# Efficient Generation and Selection of Combined Features for Improved Classification

Thesis By
**Ahmad Shono**

In Partial Fulfillment of the Requirements
For the Degree of
**Master of Science in Computer Science**

King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia

May 2014

# EXAMINATION COMMITTEE APPROVAL

The thesis of Ahmad Shono is approved by the examination committee.

Committee Chairperson: Vladimir Bajic
Committee Member: Mikhail Moshkov
Committee Member: Xin Gao

# ABSTRACT


Efficient Generation and Selection of Combined Features
for Improved Classification

Ahmad Shono

This study contributes a methodology and associated toolkit developed to allow users to experiment with the use of combined features in classification problems. Methods are provided for efficiently generating combined features from an original feature set, for efficiently selecting the most discriminating of these generated combined features, and for efficiently performing a preliminary comparison of the classification results when using the original features exclusively against the results when using the selected combined features. The potential benefit of considering combined features in classification problems is demonstrated by applying the developed methodology and toolkit to three sample data sets where the discovery of combined features containing new discriminating information led to improved classification results.

# ACKNOWLEDGEMENTS

I would like to acknowledge the Saudi Aramco company for sponsoring my studies at KAUST, and to thank my Saudi Aramco colleagues who made it possible. I am greatly and particularly thankful to my supervisor, Professor Vladimir Bajic, as this thesis would not have been possible from conception to conclusion without his continuous support and encouragement.

I would also like to thank my thesis committee members Professor Mikhail Moshkov, and Professor Xin Gao. I consider myself fortunate for having benefited from their core courses in my first semester at KAUST. I would also like to thank Professor Xiangliang Zhang for teaching me most of what I know about data mining methods and algorithms during my attendance of one of her excellent courses.

Thanks are also due to my fellow graduate students Ghofran Othoum and Haitham Ashoor for providing the data samples and to Othman Soufan for his plentiful and valuable advice.

I am also thankful for the great fortune of having unconditionally loving and supportive parents without whom I wouldn't be the thesis-writing man I am today. Finally, I offer my deepest thanks to my wife, Dimah Abu-Daff, for her unwavering and steadfast belief in me despite all my doubts, for her unquestionably greater contribution to raising our two wonderful daughters, Jenna and Judy, despite her demanding full time work and my irregular hours, and for helping them with their homework while I was busy doing mine.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ACC | Accuracy |
| ADP | Absolute Difference Percentage |
| API | Application Programming Interface |
| ARFF | Attribute-Relation File Format |
| BP | Base Pairs |
| CLI | Command Line Interface |
| CSV | Comma Separated Values |
| DCFD | Dragon Combined Features Discovery Toolkit |
| DIP | Distance from the Ideal Predictor |
| F-M | F-Measure |
| FN | False Negative |
| FP | False Positive |
| GUI | Graphical User Interface |
| JAR | Java Archive |
| KAUST | King Abdullah University of Science and Technology |
| KNN | K-Nearest Neighbors |
| KWIC | KAUST Weka Initial Classification Tool |
| LEP | Least Error Percentage |
| NF-kB | Nuclear Factor kappa-light-chain-enhancer of activated B cells |
| NNGE | Non-Nested Generalized Exemplars classifier |
| PRC | Precision |
| SENS | Sensitivity |
| SPEC | Specificity |
| SVM | Support Vector Machines |
| TN | True Negative |
| TNR | True Negative Rate |
| TP | True Positive |
| TPR | True Positive Rate |
| WEKA | Waikato Environment for Knowledge Analysis |

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1: INTRODUCTION

## 1.1 A Brief Review of Binary Classification

In bioinformatics and in other fields, a common problem that arises is finding a way to distinguish between two data sets so that given any data item we can reliably determine to which data set it belongs. This is known as the binary classification problem [1] where the data items are called instances and the data sets are called classes. For example, the problem of determining whether a patient has a certain disease or not can be considered a binary classification problem where the patient is an instance and the classes are 'diseased' and 'not diseased'.

Machine learning algorithms known as classifiers [1] address the binary classification problem by learning a classification model from a training set of instances. This trained model can then be used to map new instances to the classes. Classifiers train a model by analyzing a chosen number of measurable or observable characteristics of the instances. Depending on context, these characteristics may be called variables, attributes, or features [1]. For the remainder of this document, we will refer to these characteristics as features.

The performance of each classifier varies depending on the nature of the available training set of instances and the chosen features. Although there is no general methodology that always works well, significant improvement in classifier performance may be attained by optimizing the choice of features. This is known as feature selection [1] and can be done through the removal of redundant or irrelevant features, the selection of the best discriminating subset of features, or the discovery and use of new features that contain additional discriminating information.

## 1.2 Generation of Combined Features

As mentioned at the end of the previous section, one way to possibly attain significant improvement in classifier performance is through the discovery and use of new features that contain additional discriminating information. One method of feature discovery is the generation of new features derived from the original features, as is done for example by the Support Vector Machines (SVM) classifier [2]. This study also generates new features but through a different methodology that is based on the construction of combinations of the original features up to a certain maximum combination size. The intuition is that these combinations of features – hereafter referred to as combined features – may contain additional discriminating information, and classifiers that are capable of taking advantage of any such additional information may show significantly improved results.

As an illustrative example, consider the small data set shown in Table 1-1. By considering only the original features A and B, we cannot intuitively discern a good solution, especially since the features appear to have similar value ranges and distributions in both the positive and negative classes. However, if we generate the combined 2-feature AB with values A×B as shown in Table 1-2, then a perfect solution becomes apparent based on the following rule:

$$if\ AB > 2, then\ class\ is\ positive$$

**Table 1-1**: Data Set (before feature generation)

| Positive | | | Negative | | |
|---|---|---|---|---|---|
| **Instance** | **Feature A** | **Feature B** | **Instance** | **Feature A** | **Feature B** |
| 1 | 3 | 1 | 1 | 1 | 2 |
| 2 | 2 | 2 | 2 | 3 | 0 |
| 3 | 1 | 3 | 3 | 2 | 1 |
| 4 | 3 | 2 | 4 | 0 | 3 |

**Table 1-2**: Data Set (after generating new features by multiplication)

| Positive | | Negative | |
|---|---|---|---|
| **Instance** | **Feature AB (A×B)** | **Instance** | **Feature AB (A×B)** |
| 1 | 3 | 1 | 2 |
| 2 | 4 | 2 | 0 |
| 3 | 3 | 3 | 2 |
| 4 | 6 | 4 | 0 |

Note that in this example, calculating the value of the generated combined feature through addition instead of multiplication, as shown in Table 1-3, would also provide a perfect solution based on the following rule:

$$if\ AB > 3, then\ class\ is\ positive$$

**Table 1-3**: Data Set (after generating new features by addition)

| Positive | | Negative | |
|---|---|---|---|
| **Instance** | **Feature AB (A+B)** | **Instance** | **Feature AB (A+B)** |
| 1 | 4 | 1 | 3 |
| 2 | 4 | 2 | 3 |
| 3 | 4 | 3 | 3 |
| 4 | 5 | 4 | 3 |

Ultimately, the choice of a method used to calculate the values of generated combined features should be guided by the nature of the data and the original features in order to preserve interpretability and hopefully gain better insight into the specific classification problem being addressed. For example, if feature A and B from Table 1-1 represent the height and width of an instance, then feature AB (A×B) from Table 1-2 would represent the area of that instance, and the classification results would provide the insight that the positive instances have larger areas compared to the negative instances.

To clarify these ideas further in a more common framework, consider data sets where the feature values represent the number of occurrences or frequency of the features. In that case, the value of a combination of two features through multiplication can be interpreted

as the number of paired occurrences of the two features. Figure 1-1 shows an example of an instance where the value of the combined feature AB (A×B) represents six paired occurrences of feature A (occurring 3 times) and feature B (occurring 2 times).

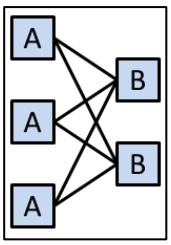


**Figure 1-1**: Combined Features Interpretability Example

Another notable aspect of generation of combination of feature is that the additional discriminating information – if it exists – may be found in combinations of more than two features. It is therefore desirable to generate feature combinations of size two, three, four, and so on. For explanation, here, a feature combination of size 3 means a new feature generated from a combination of 3 original features. We highlight here that the way of combining original features into new one can take various forms, e.g. addition of values, multiplication of values, or in general any nonlinear combination of values.

However, the computational cost of generating combined features can grow very rapidly due to combinatorial explosion, depending on the number of original features, the way how they are combined, and the desired maximum combination size. For example, for a data set with 100 original features and a desired maximum combination size of four, there are over four million combined features to generate:

$$Number\ of\ combined\ features\ to\ generate = \binom{100}{2} + \binom{100}{3} + \binom{100}{4}$$

$$= \frac{100!}{2!\,(100-2)!} + \frac{100!}{3!\,(100-3)!} + \frac{100!}{4!\,(100-4)!}$$

$$= 4950 + 161{,}700 + 3{,}921{,}225 = 4{,}087{,}875$$

## 1.3 Dragon Combined Features Discovery (DCFD) Toolkit

This study contributes a methodology and associated toolkit (DCFD) developed to allow users to experiment with the use of combined features in classification problems. The DCFD toolkit provides efficient tools for generating combined features from an original feature set, for selecting the most discriminating of these generated combined features, and for performing a preliminary comparison of the classification results when using the original features exclusively against the results when using the selected combined features. This study also reports the results of the practical application of the DCFD toolkit on three sample data sets.

# CHAPTER 2: METHODOLOGY

## 2.1 Problem Formulation

In order to investigate the effect of using combined features in binary classification problems, the following tasks are essential:

1) **Efficient Generation of Combination of Features**: The efficient generation of as many combined features as possible is essential since generating more combined features increases the chance of finding additional discriminating information.
2) **Efficient Feature Selection**: The efficient selection of the best discriminating subset of original and generated combined features is essential since the number of generated combined features tends to be significantly large due to combinatorial explosion.
3) **Fair Comparison of Classification Results**: In order to properly investigate the effect of using combined features, it is essential to perform fair comparisons of classification results obtained with the original features versus those obtained with the selected combined features across a wide variety of different types of classifiers.

## 2.2 Generation of Combined Features

Given any binary class data set with numeric features, a maximum combination size, and a method for calculating combined feature values, the objective is to generate a new data set with all possible combined features. In order to generate combined features from data sets that include non-numeric features, the non-numeric features must first be converted into numeric features.

In order to mitigate the effects of combinatorial explosion, two optional and mutually non-exclusive methods can be used for reducing the number of original features and the number of generated combined features. They are described in the following two sections.

### 2.2.1 Occurrence Percentage Cutoff

A feature is said to occur in an instance if its value is greater than zero. Any feature that has an occurrence percentage less than a certain value in both the positive and negative classes is ignored. The occurrence percentage in a class (positive/negative) is calculated as follows:

$$Occurrence\ Percentage\ in\ Class = \frac{\#\ of\ Occurrences\ in\ Class}{\#\ of\ Class\ Instances} \times 100$$

The intuition behind this cutoff is that any feature with a low occurrence percentage in both classes is less likely to be part of any significantly discriminating combined feature. Selecting the occurrence percentage cutoff is a tradeoff between efficient computation and the size of the combined feature search space. Typically, a small occurrence cutoff of 1% – 5% is used since a larger cutoff would acutely increase the chance of overlooking significantly discriminating combined features.

### 2.2.2 Absolute Difference Percentage (ADP) Cutoff

A feature is said to occur in an instance if its value is greater than zero. Any feature that occurs in both the positive and negative classes with an absolute difference percentage less than a certain value is ignored. The absolute difference percentage may be calculated in many different ways, but here it is calculated as follows:

$$Absolute\ Differnece\ Percentage = \frac{|\#\ Positive\ Occurrences - \#\ Negative\ Occurrences|}{\max\{\#\ Positive\ Occurrences, \#\ Negative\ Occurrences\}} \times 100$$

This parameter generally works fine for approximately balanced data sets, but for highly imbalanced data sets should be replaced by:

$$Absolute\ Differnece\ Percentage = |\frac{\#\ Positive\ Occurrences}{\#\ Positive\ Instances} - \frac{\#\ Negative\ Occurrences}{\#\ Negative\ Instances}| \times 100$$

The intuition behind this cutoff is that any feature with a similar number of occurrences in the positive and negative classes is less likely to contain discriminating information, and therefore less likely to be part of any significantly discriminating combined feature. However, this intuition can be misleading for some data sets. As an illustrative example, consider the data set depicted in Table 2-1 where feature A and B occur 50 times in the same positive instances and occur 50 times each in different negative instances. Since features A and B have zero difference percentages, they will be ignored according to the absolute difference percentage cutoff rule, and the significantly discriminating combined feature AB is never discovered.

