4 5 6 7 0 0 0 0 0 0 0 15 0
-1: 1 2 3 0 0 6 7 8 0 10 11 12 13 14 15 16
1: 1 0 0 4 5 6 7 8 9 0 11 0 13 14 15 0

Overall: 1 2 0 4 5 6 7 8 0 10 11 12 13 14 0 16
-1: 0 0 3 0 0 6 0 0 0 10 11 0 13 0 15 16
1: 1 0 0 4 0 0 0 8 0 0 0 12 13 14 0 0

Overall: 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0
-1: 0 2 3 0 5 6 0 0 0 0 11 0 13 0 0 0
1: 1 2 0 4 5 0 7 8 9 10 11 0 13 14 15 0

Overall: 0 2 3 4 5 6 7 0 9 10 11 0 13 14 15 16
-1: 0 2 3 0 0 0 0 0 9 10 11 0 13 0 15 16
1: 1 0 0 4 0 6 0 8 0 0 11 12 13 14 15 0

Overall: 1 0 0 4 5 0 7 0 0 0 0 0 0 14 15 16
-1: 0 2 3 0 5 6 0 0 0 0 0 0 0 0 0 16
1: 1 0 0 4 5 6 7 8 0 0 11 0 13 14 15 16

Overall: 1 0 0 0 0 0 0 0 0 10 0 0 13 14 0 0
-1: 1 2 0 0 5 0 7 0 0 0 11 0 0 0 0 16
1: 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
-1: 0 0 3 0 5 0 0 0 0 0 0 0 0 0 15 16
1: 1 0 0 4 0 0 0 8 0 0 11 0 13 0 0 0

Overall: 1 2 3 4 5 0 7 0 0 10 11 12 13 0 0 0
-1: 0 2 3 0 5 6 0 0 0 0 11 0 0 0 15 16
1: 1 0 3 4 5 0 7 8 0 10 11 12 13 14 0 16

Overall: 0 2 0 0 0 0 0 8 0 0 0 0 0 0 15 0
-1: 0 2 3 0 5 6 7 0 0 10 11 12 13 14 15 0
1: 0 0 0 0 0 6 7 8 0 10 0 0 0 14 15 0

Overall: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1: 1 2 0 0 5 6 7 0 0 10 0 12 13 0 15 16
1: 0 2 0 0 0 6 7 8 0 10 0 0 0 14 0 0

(*k*NN/SFS)
Overall: 0 0 0 0 0 5 0 0 0 0 0 11 12 0 14 0 0
-1: 0 2 0 0 5 6 7 0 9 0 0 0 0 14 15 16
1: 0 0 0 0 0 0 0 0 9 0 0 12 0 0 0 0

Overall: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1: 0 0 0 0 0 0 0 0 0 0 11 0 0 0 15 0
1: 0 0 0 0 5 0 0 8 0 0 0 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
-1: 0 2 3 0 5 6 7 0 9 10 11 12 0 0 15 16
1: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Overall: 1 2 0 0 0 6 0 0 9 0 0 12 0 14 0 0
-1: 0 2 3 0 5 6 0 0 0 10 11 0 0 0 15 16
1: 1 0 0 4 5 0 0 8 9 10 0 12 0 14 0 0

Overall: 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0
-1: 1 2 0 0 5 6 0 0 9 0 11 12 13 14 15 16
1: 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0

Overall: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1: 1 2 0 0 0 0 0 0 9 0 11 12 0 14 15 16
1: 1 0 0 0 5 6 0 8 9 10 0 12 0 14 0 16

Overall: 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0
-1: 0 0 3 0 0 0 0 8 9 10 0 12 13 14 15 16
1: 0 0 0 0 5 0 0 0 0 0 0 0 0 14 0 0

Overall: 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0
-1: 1 2 3 0 5 6 7 8 9 10 11 12 13 14 15 16
1: 1 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 0 0 0 12 13 14 0 16
-1: 0 2 3 0 0 0 0 0 9 10 11 0 13 0 15 16
1: 1 2 0 4 0 6 0 8 9 10 11 12 13 14 15 16

Overall: 0 0 0 0 0 0 0 0 0 10 11 0 0 0 0 16
-1: 0 0 0 0 0 0 7 0 0 10 11 0 0 14 15 16
1: 0 0 0 0 5 0 7 0 0 0 0 0 0 0 0 16