**Table 2-1**: Difference Percentage Cutoff Example

| | **Positive** | **Negative** |
|---|---|---|
| **Feature A Occurrences** | 50 | 50 |
| **Feature B Occurrences** | 50 | 50 |
| **Feature AB Occurrences** | 50 | 0 |

Therefore, although the absolute difference percentage cutoff is similar to the occurrence percentage cutoff in that it mitigates combinatorial explosion, it also introduces an additional and distinct possibility for overlooking significantly discriminating combined

features, and should consequently only be used when the occurrence percentage cutoff does not provide enough combinatorial explosion mitigation on its own.

## 2.3 Feature Selection

Due to the typically large number of generated combined features, it is necessary to apply feature selection methods that are capable of efficiently processing a large number of features. Since wrapper and advanced filter methods are prohibitively expensive from a computational perspective for such a task, the use of less sophisticated feature ranking methods is warranted. The following sections describe the feature ranking methods used in this study.

### 2.3.1 Rank by Least Error Percentage (LEP)

The error percentage per feature is calculated as follows:

$$Error\ Percentage = \frac{FP + FN}{\#\ of\ Instances} \times 100$$

The confusion matrix [1] – which includes the number of False Positives (FP) and False Negatives (FN) – is determined based on a certain threshold, which is either given as a parameter or found through exhaustive search for the best associated error percentage. For each feature, the exhaustive search considers all possible thresholds between the largest minimum value and the smallest maximum value.

### 2.3.2 Rank by Absolute Difference Percentage (ADP)

Features could be ranked based on the decreasing value of this parameter, described previously in section 2.2.2.

### 2.3.3 Rank by Distance from Ideal Predictor (DIP)

Following ideas from (Bajic 2000, Briefings in Bioinformatics) [3] we can use the concept of distance to represent the quality of our predictor when we use the selected feature. This approach, based on two or more measurements of prediction quality and any type of distance measure, is particularly useful when dealing with highly imbalanced classes of data and can be applied to any set of features and any predictor. This study uses the True Positive Rate (TPR) and True Negative Rate (TNR) prediction quality measurements and Euclidean distance. Since an ideal predictor would classify the entire data set with 100% TPR (sensitivity) and 100% TNR (specificity), we can represent this ideal predictor as the point (1,1) in the plane (TPR,TNR) as depicted in Figure 2-1. Features can then be ranked by their Euclidean "distance from the ideal predictor" (DIP) according to the following calculation:

$$Distance\ from\ Ideal\ Predictor = \sqrt{(1 - TPR)^2 + (1 - TNR)^2}$$

The confusion matrix is determined based on a certain threshold, which is either given as a parameter or found through exhaustive search for the best associated DIP. For each feature, the exhaustive search considers all possible thresholds between the largest minimum value and the smallest maximum value.

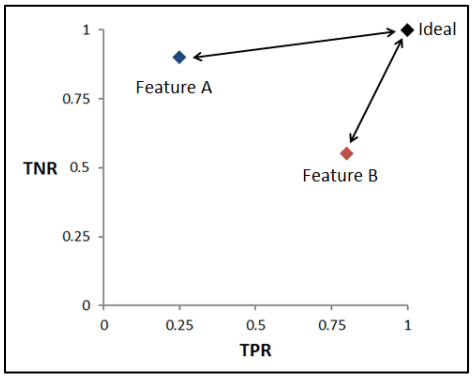


**Figure 2-1**: Distance from the Ideal Predictor (DIP)

### 2.3.4 Rank by P-Value

The p-value [4] per feature is calculated according to the Numerical Recipes [5] computation for two-tailed T distribution test statistic as implemented in [6]. Assuming unequal variance, the t-score and degrees of freedom per feature are calculated as follows:

$$t - score = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

$$degrees\ of\ freedom = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}}$$

Where:

$$n_1 = \#\ positive\ instances \qquad n_2 = \#\ negative\ instances$$

$$\overline{X_1} = average\ positive\ value \qquad \overline{X_2} = average\ negative\ value$$

$$s_1^2 = positive\ variance \qquad s_2^2 = negative\ variance$$

## 2.4 Classification Results Comparison

In order to ensure fair comparison between classification results obtained with the original features versus those obtained with the selected combined features, the following guidelines should be followed:

**1)** Any occurrence or difference percentage cutoffs applied in the generation of the combined features should also be applied on the original features. This ensures that any improvement in the classification results can be attributed to new discriminating information gained from the combined features and not to the removal of irrelevant or redundant original features.

**2)** Feature selection methods applied on the generated combined features should also be applied on the original features. This ensures that any improvement in the classification results can be attributed to new discriminating information gained from the combined features and not to any optimization gained through feature selection.

**3)** Since classifier performance varies greatly depending on the nature of the data set, a wide variety of different types of classifiers should be used for each experiment, thereby ensuring a fair overall comparison.

The following sections provide brief descriptions of the 15 classifiers used in the experiments described in this study. All classifiers were applied using the implementations available in the Weka [7] toolkit.

### 2.4.1 Naïve Bayes classifier

The Naïve Bayes classifier [8] is a probabilistic classifier based on Bayes' theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The classification problem is transformed into the problem of finding the probability of the instance B belonging to the class A. The classifier is considered naïve because it assumes that the features are statistically independent.

### 2.4.2 Bayesian Network classifier

The Bayesian Network classifier [9] searches for a Bayesian Network that best matches the feature probability distribution in the training set of instances. This classifier therefore

avoids the naïve assumption that the features are statistically independent. Once a suitable network is found, it can be used to predict the class label of new instances.

### 2.4.3 Support Vector Machines (SVM) classifier

The SVM classifier [2] projects instances to a higher dimensional space by using a kernel function, and searches for a hyper-surface that separates the training set of instances into the two classes with the widest possible margin between the boundary instances of the two classes. These boundary instances are called support vectors. Once a suitable hyper-surface is found, it can be used to predict the class label of new instances. In this study, the SVM classifier was applied using the Weka integrated LIBSVM [10] implementation.

### 2.4.4 Logistic classifier

The Logistic classifier [11] is a probabilistic classifier based on logistic regression with a ridge estimator. Logistic regression measures the relationship between the class label and the features by using probability scores.

### 2.4.5 Instance-Based K-Nearest Neighbors (IBk) classifier

The IBk classifier [12] is a lazy instance-based learning classifier derived from the nearest neighbors algorithm [13]. Instead of learning a classification model from the training set of instances, this classifier predicts the class label of a new instance based on the proximity of that instance to a certain number (k) of training instances.

### 2.4.6 Bagging classifier

The Bagging classifier [14] is an ensemble classifier that uses random samples of the training set of instances to construct multiple classification models of a certain classifier. In

this study, the Reduced-Error Pruning Tree (REPTree) classifier is used. The REPTree classifier uses information gain as the splitting criterion. The Bagging classifier predicts the class label of new instances by aggregating the predictions of the constructed REPTree classification models.

### 2.4.7 Random Committee classifier

The Random Committee classifier [1] is an ensemble classifier that uses the entire training set of instances to construct multiple classification models of a certain classifier with different random number seeds. In this study, the Random Tree classifier is used. The Random Tree classifier constructs a decision tree classification model that considers a random feature subset at each node controlled by the seed. The Random Committee classifier predicts the class label of new instances by aggregating the predictions of the constructed Random Tree classification models.

### 2.4.8 Random Subspace classifier

The Random Subspace classifier [15] is an ensemble classifier that uses the entire training set of instances to construct multiple classification models of a certain decision tree based classifier with different pseudo randomly selected feature subsets (subspaces). In this study, the REPTree classifier is used. The Random Subspace classifier predicts the class label of new instances by aggregating the predictions of the constructed REPTree classification models.

### 2.4.9 Rotation Forest classifier

The Rotation Forest [16] classifier is an ensemble classifier that uses random samples of the training set of instances to construct multiple classification models of a certain decision tree based classifier with the principal components of different disjoint feature subsets. In this study, the J48 classifier is used. The Rotation Forest classifier predicts the class label of new instances by aggregating the predictions of the constructed J48 classification models.

### 2.4.10 Threshold Selector classifier

The Threshold Selector classifier [1] selects a mid-point threshold on the output of a probabilistic classifier such that a given performance measure is optimized. In this study, the Logistic classifier is used and the F-Measure [1] is optimized.

### 2.4.11 Decision Table classifier

The Decision Table classifier [17] is a simple decision table majority classifier. It predicts the class label of a new instance by the majority vote of any matching training instances in the decision table. If there are no matching training instances, it predicts the class label by the majority vote of all training instances in the decision table.

### 2.4.12 Non-Nested Generalized Exemplars (NNge) classifier

The NNge classifier [18] is an instance based classifier derived from the nearest neighbor algorithm. Instead of learning a classification model from the training set of instances, this classifier predicts the class label of a new instance based on the nearest non-nested generalized exemplar. These exemplars are sets of if-then rules that accurately

classify a certain number of training instances. These exemplars can also be viewed as non-overlapping hyper-rectangles in the feature space.

### 2.4.13 Alternating Decision Tree (ADTree) classifier

The ADTree classifier [19][20] is a decision tree based classifier where a new instance may follow multiple paths as opposed to a single path as is the case with other decision trees such as C4.5. An alternating decision tree consists of decision nodes which specify a condition, and prediction nodes which contain a single number. The class label of a new instance is predicted by following all paths that meet the conditions of the decision nodes and summing the number values of the prediction nodes. If the final sum is positive then the positive class is predicted.

### 2.4.14 J48 classifier

The J48 [1] classifier is an open source implementation of the C4.5 classifier [21] which is a decision tree based classifier where the decision nodes are generated according to the features with the highest normalized information gain. The class label of a new instance is predicted by following the path that meets the conditions of the decision nodes.

### 2.4.15 Random Forest classifier

The Random Forest classifier [22] uses random samples of the training set of instances to construct multiple Random Tree classification models with a fixed number of randomly selected features. The Random Forest classifier predicts the class label of new instances by aggregating the predictions of the constructed Random Tree classification models.

# CHAPTER 3: IMPLEMENTATION

## 3.1 DCFD Toolkit Overview

Figure 3-1 provides an overview of the tools – highlighted in blue – available in the DCFD toolkit. The following sections describe each tool.

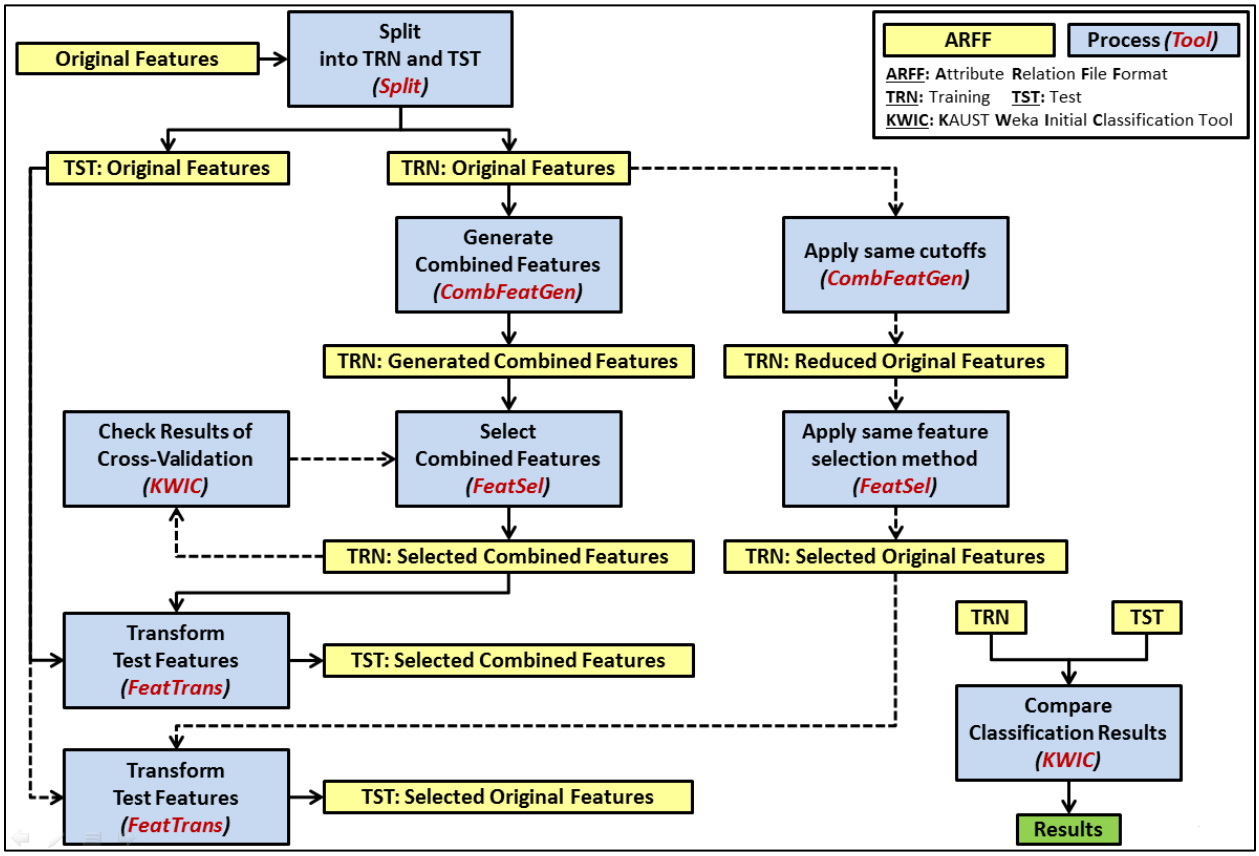


**Figure 3-1**: DCFD Toolkit Overview

## 3.2 Combined Features Generation (CombFeatGen) Tool

The Combined Features Generation Tool is an executable command line interface (CLI) program developed in the C programming language, and utilizes the uthash [23] header file for efficient search and the OpenMp [24] library for parallel processing. Given a binary class data set in a Weka compatible Attribute-Relation File Format (ARFF), the program

outputs the same data set with the original features plus generated combined features of a desired maximum combination size. The mandatory and optional parameters are listed in the following usage printout:

```
Program version: 1.19

usage: CombFeatGen label data [-s size] [-m method] [-o percent] [-d percent] [-w
wdir] [-p]
    label:            output label (alphanumeric, max 20 chars)
    data:             data file (arff format, max 200 chars)
    -s size:          max combination size (1-4, default 2)
    -m method:        combined features value calculation method (add/mult, default
mult)
    -o percent:       apply occurrence cutoff percent (1-99)
    -d percent:       apply difference cutoff percent (1-99)
    -w wdir:          working directory (max 200 chars)
    -p:               pause at the end of the program
```

## 3.3 Feature Selection (FeatSel) Tool

The Feature Selection Tool is also an executable CLI program developed in the C programming language and also utilizes the uthash header file for efficient search and the OpenMP library for parallel processing. Given a binary class data set in a Weka compatible ARFF, the program outputs the same data set with the original features replaced by the desired number of top ranked features by LEP, ADP, DIP, or p-value. The mandatory and optional parameters are listed in the following usage printout:

```
Program version: 1.25

usage: FeatSel label data num [-m method] [-z threshold] [-t features] [-w wdir] [-p]
    label:            output label (alphanumeric, max 20 chars)
    data:             data file (arff format, max 200 chars)
    num:              number of features to select
    -m method:        feature selection method (le/ad/dip/pv, default le)
    -z threshold:     fixed threshold (default exhaustive threshold search)
    -t features:      write sorted comparison table with (all/sel) features
    -w wdir:          working directory (max 200 chars)
    -p:               pause at the end of the program
```

## 3.4 KAUST Weka Initial Classification (KWIC) Tool

The KAUST Weka Initial Classification (KWIC) Tool is an executable Java Archive (JAR) file that provides a Graphical User Interface (GUI) for running and reporting the results of multiple Weka classifiers using default parameters. The tool also provides various functions that are convenient to use when conducting experiments with original and generated combined features. Figure 3-2 shows the main user interface. The run button applies any selected Weka classifier implementations on the given data set either with cross validation or against a given test data set and writes the results summary to a generated *<output label>_Results* text file. Runtime output is displayed in the Output text area and any occurring errors, exceptions, or warnings are displayed in the Errors text area.

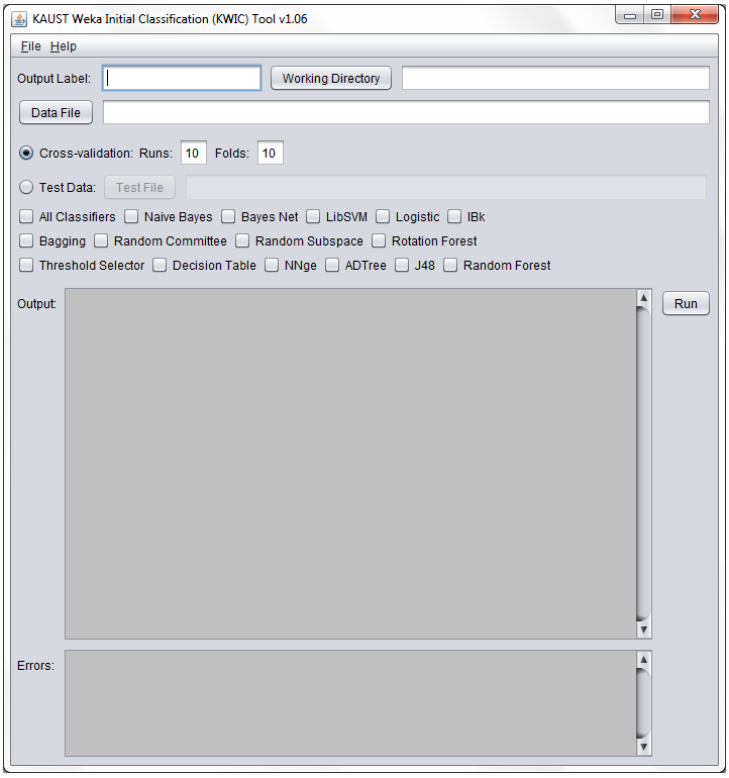


**Figure 3-2**: KWIC: Main User Interface

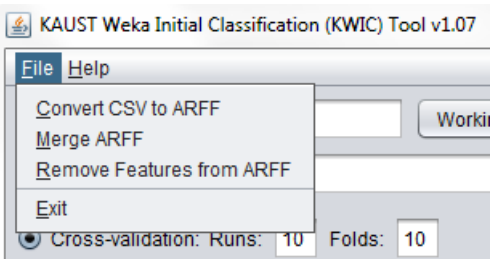


**Figure 3-3**: KWIC: File Menu

The following additional functions may be accessed from the File menu as shown in Figure 3-3:

1) **Convert CSV to ARFF**: Allows the user to convert data files from the comma-separated values (CSV) file format to the Weka compatible ARFF file format. The associated user interface is shown in Figure 3-4.


**Figure 3-4**: KWIC: Convert CSV to ARFF User Interface

2) **Merge ARFF**: Allows the user to merge two ARFF files that contain the same data set with different feature sets into a new ARFF file containing the same data set with both feature sets. This function was commonly used to merge original features with

generated combined features. Note that any common features must first be removed from one of the files. The associated user interface is shown in Figure 3-5.


**Figure 3-5**: KWIC: Merge ARFF User Interface

3) **Remove Features from ARFF**: Allows the user to remove a certain number of features from the start or end of the features list. This function was commonly used to extract a smaller number of the top ranked features without the need to rerun the FeatSel tool. The associated user interface is shown in Figure 3-6.

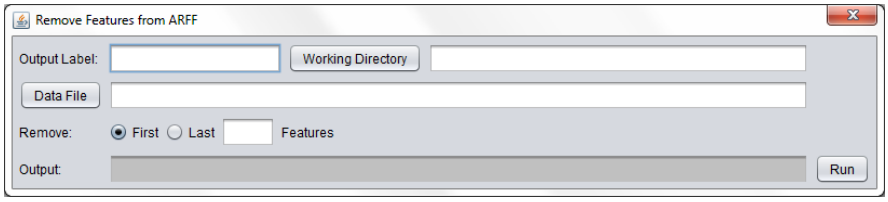


**Figure 3-6**: KWIC: Remove Features from ARFF User Interface

## 3.5 Split Tool

The Split Tool is an executable CLI program developed in the C programming language and utilizes the uthash header file for efficient search. Given a binary class data set in a Weka compatible ARFF, the program randomly splits the data set into training and test data sets according to a given test percentage. The resulting training data set is written to a *<data filename>_<percentage>TRN* ARFF file, and the resulting test data set is written to a *<data filename>_<percentage>TST* ARFF file. Given two binary class data sets representing the same instances, the program performs the same random split on both data sets. The mandatory and optional parameters are listed in the following usage printout:

```
Program version: 1.06

usage: Split data1 [-d data2] [-t percent] [-s seed] [-w wdir] [-p]
    data1:           data file to split (arff format, max 200 chars)
    -d data2:        2nd data file to split (arff format, max 200 chars)
    -t percent:      percentage of data marked for testing (1-99, default 10)
    -s seed:         random seed (default 1)
    -w wdir:         working directory (max 200 chars)
    -p:              pause at the end of the program
```

## 3.6 Feature Transformation (FeatTrans) Tool

The Feature Transformation Tool is an executable CLI program developed in the C programming language and utilizes the uthash header file for efficient search. Given two binary class data sets in Weka compatible ARFF where the features in the first data set are individual components of the combined features in the second data set, the program outputs the first data set with same combined features of the second data set. This is commonly used to transform the features in the test data set to the selected combined features in the training data set. This is necessary in order to perform classification on the same features for both the training and test instances. The mandatory and optional parameters are listed in the following usage printout:

```
Program version: 1.02

usage: FeatTrans label data feat [-m method] [-w wdir] [-p]
    label:           output label (alphanumeric, max 50 chars)
    data:            data file (arff format, max 250 chars)
    feat:            combined features file (arff format, max 250 chars)
    -m method:       combined features value calculation method (add/mult, default
mult)
    -w wdir:         working directory (max 250 chars)
    -p:              pause at the end of the program
```

# CHAPTER 4: EXPERIMENTS AND RESULTS

## 4.1 Experiment 0: Artificial Data Set

The small artificial data set shown in Table 4-1 was developed to provide a simple, compelling, reproducible, and efficiently executable example of the potential benefits of considering combined features in classification problems. It is also convenient to use it as an introductory example to better understand the different tools made available in the DCFD toolkit.