Overall: 0 0 0 0 0 0 0 8 9 0 0 0 0 0 0 0
-1: 0 2 0 0 5 6 0 8 9 10 11 12 0 0 0 16
1: 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 9 0 0 0 0 0 0 0
-1: 0 2 3 0 5 0 0 0 9 10 11 0 13 0 15 16
1: 0 0 0 0 5 0 0 0 9 0 0 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 8 9 0 0 0 0 14 0 0
-1: 0 2 3 0 5 0 7 0 9 0 11 0 0 0 15 16
1: 0 0 0 0 0 0 0 8 9 10 0 0 0 14 0 0

Overall: 0 2 0 0 0 0 0 8 0 0 0 0 0 0 15 0
-1: 1 2 3 0 5 6 7 8 9 10 11 12 13 14 15 16
1: 0 0 0 0 0 0 0 8 9 0 0 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0
-1: 0 0 0 0 0 0 0 8 0 10 0 0 0 0 0 0
1: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0

(*k*NN/GA)
Overall: 0 0 0 0 0 0 0 0 9 0 0 12 0 14 15 16
-1: 0 2 0 0 0 0 7 0 0 0 0 0 0 14 15 16
1: 0 0 0 0 0 0 0 0 9 0 0 12 0 0 0 0

Overall: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1: 0 2 0 0 5 0 7 0 9 0 0 0 13 0 15 16
1: 0 0 0 0 0 0 0 8 0 0 0 0 0 0 14 15 0

Overall: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
-1: 0 2 3 0 5 0 0 0 0 0 11 0 13 0 15 16
1: 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Overall: 1 2 0 0 0 0 0 0 0 0 0 0 0 14 0 0
-1: 0 2 0 0 0 6 0 0 0 0 0 0 0 0 15 16
1: 0 2 0 4 0 0 0 8 0 10 0 12 13 14 0 0

Overall: 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0
-1: 0 0 0 0 0 0 6 0 0 9 0 11 12 0 0 15 0
1: 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-1: 0 2 0 0 0 6 0 0 9 0 11 12 0 14 15 16
1: 0 0 0 0 0 0 0 8 9 0 0 12 0 0 15 0

Overall: 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
-1: 0 2 0 0 5 0 7 0 0 0 11 12 0 0 15 16
1: 0 0 0 0 0 0 0 7 0 9 0 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0
-1: 1 2 3 0 0 0 0 8 9 10 0 12 13 0 15 16
1: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 15 0

Overall: 1 0 0 0 0 0 0 8 0 0 0 0 13 14 0 16
-1: 0 2 3 0 5 0 0 0 9 10 11 0 13 14 15 16
1: 0 0 0 4 0 6 0 8 9 10 11 0 13 14 15 0

Overall: 0 0 0 0 5 0 0 0 9 0 0 0 0 14 15 0
-1: 1 2 3 0 5 0 0 8 9 10 11 0 0 0 0 16
1: 0 0 0 0 0 0 0 0 0 0 10 11 0 0 0 0 16

Overall: 0 0 0 0 0 0 0 8 9 0 0 0 0 0 0 0
-1: 0 0 0 0 0 0 7 0 9 0 0 12 13 0 0 16
1: 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0

Overall: 0 0 0 0 5 0 0 0 9 0 0 0 0 0 0 0
-1: 0 2 3 0 5 0 0 0 9 10 11 0 13 0 15 16
1: 0 0 3 4 5 0 0 8 9 10 0 0 0 14 0 0

Overall: 0 0 0 4 5 6 0 0 0 10 0 12 0 14 0 16
-1: 0 0 3 0 5 0 0 0 9 0 0 0 13 0 15 0
1: 0 0 0 0 0 0 0 8 9 0 11 0 0 0 0 0

Overall: 0 0 0 0 0 0 0 8 0 0 0 0 0 0 15 0
-1: 0 2 3 0 5 6 7 0 9 10 11 12 13 14 15 0
1: 0 2 0 0 0 0 0 8 0 0 0 0 0 0 0 0

Overall: 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1: 0 0 0 0 0 0 0 8 0 10 0 0 0 0 0 0
1: 0 0 0 0 0 0 7 8 9 10 0 0 0 14 0 0