**Table 4-1**: Experiment 0: Artificial Data Set

| positive | | | | | negative | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 0 | 1 | 4 | 3 | 5 |
| 5 | 1 | 2 | 3 | 4 | 4 | 0 | 1 | 2 | 3 |
| 4 | 5 | 0 | 2 | 3 | 3 | 4 | 0 | 0 | 2 |
| 3 | 4 | 5 | 1 | 2 | 2 | 3 | 4 | 6 | 1 |
| 4 | 5 | 1 | 2 | 3 | 1 | 2 | 5 | 4 | 0 |
| 5 | 1 | 2 | 3 | 4 | 5 | 0 | 1 | 4 | 3 |
| 4 | 5 | 1 | 2 | 3 | 3 | 4 | 0 | 1 | 2 |
| 3 | 4 | 5 | 0 | 2 | 2 | 3 | 4 | 0 | 0 |
| 2 | 3 | 4 | 5 | 1 | 1 | 2 | 3 | 4 | 6 |
| 3 | 4 | 5 | 1 | 2 | 0 | 1 | 2 | 5 | 4 |
| 4 | 5 | 1 | 2 | 3 | 3 | 5 | 0 | 1 | 4 |
| 3 | 4 | 5 | 1 | 2 | 2 | 3 | 4 | 0 | 1 |
| 2 | 3 | 4 | 5 | 0 | 0 | 2 | 3 | 4 | 0 |
| 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 | 1 | 4 | 0 | 1 | 2 | 5 |
| 3 | 4 | 5 | 1 | 2 | 4 | 3 | 5 | 0 | 1 |
| 2 | 3 | 4 | 5 | 1 | 1 | 2 | 3 | 4 | 0 |
| 0 | 2 | 3 | 4 | 5 | 0 | 0 | 2 | 3 | 4 |
| 5 | 1 | 2 | 3 | 4 | 4 | 6 | 1 | 2 | 3 |
| 4 | 5 | 1 | 2 | 3 | 5 | 4 | 0 | 1 | 2 |
| 2 | 3 | 4 | 5 | 1 | 1 | 4 | 3 | 5 | 0 |
| 1 | 2 | 3 | 4 | 5 | 0 | 1 | 2 | 3 | 4 |
| 5 | 0 | 2 | 3 | 4 | 4 | 0 | 0 | 2 | 3 |
| 4 | 5 | 1 | 2 | 3 | 3 | 4 | 6 | 1 | 2 |
| 5 | 1 | 2 | 3 | 4 | 2 | 5 | 4 | 0 | 1 |

### 4.1.1 Experiment 0: Data

The artificial data set consists of 50 instances and 5 features (F1, F2, F3, F4, F5) whose values range between 0 and 6. The instances are evenly divided into 25 positive and 25

negative instances. The negative instances are slightly sparser than the positive, i.e. they contain more 0 feature values.

### 4.1.2 Experiment 0: Steps

#### Step 1: Split

The original features data set was randomly split into 80% training and 20% test data sets. The runtime printout of the Split tool is shown below:

```
>Split Exp0.arff -t 20

Program (v1.06) launched with:
  Test Percentage: 20%
  Random Seed: 1

=====================================================================
Data File: Exp0.arff
=====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 5 features
              50 instances
Randomly shuffling data & assigning to test/train..     done <0s>
  10 test instances
  40 train instances
Splitting data & writing files..                        done <0s>
=====================================================================

Generated Files:
Exp0_20TST.arff
Exp0_80TRN.arff

Program completed successfully <0s>
```

#### Step 2: Generate Combined Features

Using the CombFeatGen tool, a total of 30 combined features were generated from the original 5 features training data set using a maximum combination size of four. Note that the original 5 features are included in the total of 30 combined features. The runtime printout of the CombFeatGen tool is shown below:

```
>CombFeatGen Exp0_4m_80TRN Exp0_80TRN.arff -s 4

Program (v1.19) launched with:
  Max Combination Size: 4
```

```
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 5 features
              22 positive instances
              18 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
================================================================

================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
  5 features
Writing features to output ARFF..                       done <0s>
================================================================

================================================================
Generating 2-feature combinations..                     done <0s>
  10 2-features
Generating positive 2-feature vectors..                 done <0s>
Updating 2-feature comparison table positive column..   done <0s>
Generating negative 2-feature vectors..                 done <0s>
Updating 2-feature comparison table negative column..   done <0s>
Appending 2-features to output ARFF..                   done <0s>
Writing 2-feature vectors to temporary file..           done <0s>
================================================================

================================================================
Generating 3-feature combinations..                     done <0s>
  10 3-features
Generating positive 3-feature vectors..                 done <0s>
Updating 3-feature comparison table positive column..   done <0s>
Generating negative 3-feature vectors..                 done <0s>
Updating 3-feature comparison table negative column..   done <0s>
Appending 3-features to output ARFF..                   done <0s>
Writing 3-feature vectors to temporary file..           done <0s>
================================================================

================================================================
Generating 4-feature combinations..                     done <0s>
  5 4-features
Generating positive 4-feature vectors..                 done <0s>
Updating 4-feature comparison table positive column..   done <0s>
Generating negative 4-feature vectors..                 done <0s>
Updating 4-feature comparison table negative column..   done <0s>
Appending 4-features to output ARFF..                   done <0s>
Writing 4-feature vectors to temporary file..           done <0s>
================================================================

Appending all feature vectors to output ARFF..          done <0s>

Total Extracted Features: 30

Generated Files:
Exp0_4m_80TRN.arff

Program completed successfully <0s>
```

**Step 3: Select Features**

Several feature selection methods were applied on the generated combined features training data set using the FeatSel tool. Each of the resulting selected feature subsets were tested via cross validation using the KWIC tool. The best results were obtained from the top 10 features ranked by LEP according to an exhaustive threshold search. Tables 4-2, 4-3, and 4-4 show comparisons of the top ranked combined features versus the top ranked original features by LEP, DIP, and ADP, respectively. The runtime printout of the FeatSel tool is shown below:

```
>FeatSel Exp0_4m_10le_80TRN Exp0_4m_80TRN.arff 10 -m le -t sel

Program (v1.25) launched with:
  Number of Features to Select: 10
  Selection Method: least error with exhaustive threshold search

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 30 features
              22 positive instances
              18 negative instances
Reading data (loading into memory)..
  Reading features..                                    done <0s>
  Generating positive and negative feature vectors..    done <0s>
================================================================

================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
Sorting feature comparison table..                      done <0s>
Writing feature comparison table..                      done <0s>
Selecting top records of feature comparison table..     done <0s>
  10 selected features
Generating positive selected feature vectors..          done <0s>
Generating negative selected feature vectors..          done <0s>
================================================================

Writing output ARFF..                                   done <0s>

Total Selected Features: 10

Generated Files:
Exp0_4m_10le_80TRN.arff
Exp0_4m_10le_80TRN_FeatCompTable.txt

Program completed successfully <0s>
```

**Table 4-2**: Experiment 0: Top Features Ranked by LEP on Training Data

| Original Features | | | | | | | Combined Features | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feature | Threshold | TP | FP | TN | FN | Error % | Feature | Threshold | TP | FP | TN | FN | Error % |
| F1 | > 1 | 20 | 11 | 7 | 2 | 32.5 | F1_F3_F4_F5 | > 0 | 18 | 4 | 14 | 4 | 20 |
| F3 | > 0 | 21 | 13 | 5 | 1 | 35 | F1_F4 | > 5 | 16 | 3 | 15 | 6 | 22.5 |
| F4 | > 1 | 18 | 10 | 8 | 4 | 35 | F1_F3_F4 | > 0 | 19 | 6 | 12 | 3 | 22.5 |
| F5 | > 0 | 21 | 14 | 4 | 1 | 37.5 | F1_F3_F5 | > 0 | 19 | 6 | 12 | 3 | 22.5 |
| F2 | > 2 | 15 | 9 | 9 | 7 | 40 | F1_F2_F3_F4 | > 24 | 17 | 4 | 14 | 5 | 22.5 |
| | | | | | | | F1_F3 | > 3 | 19 | 7 | 11 | 3 | 25 |
| | | | | | | | F1_F2_F4 | > 8 | 18 | 6 | 12 | 4 | 25 |
| | | | | | | | F3_F4_F5 | > 0 | 19 | 7 | 11 | 3 | 25 |
| | | | | | | | F1_F2_F3_F5 | > 0 | 18 | 6 | 12 | 4 | 25 |
| | | | | | | | F3_F5 | > 0 | 20 | 9 | 9 | 2 | 27.5 |

**Table 4-3**: Experiment 0: Top Features Ranked by DIP on Training Data

| Original Features | | | | | | | Combined Features | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feature | Threshold | TP | FP | TN | FN | DIP | Feature | Threshold | TP | FP | TN | FN | DIP |
| F1 | > 2 | 15 | 8 | 10 | 7 | 0.5466 | F1_F3_F4_F5 | > 0 | 18 | 4 | 14 | 4 | 0.2871 |
| F4 | > 1 | 18 | 10 | 8 | 4 | 0.5846 | F1_F2_F3_F4 | > 24 | 17 | 4 | 14 | 5 | 0.3179 |
| F2 | > 2 | 15 | 9 | 9 | 7 | 0.5927 | F1_F4 | > 5 | 16 | 3 | 15 | 6 | 0.3196 |
| F5 | > 2 | 13 | 8 | 10 | 9 | 0.6041 | F1_F3_F4 | > 0 | 19 | 6 | 12 | 3 | 0.3601 |
| F3 | > 3 | 9 | 6 | 12 | 13 | 0.6784 | F1_F3_F5 | > 0 | 19 | 6 | 12 | 3 | 0.3601 |
| | | | | | | | F1_F2_F4 | > 8 | 18 | 6 | 12 | 4 | 0.3797 |
| | | | | | | | F1_F2_F3_F5 | > 0 | 18 | 6 | 12 | 4 | 0.3797 |
| | | | | | | | F1_F3 | > 3 | 19 | 7 | 11 | 3 | 0.4121 |
| | | | | | | | F3_F4_F5 | > 0 | 19 | 7 | 11 | 3 | 0.4121 |
| | | | | | | | F1_F2_F4_F5 | > 24 | 15 | 5 | 13 | 7 | 0.4224 |

**Table 4-4**: Experiment 0: Top Features Ranked by ADP on Training Data

| Original Features | | | | Combined Features | | | |
|---|---|---|---|---|---|---|---|
| Feature | Positive | Negative | Abs Diff % | Feature | Positive | Negative | Abs Diff % |
| F3 | 21 | 13 | 38.10 | F1_F3_F4_F5 | 18 | 4 | 77.78 |
| F1 | 21 | 14 | 33.33 | F1_F3_F4 | 19 | 6 | 68.42 |
| F4 | 21 | 14 | 33.33 | F1_F3_F5 | 19 | 6 | 68.42 |
| F5 | 21 | 14 | 33.33 | F1_F2_F3_F4 | 18 | 6 | 66.67 |
| F2 | 21 | 17 | 19.05 | F1_F2_F3_F5 | 18 | 6 | 66.67 |
| | | | | F3_F4_F5 | 19 | 7 | 63.16 |
| | | | | F1_F2_F4_F5 | 18 | 7 | 61.11 |
| | | | | F2_F3_F4_F5 | 18 | 7 | 61.11 |
| | | | | F1_F4_F5 | 19 | 8 | 57.89 |
| | | | | F1_F3 | 20 | 9 | 55.00 |

**Step 4: Transform Features**

The original features in the test data set were transformed to the selected combined features in the training data set using the FeatTrans tool. The runtime printout of the FeatTrans tool is shown below:

```
>FeatTrans Exp0_4m_10le_20TST Exp0_20TST.arff Exp0_4m_10le_80TRN.arff

Program (v1.02) launched with:
  Data File: Exp0_20TST.arff
  Features File: Exp0_4m_10le_80TRN.arff
  Combined Features Value Calculation Method: multiplication

=====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 5 features
              3 positive instances
              7 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
=====================================================================

=====================================================================
Reading target combined features..                      done <0s>
  10 target combined features
=====================================================================

=====================================================================
Writing output file..                                   done <0s>
=====================================================================

Generated Files:
Exp0_4m_10le_20TST.arff

Program completed successfully <0s>
```

**Step 5: Obtain Classification Results**

The KWIC tool was used to obtain the classification results when using the original features and when using the selected combined features. Figures 4-1 and 4-2 show the associated KWIC screenshots.

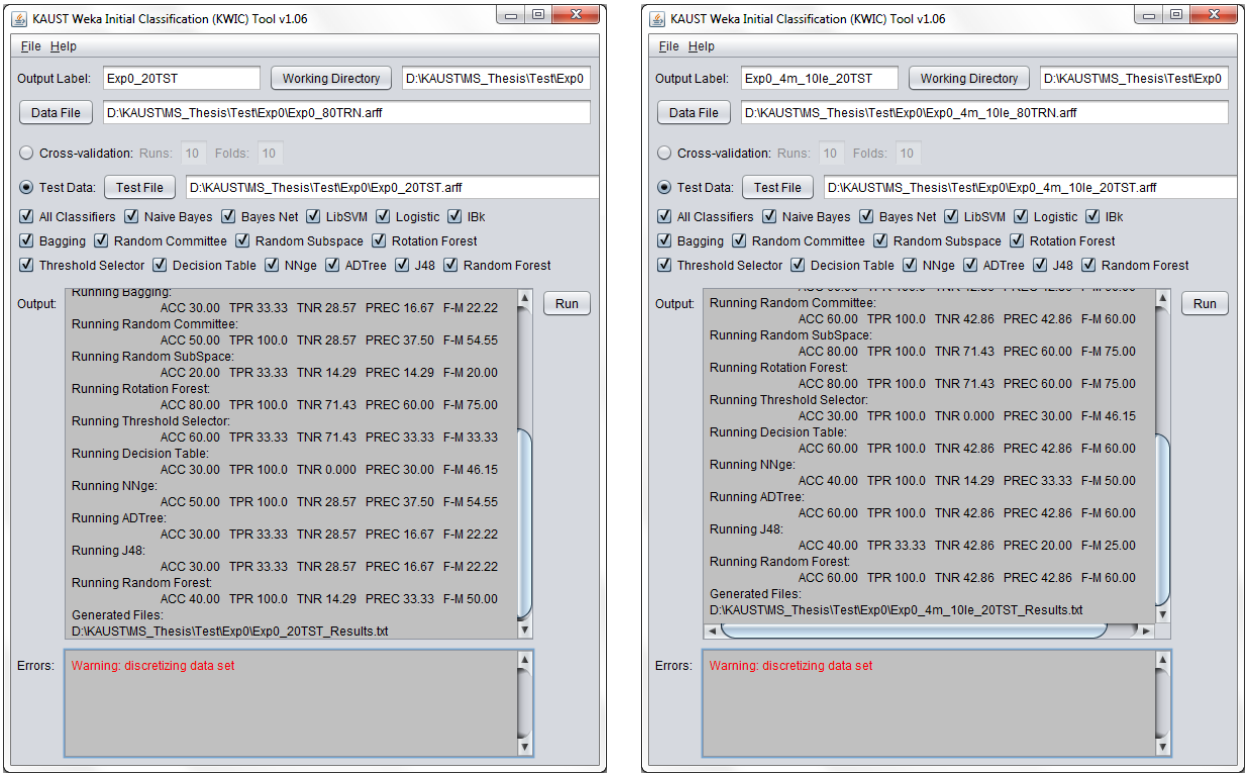


**Figure 4-1**: Experiment 0: KWIC Results Screenshots for Original and Combined Features

### 4.1.3 Experiment 0: Classification Results Comparison

Table 4-5 compares the classification results obtained with the original features versus those obtained with the selected combined features. In general we see a clear improvement in the classification results for the majority of classifiers. The Rotation Forest classifier provided the best results (ACC 80%) for the 5 original features, but perfect results (ACC 100%) were obtained from the SVM classifier using the 10 selected combined features.

Of course this is a biased artificially constructed example, but it demonstrates the potential benefits of considering combined features when addressing certain classification problems.

**Table 4-5**: Experiment 0: Classification Results Comparison

| | Original Features | | | | | Combined Features | | | | | Difference | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Classifier** | **ACC** | **TPR** | **TNR** | **PRC** | **F-M** | **ACC** | **TPR** | **TNR** | **PRC** | **F-M** | **ACC** | **TPR** | **TNR** | **PRC** | **F-M** |
| NaiveBayes | 50.0 | 33.3 | 57.1 | 25.0 | 28.6 | 60.0 | 33.3 | 71.4 | 33.3 | 33.3 | **10.0** | **0.0** | **14.3** | **8.3** | **4.8** |
| BayesNet | 30.0 | 100.0 | 0.0 | 30.0 | 46.2 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **30.0** | **0.0** | **42.9** | **12.9** | **13.8** |
| LibSVM | 40.0 | 33.3 | 42.9 | 20.0 | 25.0 | **100.0** | **100.0** | **100.0** | **100.0** | **100.0** | **60.0** | **66.7** | **57.1** | **80.0** | **75.0** |
| Logistic | 70.0 | 100.0 | 57.1 | 50.0 | 66.7 | 40.0 | 33.3 | 42.9 | 20.0 | 25.0 | **-30.0** | **-66.7** | **-14.3** | **-30.0** | **-41.7** |
| IBk | 50.0 | 100.0 | 28.6 | 37.5 | 54.5 | 70.0 | 100.0 | 57.1 | 50.0 | 66.7 | **20.0** | **0.0** | **28.6** | **12.5** | **12.1** |
| Bagging | 30.0 | 33.3 | 28.6 | 16.7 | 22.2 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **30.0** | **66.7** | **14.3** | **26.2** | **37.8** |
| RandomCommittee | 50.0 | 100.0 | 28.6 | 37.5 | 54.5 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **10.0** | **0.0** | **14.3** | **5.4** | **5.5** |
| RandomSubSpace | 20.0 | 33.3 | 14.3 | 14.3 | 20.0 | 80.0 | 100.0 | 71.4 | 60.0 | 75.0 | **60.0** | **66.7** | **57.1** | **45.7** | **55.0** |
| RotationForest | **80.0** | **100.0** | **71.4** | **60.0** | **75.0** | 80.0 | 100.0 | 71.4 | 60.0 | 75.0 | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| ThresholdSelector | 60.0 | 33.3 | 71.4 | 33.3 | 33.3 | 30.0 | 100.0 | 0.0 | 30.0 | 46.2 | **-30.0** | **66.7** | **-71.4** | **-3.3** | **12.8** |
| DecisionTable | 30.0 | 100.0 | 0.0 | 30.0 | 46.2 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **30.0** | **0.0** | **42.9** | **12.9** | **13.8** |
| NNge | 50.0 | 100.0 | 28.6 | 37.5 | 54.5 | 40.0 | 100.0 | 14.3 | 33.3 | 50.0 | **-10.0** | **0.0** | **-14.3** | **-4.2** | **-4.5** |
| ADTree | 30.0 | 33.3 | 28.6 | 16.7 | 22.2 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **30.0** | **66.7** | **14.3** | **26.2** | **37.8** |
| J48 | 30.0 | 33.3 | 28.6 | 16.7 | 22.2 | 40.0 | 33.3 | 42.9 | 20.0 | 25.0 | **10.0** | **0.0** | **14.3** | **3.3** | **2.8** |
| RandomForest | 40.0 | 100.0 | 14.3 | 33.3 | 50.0 | 60.0 | 100.0 | 42.9 | 42.9 | 60.0 | **20.0** | **0.0** | **28.6** | **9.5** | **10.0** |

## 4.2 Experiment 1: Gene Expression Prediction

### 4.2.1 Experiment 1: Data

Annotated feature vectors for this experiment were provided by Ghofran Othoum, fellow colleague at KAUST and they are obtained as follows. Dragon Database for Exploration of Ovarian Cancer Genes [25] contains information about genes implicated in ovarian cancer that are collected using reliable experimental evidence. This database was used to extract information about genes that are over-expressed and genes that are under-expressed in ovarian cancer. In total, 227 over-expressed genes and 41 under-expressed ones were identified from the database. Each gene is considered an instance, while the labels were over-expressed and under-expressed. For each of the genes, a DNA region covering 2000 bp (base pairs) upstream and 2000 bp downstream of the gene end was extracted. To generate the features that can describe this dataset, 800 DNA motifs were generated using DMF tool [26] [27]. Of these, 400 were generated using 206 over-expressed genes as the target set, while 20 under-expressed genes were considered as the background set. The remaining 400 DNA motifs were identified using 20 under-expressed genes as the target,

while 206 over-expressed genes were used as the background set. These motifs were mapped back to the two remaining sets of 21 over-expressed genes and 21 under-expressed genes to create the feature vectors for each instance. Each feature value represents the frequency of the associated motifs in each of the instances. An instance is labeled positive if the associated gene is over-expressed, and negative otherwise.

### 4.2.2 Experiment 1: Steps

**Step 1: Split**

The original data set of 42 instances (21 over-expressed and 21 under-expressed genes) was randomly split into 60% training and 40% test data sets. The runtime printout of the Split tool is shown below:

```
>Split Exp1.arff -t 40

Program (v1.06) launched with:
  Test Percentage: 40%
  Random Seed: 1

=====================================================================
Data File: Exp1.arff
=====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 800 features
              42 instances
Randomly shuffling data & assigning to test/train..     done <0s>
  16 test instances
  26 train instances
Splitting data & writing files..                        done <0s>
=====================================================================

Generated Files:
Exp1_40TST.arff
Exp1_60TRN.arff

Program completed successfully <0s>
```

**Step 2: Generate Combined Features**

Using the CombFeatGen tool, a total of 35,812,175 combined features were generated from the original 800 features training data set using a maximum combination size of three,

an occurrence cutoff of 5%, and a difference cutoff of 5%. Without the specified cutoffs the tool would have attempted to extract over 85 million combined features. The values for the generated combined features were calculated by multiplying the individual feature values. The same occurrence and difference cutoffs were applied on the original features training data set which reduced the original 800 features to 706 features. The runtime printouts of the CombFeatGen tool are shown below:

```
>CombFeatGen Exp1_3m55_60TRN Exp1_60TRN.arff -s 3 -o 5 -d 5

Program (v1.19) launched with:
  Max Combination Size: 3
  Occurrence Cutoff: 5%
  Difference Cutoff: 5%
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 800 features
              14 positive instances
              12 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
================================================================

================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
  800 features
Marking features to be ignored..                        done <0s>
  800 - 94 = 706 features
Reducing positive feature vectors..                     done <0s>
Reducing negative feature vectors..                     done <0s>
Writing features to output ARFF..                       done <0s>
================================================================

================================================================
Generating 2-feature combinations..                     done <0s>
  248865 2-features
Generating positive 2-feature vectors..                 done <0s>
Updating 2-feature comparison table positive column..   done <0s>
Generating negative 2-feature vectors..                 done <0s>
Updating 2-feature comparison table negative column..   done <0s>
Marking 2-features to be ignored..                      done <0s>
  248865 - 31594 = 217271 2-features
Reducing positive 2-feature vectors..                   done <0s>
Reducing negative 2-feature vectors..                   done <0s>
Reducing 2-feature comparison table..                   done <0s>
Appending 2-features to output ARFF..                   done <0s>
Writing 2-feature vectors to temporary file..           done <0s>
================================================================

================================================================
```

```
Generating 3-feature combinations..                      done <37s>
  39685780 3-features
Generating positive 3-feature vectors..                  done <13s>
Updating 3-feature comparison table positive column..    done <4s>
Generating negative 3-feature vectors..                  done <12s>
Updating 3-feature comparison table negative column..    done <4s>
Marking 3-features to be ignored..                       done <0s>
  39685780 - 4091582 = 35594198 3-features
Reducing positive 3-feature vectors..                    done <7s>
Reducing negative 3-feature vectors..                    done <5s>
Reducing 3-feature comparison table..                    done <1s>
Appending 3-features to output ARFF..                    done <32s>
Writing 3-feature vectors to temporary file..            done <128s>
================================================================

Appending all feature vectors to output ARFF..           done <26s>

Total Extracted Features: 35812175

Generated Files:
Exp1_3m55_60TRN.arff

Program completed successfully <282s>

>CombFeatGen Exp1_1m55_60TRN Exp1_60TRN.arff -s 1 -o 5 -d 5

Program (v1.19) launched with:
  Max Combination Size: 1
  Occurrence Cutoff: 5%
  Difference Cutoff: 5%
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..           done <0s>
  Identified: 800 features
              14 positive instances
              12 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..     done <0s>
================================================================

================================================================
Initializing feature comparison table..                  done <0s>
Updating feature comparison table..                      done <0s>
  800 features
Marking features to be ignored..                         done <0s>
  800 - 94 = 706 features
Reducing positive feature vectors..                      done <0s>
Reducing negative feature vectors..                      done <0s>
Writing features to output ARFF..                        done <0s>
================================================================

Appending all feature vectors to output ARFF..           done <0s>

Total Extracted Features: 706

Generated Files:
Exp1_1m55_60TRN.arff

Program completed successfully <0s>
```

**Step 2: Select Features**

Several feature selection methods were applied on the generated combined features training data set using the FeatSel tool. Each of the resulting selected feature subsets were tested via cross validation using the KWIC tool. The best results for the original features were obtained from the top 400 features ranked by DIP according to an exhaustive threshold search. The best results for the combined features were obtained from the top 600 features ranked by DIP also according to an exhaustive threshold search. Table 4-6 shows a comparison of the top 10 ranked combined features versus the top 10 ranked original features. The runtime printouts from selecting the top features ranked by DIP are shown below as a representative example:

```
>FeatSel Exp1_3m55_1000dip_60TRN Exp1_3m55_60TRN.arff 1000 -m dip -t sel

Program (v1.25) launched with:
  Number of Features to Select: 1000
  Selection Method: DIP with exhaustive threshold search

=================================================================
Reading data (counting features & instances)..          done <139s>
  Identified: 35812175 features
              14 positive instances
              12 negative instances
Reading data (loading into memory)..
  Reading features..                                    done <16s>
  Generating positive and negative feature vectors..    done <140s>
=================================================================

=================================================================
Initializing feature comparison table..                 done <24s>
Updating feature comparison table..                     done <309s>
Sorting feature comparison table..                      done <68s>
Writing feature comparison table..                      done <0s>
Selecting top records of feature comparison table..     done <0s>
  1000 selected features
Generating positive selected feature vectors..          done <0s>
Generating negative selected feature vectors..          done <0s>
=================================================================

Writing output ARFF..                                   done <0s>

Total Selected Features: 1000
```

```
Generated Files:
Exp1_3m55_1000dip_60TRN.arff
Exp1_3m55_1000dip_60TRN_FeatCompTable.txt

Program completed successfully <710s>

>FeatSel Exp1_1m55_706dip_60TRN Exp1_1m55_60TRN.arff 706 -m dip -t sel

Program (v1.25) launched with:
  Number of Features to Select: 706
  Selection Method: DIP with exhaustive threshold search

=====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 706 features
              14 positive instances
              12 negative instances
Reading data (loading into memory)..
  Reading features..                                    done <0s>
  Generating positive and negative feature vectors..    done <0s>
=====================================================================

=====================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
Sorting feature comparison table..                      done <0s>
Writing feature comparison table..                      done <0s>
Selecting top records of feature comparison table..     done <0s>
  706 selected features
Generating positive selected feature vectors..          done <0s>
Generating negative selected feature vectors..          done <0s>
=====================================================================

Writing output ARFF..                                   done <0s>

Total Selected Features: 706

Generated Files:
Exp1_1m55_706dip_60TRN.arff
Exp1_1m55_706dip_60TRN_FeatCompTable.txt

Program completed successfully <0s>
```

**Table 4-6**: Experiment 1: Top 10 Features Ranked by DIP on Training Data

| Original Features | | | | | | | Combined Features | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feature | Threshold | TP | FP | TN | FN | DIP | Feature | Threshold | TP | FP | TN | FN | DIP |
| F680 | <= 0 | 11 | 2 | 10 | 3 | 0.2715 | F56_F285_F374 | > 0 | 13 | 1 | 11 | 1 | 0.1098 |
| F35 | <= 0 | 12 | 3 | 9 | 2 | 0.2879 | F122_F168_F374 | > 0 | 13 | 1 | 11 | 1 | 0.1098 |
| F483 | > 0 | 10 | 1 | 11 | 4 | 0.2976 | F331_F374_F572 | > 10 | 13 | 1 | 11 | 1 | 0.1098 |
| F59 | <= 0 | 10 | 2 | 10 | 4 | 0.3308 | F9_F285_F374 | > 0 | 12 | 0 | 12 | 2 | 0.1429 |
| F66 | <= 1 | 12 | 4 | 8 | 2 | 0.3627 | F168_F285_F374 | > 0 | 12 | 0 | 12 | 2 | 0.1429 |
| F159 | > 1 | 12 | 4 | 8 | 2 | 0.3627 | F168_F365_F374 | > 0 | 12 | 0 | 12 | 2 | 0.1429 |
| F571 | <= 1 | 12 | 4 | 8 | 2 | 0.3627 | F282_F365_F776 | > 15 | 12 | 0 | 12 | 2 | 0.1429 |
| F54 | > 0 | 9 | 1 | 11 | 5 | 0.3667 | F388_F424_F501 | > 3 | 12 | 0 | 12 | 2 | 0.1429 |
| F370 | > 2 | 9 | 1 | 11 | 5 | 0.3667 | F302_F370 | > 20 | 12 | 1 | 11 | 2 | 0.1654 |
| F75 | <= 2 | 10 | 3 | 9 | 4 | 0.3796 | F9_F31_F122 | > 0 | 12 | 1 | 11 | 2 | 0.1654 |

### Step 4: Transform Features

The features in the test data sets were transformed to the selected features in the training data sets using the FeatTrans tool. The runtime printouts of the FeatTrans tool are shown below:

```
>FeatTrans Exp1_3m55_1000dip_40TST Exp1_40TST.arff Exp1_3m55_1000dip_60TRN.arff

Program (v1.02) launched with:
  Data File: Exp1_40TST.arff
  Features File: Exp1_3m55_1000dip_60TRN.arff
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 800 features
              7 positive instances
              9 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
================================================================

================================================================
Reading target combined features..                      done <0s>
  1000 target combined features
================================================================

================================================================
Writing output file..                                   done <0s>
================================================================

Generated Files:
Exp1_3m55_1000dip_40TST.arff

Program completed successfully <0s>

>FeatTrans Exp1_1m55_706dip_40TST Exp1_40TST.arff Exp1_1m55_706dip_60TRN.arff
```

```
Program (v1.02) launched with:
  Data File: Exp1_40TST.arff
  Features File: Exp1_1m55_706dip_60TRN.arff
  Combined Features Value Calculation Method: multiplication

=====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 800 features
              7 positive instances
              9 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
=====================================================================

=====================================================================
Reading target combined features..                      done <0s>
  706 target combined features
=====================================================================

=====================================================================
Writing output file..                                   done <0s>
=====================================================================

Generated Files:
Exp1_1m55_706dip_40TST.arff

Program completed successfully <0s>
```

### Step 5: Obtain Classification Results

The KWIC tool was used to obtain the classification results when using the original features and when using the selected combined features. Figures 4-2 shows the associated KWIC screenshots.

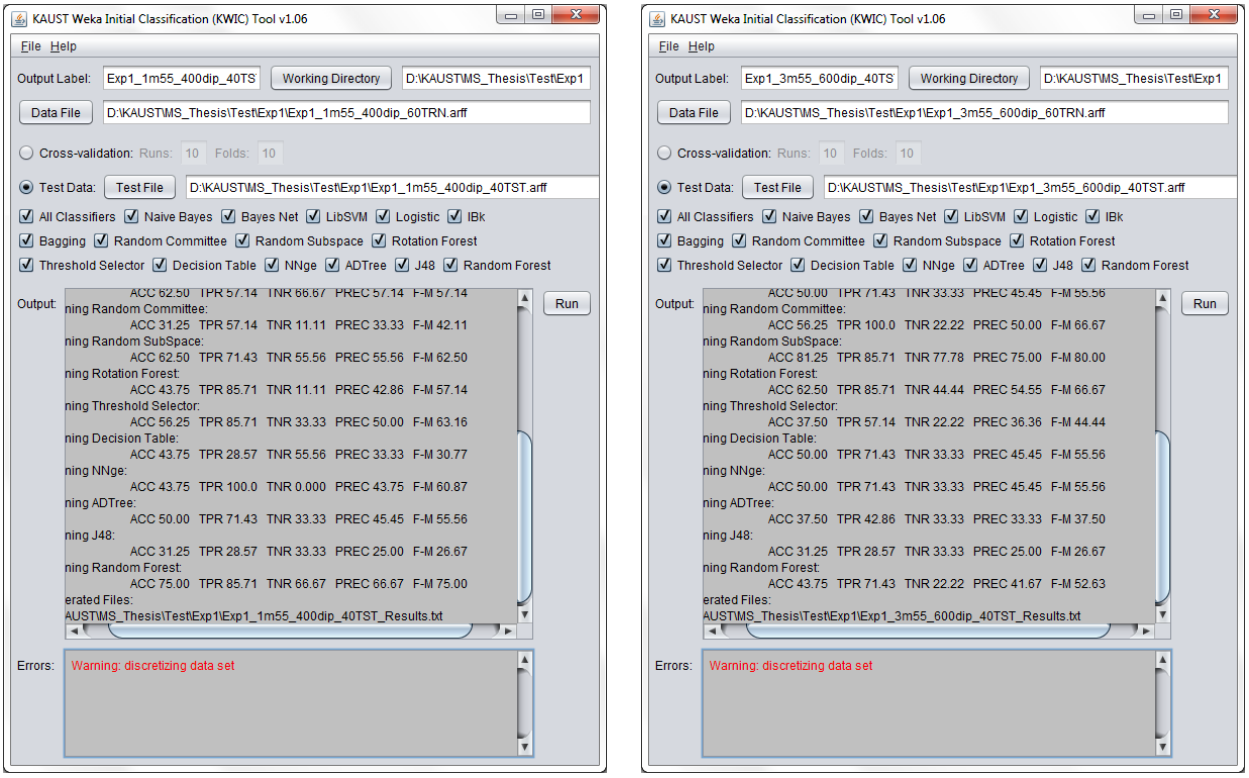


**Figure 4-2**: Experiment 1: KWIC Results Screenshots for Original and Combined Features

### 4.2.3 Experiment 1: Classification Results Comparison

Table 4-5 compares the classification results obtained with the selected original features versus those obtained with the selected combined features. The Random Forest classifier provided the best results for the selected original features (ACC 75%), but better results were obtained by the Random Subspace classifier using the selected combined features (ACC 81.3%).

**Table 4-7**: Experiment 1: Classification Results Comparison

| | Original Features | | | | | Combined Features | | | | | Difference | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifiers | ACC | TPR | TNR | PRC | F-M | ACC | TPR | TNR | PRC | F-M | ACC | TPR | TNR | PRC | F-M |
| NaiveBayes | 62.5 | 85.7 | 44.4 | 54.5 | 66.7 | 56.3 | 71.4 | 44.4 | 50.0 | 58.8 | -6.3 | -14.3 | 0.0 | -4.5 | -7.8 |
| BayesNet | 62.5 | 57.1 | 66.7 | 57.1 | 57.1 | 56.3 | 85.7 | 33.3 | 50.0 | 63.2 | -6.3 | 28.6 | -33.3 | -7.1 | 6.0 |
| LibSVM | 43.8 | 100.0 | 0.0 | 43.8 | 60.9 | 43.8 | 100.0 | 0.0 | 43.8 | 60.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Logistic | 50.0 | 71.4 | 33.3 | 45.5 | 55.6 | 37.5 | 57.1 | 22.2 | 36.4 | 44.4 | -12.5 | -14.3 | -11.1 | -9.1 | -11.1 |
| IBk | 37.5 | 42.9 | 33.3 | 33.3 | 37.5 | 37.5 | 71.4 | 11.1 | 38.5 | 50.0 | 0.0 | 28.6 | -22.2 | 5.1 | 12.5 |
| Bagging | 62.5 | 57.1 | 66.7 | 57.1 | 57.1 | 50.0 | 71.4 | 33.3 | 45.5 | 55.6 | -12.5 | 14.3 | -33.3 | -11.7 | -1.6 |
| RandomCommittee | 31.3 | 57.1 | 11.1 | 33.3 | 42.1 | 56.3 | 100.0 | 22.2 | 50.0 | 66.7 | 25.0 | 42.9 | 11.1 | 16.7 | 24.6 |
| RandomSubSpace | 62.5 | 71.4 | 55.6 | 55.6 | 62.5 | 81.3 | 85.7 | 77.8 | 75.0 | 80.0 | 18.8 | 14.3 | 22.2 | 19.4 | 17.5 |
| RotationForest | 43.8 | 85.7 | 11.1 | 42.9 | 57.1 | 62.5 | 85.7 | 44.4 | 54.5 | 66.7 | 18.8 | 0.0 | 33.3 | 11.7 | 9.5 |
| ThresholdSelector | 56.3 | 85.7 | 33.3 | 50.0 | 63.2 | 37.5 | 57.1 | 22.2 | 36.4 | 44.4 | -18.8 | -28.6 | -11.1 | -13.6 | -18.7 |
| DecisionTable | 43.8 | 28.6 | 55.6 | 33.3 | 30.8 | 50.0 | 71.4 | 33.3 | 45.5 | 55.6 | 6.3 | 42.9 | -22.2 | 12.1 | 24.8 |
| NNge | 43.8 | 100.0 | 0.0 | 43.8 | 60.9 | 50.0 | 71.4 | 33.3 | 45.5 | 55.6 | 6.3 | -28.6 | 33.3 | 1.7 | -5.3 |
| ADTree | 50.0 | 71.4 | 33.3 | 45.5 | 55.6 | 37.5 | 42.9 | 33.3 | 33.3 | 37.5 | -12.5 | -28.6 | 0.0 | -12.1 | -18.1 |
| J48 | 31.3 | 28.6 | 33.3 | 25.0 | 26.7 | 31.3 | 28.6 | 33.3 | 25.0 | 26.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| RandomForest | 75.0 | 85.7 | 66.7 | 66.7 | 75.0 | 43.8 | 71.4 | 22.2 | 41.7 | 52.6 | -31.3 | -14.3 | -44.4 | -25.0 | -22.4 |

## 4.3 Experiment 2: Promoter Region Recognition

### 4.3.1 Experiment 2: Data

Annotated feature vectors for this experiment were provided by Haitham Ashoor, fellow colleague at KAUST and they are obtained as follows. Data were retrieved from the ENCODE Consortium [28] and relates to ChIP-Seq data (ChIP-Seq data points to region where TF is more likely to bind, and it represents the positive class) for NF-kB (Nuclear Factor kappa-light-chain-enhancer of activated B cells) transcription factor binding to DNA in Gm12878 cell line. Raw ChIP-Seq data were processed by MACS program [29] to get the ChIP-Seq peaks. Top 1000 peaks were selected, with 500 used for training and 500 for testing. These peak sequences were annotated by motifs obtained using ChIPDragon pipeline [30] based on DMF software. The training data was used as the target set, while the background set was obtained by randomly selecting genomic regions of the same size and number that do not belong to the generated ChIP-Seq peaks. Features are selected as overrepresented motifs in the ChIP-Seq peaks over the random data. These motifs were

mapped to the testing data, which included additional 500 background sequences not used for motif generation. In total 313 features represented motif families that are found in those regions. Each feature value represents the frequency of the associated motifs. An instance is labeled positive if it represents a promoter that binds with the NF-kB transcription factor, and it is labeled negative otherwise (represents a random genomic region).

### 4.3.2 Experiment 2: Steps

**Step 1: Split**

As described in the previous section, the data is already split into 50% training and 50% test data sets, so there was no need to run the Split tool.

**Step 2: Generate Combined Features**

Using the CombFeatGen tool, a total of 325,623 combined features were generated from the original 313 features training data set using a maximum combination size of three and an occurrence cutoff of 1%. The values for the generated combined features were calculated by multiplying the individual feature values. The same occurrence cutoff was applied on the original features training data set which reduced the original 313 features to 125 features. The runtime printouts of the CombFeatGen tool are shown below:

```
>CombFeatGen Exp2_3m1_50TRN Exp2_50TRN.arff -s 3 -o 1

Program (v1.19) launched with:
  Max Combination Size: 3
  Occurrence Cutoff: 1%
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 313 features
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
================================================================
```

```
================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
  313 features
Marking features to be ignored..                        done <0s>
  313 - 188 = 125 features
Reducing positive feature vectors..                     done <0s>
Reducing negative feature vectors..                     done <0s>
Writing features to output ARFF..                       done <0s>
================================================================

================================================================
Generating 2-feature combinations..                     done <0s>
  7750 2-features
Generating positive 2-feature vectors..                 done <0s>
Updating 2-feature comparison table positive column..   done <0s>
Generating negative 2-feature vectors..                 done <0s>
Updating 2-feature comparison table negative column..   done <0s>
Marking 2-features to be ignored..                      done <0s>
  7750 - 0 = 7750 2-features
Appending 2-features to output ARFF..                   done <0s>
Writing 2-feature vectors to temporary file..           done <1s>
================================================================

================================================================
Generating 3-feature combinations..                     done <0s>
  317750 3-features
Generating positive 3-feature vectors..                 done <3s>
Updating 3-feature comparison table positive column..   done <0s>
Generating negative 3-feature vectors..                 done <3s>
Updating 3-feature comparison table negative column..   done <0s>
Marking 3-features to be ignored..                      done <0s>
  317750 - 2 = 317748 3-features
Reducing positive 3-feature vectors..                   done <0s>
Reducing negative 3-feature vectors..                   done <0s>
Reducing 3-feature comparison table..                   done <0s>
Appending 3-features to output ARFF..                   done <0s>
Writing 3-feature vectors to temporary file..           done <48s>
================================================================

Appending all feature vectors to output ARFF..          done <9s>

Total Extracted Features: 325623

Generated Files:
Exp2_3m1_50TRN.arff

Program completed successfully <69s>

>CombFeatGen Exp2_1m1_50TRN Exp2_50TRN.arff -s 1 -o 1

Program (v1.19) launched with:
  Max Combination Size: 1
  Occurrence Cutoff: 1%
  Combined Features Value Calculation Method: multiplication

================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 313 features
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
```

```
  Generating positive and negative feature vectors..    done <0s>
================================================================

================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
  313 features
Marking features to be ignored..                        done <0s>
  313 - 188 = 125 features
Reducing positive feature vectors..                     done <0s>
Reducing negative feature vectors..                     done <0s>
Writing features to output ARFF..                       done <0s>
================================================================

Appending all feature vectors to output ARFF..          done <0s>

Total Extracted Features: 125

Generated Files:
Exp2_1m1_50TRN.arff

Program completed successfully <0s>
```

### Step 3: Select Features

Several feature selection methods were applied on the generated combined features training data set using the FeatSel tool. Each of the resulting selected feature subsets were tested via cross validation using the KWIC tool. The best results for the combined features were obtained from the top 400 features ranked by LEP according to an exhaustive threshold search. The best results for the original features were obtained from the top 50 features ranked by LEP also according to an exhaustive threshold search. Table 4-8 shows a comparison of the top ten ranked selected combined features versus the top ten ranked original features by LEP. The runtime printouts from selecting the top features ranked by LEP are shown below as a representative example:

```
>FeatSel Exp2_3m1_1000le_50TRN Exp2_3m1_50TRN.arff 1000 -m le -t sel

Program (v1.26) launched with:
  Number of Features to Select: 1000
  Selection Method: least error with exhaustive threshold search

================================================================
Reading data (counting features & instances)..          done <37s>
  Identified: 325623 features
```

```
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
  Reading features..                                    done <0s>
  Generating positive and negative feature vectors..    done <57s>
====================================================================

====================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <1304s>
Sorting feature comparison table..                      done <0s>
Writing feature comparison table..                      done <0s>
Selecting top records of feature comparison table..     done <0s>
  1000 selected features
Generating positive selected feature vectors..          done <0s>
Generating negative selected feature vectors..          done <0s>
====================================================================

Writing output ARFF..                                   done <0s>

Total Selected Features: 1000

Generated Files:
Exp2_3m1_1000le_50TRN.arff
Exp2_3m1_1000le_50TRN_FeatCompTable.txt

Program completed successfully <1400s>

>FeatSel Exp2_1m1_125le_50TRN Exp2_1m1_50TRN.arff 125 -m le -t sel

Program (v1.26) launched with:
  Number of Features to Select: 125
  Selection Method: least error with exhaustive threshold search

====================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 125 features
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
  Reading features..                                    done <0s>
  Generating positive and negative feature vectors..    done <0s>
====================================================================

====================================================================
Initializing feature comparison table..                 done <0s>
Updating feature comparison table..                     done <0s>
Sorting feature comparison table..                      done <0s>
Writing feature comparison table..                      done <0s>
Selecting top records of feature comparison table..     done <0s>
  125 selected features
Generating positive selected feature vectors..          done <0s>
Generating negative selected feature vectors..          done <0s>
====================================================================

Writing output ARFF..                                   done <0s>

Total Selected Features: 125

Generated Files:
Exp2_1m1_125le_50TRN.arff
Exp2_1m1_125le_50TRN_FeatCompTable.txt
```

```
Program completed successfully <0s>
```

**Table 4-8**: Experiment 2: Top 10 Features Ranked by LEP on Training Data

| Original Features | | | | | | | Combined Features | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feature | Threshold | TP | FP | TN | FN | Error % | Feature | Threshold | TP | FP | TN | FN | Error % |
| F19 | > 0 | 414 | 201 | 299 | 86 | 28.7 | F19_F71_F86 | > 0 | 354 | 117 | 383 | 146 | 26.3 |
| F71 | > 0 | 404 | 214 | 286 | 96 | 31 | F19_F71 | > 0 | 375 | 139 | 361 | 125 | 26.4 |
| F86 | > 1 | 341 | 207 | 293 | 159 | 36.6 | F19_F86 | > 0 | 387 | 173 | 327 | 113 | 28.6 |
| F97 | > 1 | 129 | 66 | 434 | 371 | 43.7 | F19_F71_F106 | > 1 | 332 | 118 | 382 | 168 | 28.6 |
| F64 | <= 3 | 306 | 252 | 248 | 194 | 44.6 | F19 | > 0 | 414 | 201 | 299 | 86 | 28.7 |
| F62 | > 0 | 239 | 188 | 312 | 261 | 44.9 | F19_F36_F71 | > 0 | 330 | 117 | 383 | 170 | 28.7 |
| F54 | <= 2 | 251 | 203 | 297 | 249 | 45.2 | F19_F71_F85 | > 0 | 331 | 118 | 382 | 169 | 28.7 |
| F103 | <= 5 | 406 | 358 | 142 | 94 | 45.2 | F19_F30_F71 | > 0 | 334 | 122 | 378 | 166 | 28.8 |
| F2 | <= 5 | 363 | 316 | 184 | 137 | 45.3 | F19_F52_F71 | > 0 | 333 | 122 | 378 | 167 | 28.9 |
| F14 | <= 2 | 243 | 196 | 304 | 257 | 45.3 | F19_F71_F95 | > 0 | 331 | 121 | 379 | 169 | 29 |

### Step 4: Transform Features

The features in the test data sets were transformed to the selected features in the training data sets using the FeatTrans tool. The runtime printouts of the FeatTrans tool are shown below:

```
>FeatTrans Exp2_3m1_1000le_50TST Exp2_50TST.arff Exp2_3m1_1000le_50TRN.arff

Program (v1.03) launched with:
  Data File: Exp2_50TST.arff
  Features File: Exp2_3m1_1000le_50TRN.arff
  Combined Features Value Calculation Method: multiplication

=================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 313 features
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
=================================================================

=================================================================
Reading target combined features..                      done <0s>
  1000 target combined features
=================================================================

=================================================================
Writing output file..                                   done <0s>
=================================================================

Generated Files:
Exp2_3m1_1000le_50TST.arff

Program completed successfully <0s>
```

```
>FeatTrans Exp2_1m1_125le_50TST Exp2_50TST.arff Exp2_1m1_125le_50TRN.arff

Program (v1.03) launched with:
  Data File: Exp2_50TST.arff
  Features File: Exp2_1m1_125le_50TRN.arff
  Combined Features Value Calculation Method: multiplication

==================================================================
Reading data (counting features & instances)..          done <0s>
  Identified: 313 features
              500 positive instances
              500 negative instances
Reading data (loading into memory)..
  Generating positive and negative feature vectors..    done <0s>
==================================================================

==================================================================
Reading target combined features..                      done <0s>
  125 target combined features
==================================================================

==================================================================
Writing output file..                                   done <0s>
==================================================================

Generated Files:
Exp2_1m1_125le_50TST.arff

Program completed successfully <0s>
```

### Step 5: Obtain Classification Results

The KWIC tool was used to obtain the classification results when using the selected original features and when using the selected combined features. Figure 4-3 shows the associated KWIC screenshots.

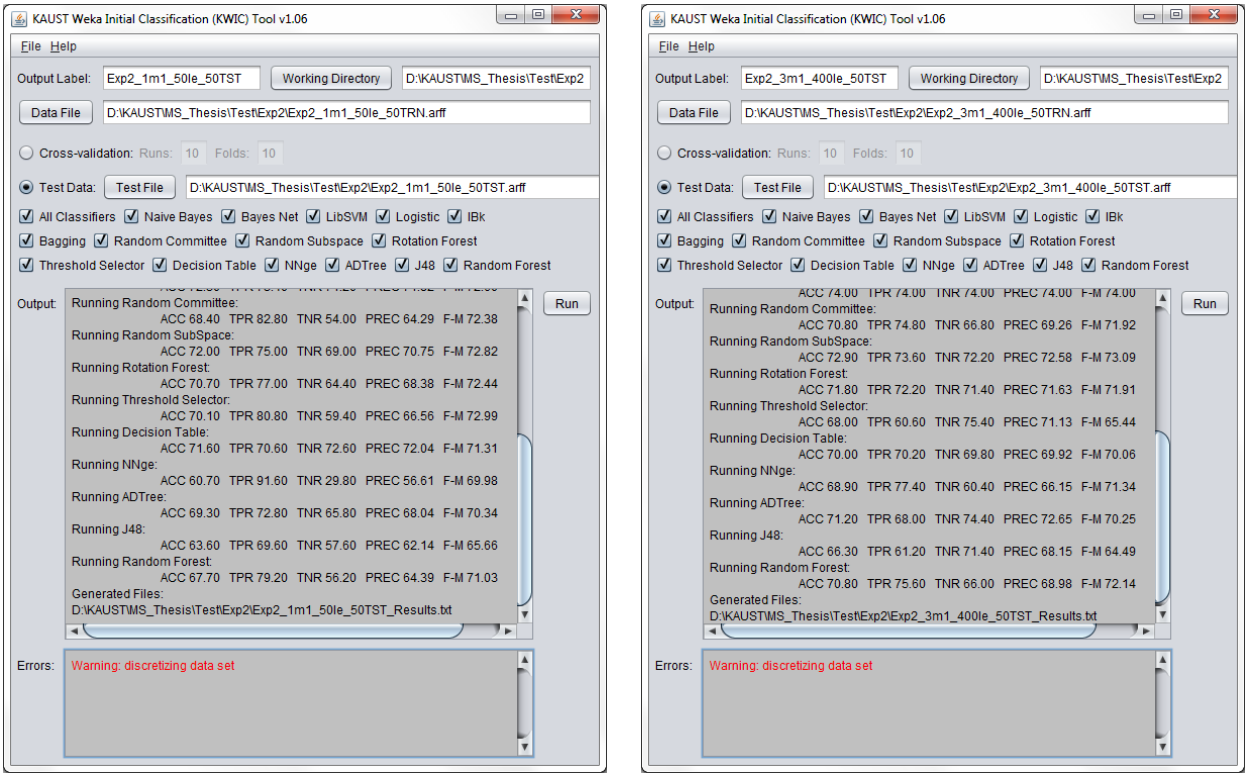


**Figure 4-3**: Experiment 2: KWIC Results Screenshots for Original and Combined Features

### 4.3.3 Experiment 2: Classification Results Comparison

Table 4-9 compares the classification results obtained with the selected original features versus those obtained with the selected combined features. The Bagging classifier provided the best results (ACC 72.3%) for the 50 selected original features, but slightly better results (ACC 74%) were obtained from the same classifier using the 400 selected combined features.

**Table 4-9**: Experiment 2: Classification Results Comparison

| | Original Features | | | | | Combined Features | | | | | Difference | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifiers | ACC | TPR | TNR | PRC | F-M | ACC | TPR | TNR | PRC | F-M | ACC | TPR | TNR | PRC | F-M |
| NaiveBayes | 56.6 | 85.8 | 27.4 | 54.2 | 66.4 | 55.4 | 18.2 | 92.6 | 71.1 | 29.0 | -1.2 | -67.6 | 65.2 | 16.9 | -37.4 |
| BayesNet | 70.6 | 68.8 | 72.4 | 71.4 | 70.1 | 66.5 | 67.4 | 65.6 | 66.2 | 66.8 | -4.1 | -1.4 | -6.8 | -5.2 | -3.3 |
| LibSVM | 59.3 | 81.8 | 36.8 | 56.4 | 66.8 | 69.1 | 79.0 | 59.2 | 65.9 | 71.9 | 9.8 | -2.8 | 22.4 | 9.5 | 5.1 |
| Logistic | 69.8 | 70.6 | 69.0 | 69.5 | 70.0 | 68.1 | 61.0 | 75.2 | 71.1 | 65.7 | -1.7 | -9.6 | 6.2 | 1.6 | -4.4 |
| IBk | 62.2 | 66.2 | 58.2 | 61.3 | 63.7 | 65.8 | 64.8 | 66.8 | 66.1 | 65.5 | 3.6 | -1.4 | 8.6 | 4.8 | 1.8 |
| Bagging | 72.3 | 73.4 | 71.2 | 71.8 | 72.6 | 74.0 | 74.0 | 74.0 | 74.0 | 74.0 | 1.7 | 0.6 | 2.8 | 2.2 | 1.4 |
| RandomCommittee | 68.4 | 82.8 | 54.0 | 64.3 | 72.4 | 70.8 | 74.8 | 66.8 | 69.3 | 71.9 | 2.4 | -8.0 | 12.8 | 5.0 | -0.5 |
| RandomSubSpace | 72.0 | 75.0 | 69.0 | 70.8 | 72.8 | 72.9 | 73.6 | 72.2 | 72.6 | 73.1 | 0.9 | -1.4 | 3.2 | 1.8 | 0.3 |
| RotationForest | 70.7 | 77.0 | 64.4 | 68.4 | 72.4 | 71.8 | 72.2 | 71.4 | 71.6 | 71.9 | 1.1 | -4.8 | 7.0 | 3.2 | -0.5 |
| ThresholdSelector | 70.1 | 80.8 | 59.4 | 66.6 | 73.0 | 68.0 | 60.6 | 75.4 | 71.1 | 65.4 | -2.1 | -20.2 | 16.0 | 4.6 | -7.5 |
| DecisionTable | 71.6 | 70.6 | 72.6 | 72.0 | 71.3 | 70.0 | 70.2 | 69.8 | 69.9 | 70.1 | -1.6 | -0.4 | -2.8 | -2.1 | -1.3 |
| NNge | 60.7 | 91.6 | 29.8 | 56.6 | 70.0 | 68.9 | 77.4 | 60.4 | 66.2 | 71.3 | 8.2 | -14.2 | 30.6 | 9.5 | 1.4 |
| ADTree | 69.3 | 72.8 | 65.8 | 68.0 | 70.3 | 71.2 | 68.0 | 74.4 | 72.6 | 70.2 | 1.9 | -4.8 | 8.6 | 4.6 | -0.1 |
| J48 | 63.6 | 69.6 | 57.6 | 62.1 | 65.7 | 66.3 | 61.2 | 71.4 | 68.2 | 64.5 | 2.7 | -8.4 | 13.8 | 6.0 | -1.2 |
| RandomForest | 67.7 | 79.2 | 56.2 | 64.4 | 71.0 | 70.8 | 75.6 | 66.0 | 69.0 | 72.1 | 3.1 | -3.6 | 9.8 | 4.6 | 1.1 |

# CHAPTER 5: CONCLUSION

## 5.1 Future Work

The natural next step is to provide a strong and unequivocal example of the beneficial use of combined features in a classification problem, including an analysis of the discovered combined features in terms of what they represent and why they improve the classification results of the problem being addressed.

From a functional perspective there is much room for improvement, perhaps the most prominent of which is the efficient implementation of more sophisticated feature selection options.

From a technical implementation perspective, there is also room for improvement. Although great care was taken to maximize the performance of developed tools in terms of processing time and memory utilization, there are likely untapped opportunities for further parallelization and more efficient designs. The introduction of web-based services for the developed tools would also be of great convenience to users.

## 5.2 Summary

This study has developed a methodology that enriches the description of the data by generation of new features from the original feature set. The methodology offers the use of a combination of original and newly derived features from which filter type selection of potentially useful combination of features is made. The filtering methods used some very simple performance assessment techniques to mitigate the explosion of the number of newly created features. In spite these simplifications, it is convincingly demonstrated that the new methodology can produce sometimes significantly better results. These are of

course data and problem dependent. Overall, this method contributes new approach that can be used for potentially improving classification performance.

The DCFD toolkit provides an efficient and convenient way to explore the use of combined features in binary classification problems. The experimental results described in this study also demonstrate the potential benefits of such exploration. Even a relatively simple implementation such as that provided in the DCFD toolkit can lead to the discovery of useful combined features that contain new discriminating information. A more robust implementation that takes advantage of the suggested future work described in the previous section would increase the possibility of discovering such combined features that may lead to improved classification results and to potentially deeper insights into the nature of the classification problems being addressed.

# REFERENCES


[1] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Morgan Kaufmann, 2011.

[2] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[3] V. B. Bajić, "Comparing the success of different prediction software in sequence analysis: a review.," *Brief. Bioinform.*, vol. 1, no. 3, pp. 214–28, Sep. 2000.

[4] W. Mendenhall and T. Sincich, *Statistics for Engineering and the Sciences, Fourth Edition*. Prentice-Hall, 1995, p. 1182.

[5] W. H. Press, A. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007, p. 1235.

[6] A. Liekens, "Compute p-values for two-tailed T distribution test statistics in C." [Online]. Available: http://anthony.liekens.net/index.php/Computers/PValue.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 10, Nov. 2009.

[8] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995, pp. 338–345.

[9] R. R. Bouckaert, "Bayesian Network Classifiers in Weka." University of Waikato, Department of Computer Science, pp. 1–23, 2004.

[10] C.-C. Chang and C.-J. Lin, "LIBSVM," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[11] A. H. Lee and M. J. Silvapulle, "Ridge estimation in logistic regression," *Commun. Stat. - Simul. Comput.*, vol. 17, no. 4, pp. 1231–1257, Jan. 1988.

[12] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, Jan. 1991.

[13] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *Am. Stat.*, vol. 46, no. 3, pp. 175–185, Aug. 1992.

[14] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[15] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[16] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, “Rotation forest: A new classifier ensemble method.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, 2006.

[17] R. Kohavi, “The Power of Decision Tables,” in *Proceedings of the European Conference on Machine Learning*, 1995, pp. 174–189.

[18] B. Martin, “Instance-based learning: nearest neighbour with generalisation.” University of Waikato, Department of Computer Science, 01-May-1995.

[19] Y. Freund and L. Mason, “The Alternating Decision Tree Learning Algorithm,” in *ICML*, 1999, pp. 124–133.

[20] B. Pfahringer, G. Holmes, and R. Kirkby, “Optimizing the Induction of Alternating Decision Trees,” in *Proc 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2001, pp. 477–487.

[21] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993, p. 302.

[22] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[23] T. D. Hanson, “uthash: a hash table for C structures,” 2005. [Online]. Available: http://troydhanson.github.io/uthash/.

[24] OpenMP Architecture Review Board, “OpenMP Application Program Interface Version 3.1,” no. July. 2011.

[25] M. Kaur, A. Radovanovic, M. Essack, U. Schaefer, M. Maqungo, T. Kibler, S. Schmeier, A. Christoffels, K. Narasimhan, M. Choolani, and V. B. Bajic, “Database for exploration of functional context of genes implicated in ovarian cancer.,” *Nucleic Acids Res.*, vol. 37, no. Database issue, pp. D820–3, Jan. 2009.

[26] “Dragon Motif Finder .” [Online]. Available: http://www.cbrc.kaust.edu.sa/dmf/. [Accessed: 04-May-2014].

[27] B. Marchand, V. B. Bajic, and D. K. Kaushik, “Highly scalable ab initio genomic motif identification,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC ’11*, 2011, p. 1.

[28] B. E. Bernstein, E. Birney, I. Dunham, E. D. Green, C. Gunter, and M. Snyder, "An integrated encyclopedia of DNA elements in the human genome.," *Nature*, vol. 489, no. 7414, pp. 57–74, Sep. 2012.

[29] Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, and X. S. Liu, "Model-based analysis of ChIP-Seq (MACS).," *Genome Biol.*, vol. 9, no. 9, p. R137, Jan. 2008.

[30] "ChIPDragon." [Online]. Available: http://www.cbrc.kaust.edu.sa/chipdragon. [Accessed: 04-May-2014].