FSMEC: A Feature Selection Method based on the Minimum Spanning Tree and Evolutionary Computation

Amer Abu Zaher, Regina Berretta, Ahmed Shamsul Arefin, Pablo Moscato
The Priority Research Centre for Bioinformatics, Biomarker Discovery and Information-based Medicine
Information-based Medicine Program, Hunter Medical Research Institute
School of Electrical Engineering and Computer Science, The University of Newcastle, Australia

Amer.AbuZaher@uon.edu.au, {Regina.Berretta, Ahmed.Arefin, Pablo.Moscato}@newcastle.edu.au.

Abstract

In feature selection we aim at reducing the dimensionality of a dataset by excluding characteristics that do not compromise, and potentially enhance, the classification of a set of samples. We present a new type of supervised and multivariate feature selection approach that works by constructing proximity graphs in such a way that the number of edges connecting samples from different classes is minimised. We present this general idea using the Minimum Spanning Tree as a proximity graph and an Evolutionary Algorithm approach is used to search for a feature subset. We compare the performance of our algorithm against other feature selection methods, *(alpha,beta)-k*-Feature Set, and a ranking-based feature selection method, based on the use of CM1-scores. We employ two publicly available real-world datasets (one with training and test variants). The classification accuracies have been evaluated using a total of 49 methods from an open source data mining and machine learning package WEKA.

Keywords: Feature selection, evolutionary algorithm, proximity graph, minimum spanning tree.

1 Introduction

Feature selection is an essential task in data mining, and it is particularly relevant in bioinformatics where, in many cases, the number of features greatly exceed the number of samples (e.g., see (Dash & Liu, 1997; Guyon & Elisseeff, 2003; Liu & Yu, 2005; Saeys, Inza, & Larrañaga, 2007; Yang et al., 2012)). The core idea is to select the best subset of features that may potentially describe the whole dataset without losing important information, which may be useful for discrimination in classes of interest. Feature selection methods are generally classified as either: filter, wrapper or hybrid by the nature of its approach. The *filter* approach is simple, fast, and generally computational efficient however, does not consider the potential benefits given by learning algorithms, which may influence the selection of features that act synergistically (Dash & Liu, 1997; Kohavi & John, 1997; Yu & Liu, 2003). The filter approach consists of two types: univariate and multivariate. *Univariate* methods start by individually ranking the features and by assigning a score to each of the available features according to a pre-defined criterion (Dash & Liu, 1997). In this manner, generally the best-scored features are selected, while the others are discarded. However, this process does not consider the mutual information between features. When the top-scoring features are highly correlated in a given dataset it may be necessary to also consider other features for discriminating all pairs of samples. On the other hand, *multivariate* methods rank a group of features instead of individual features and decide which combination of a subgroup of features is the best. The *wrapper* approach combines, in a feedback loop, the selection process into search, learning and evaluation phases, employing a classifier to evaluate the selected subset of features. Therefore, it is computationally more expensive. At the same time, it is more accurate than other methods, even though it may overfit the training data which is an issue of concern for small datasets (Dash & Liu, 1997; Kohavi & John, 1997). The *hybrid* approach includes characteristics of both the filter and wrapper approaches, allowing for a feedback loop between the feature selection process and the learning algorithm.

The proposed method (FSMEC) is a supervised filtering multivariate approach. To understand its rationale we first note that we are given an $m \times n$ matrix of values in which each of the m rows represents a feature and each of the n columns a sample. In addition, for each of the n samples we have an associated class label. If k rows are selected, we can then calculate (using the resulting $k \times n$ submatrix) an $n \times n$ distance matrix between the samples. A coefficient in this matrix represents a distance between a pair of samples only considering the subset of selected features. Using this $n \times n$ matrix as input, a Minimum Spanning Tree (MST) can be found. The number of edges in the resulting MST that are connecting samples from different classes as well as the total number of selected features can then be used as a quality measure of the subset of features. This contribution explores some variants of interest involving these quality measures.

Evolutionary Computation (EC) is a technique (Back, Fogel, & Michalewicz, 1997; Fogel, 2006), that has been successfully used in a variety of fields including feature selection (ElAlami, 2009; Wu, Tang, Hor, & Wu, 2011). We propose its use for searching a subset of features that produces a MST with the best fitness value. We investigate in this contribution as set of fitness functions and details will be given later in the paper. The proposed FSMEC method is tested on two datasets and the results are compared with those of two other feature selection techniques. One method is based on the use of CM1 scores (Marsden, Budden, Craig, & Moscato, 2013) and

the other one based on the *(α,β)-k*-Feature Set problem (Berretta, Costa, & Moscato, 2008). To evaluate the performance of each feature selection algorithm, we used several classifiers available in the widely used WEKA software package (Witten, Frank, & Hall, 2011).

The structure of this paper is as follows. Section 2 explains how we use the MST to evaluate the quality of a subset of features. In Section 3 the proposed evolutionary algorithm is described in details. Section 4 presents the results. Finally, Section 5 concludes this paper.

2 The MST and Feature Selection

The input of the problem is a matrix H_{mxn} and an array C_n, where m denotes the number of features, n denotes the number of samples, h_{ij} holds the value of feature i in sample j and c_j holds the class of the sample j. Next, giving k selected features, a distance matrix D_{nxn} is calculated, where d_{ij} is the distance between samples i and j. Note that each subset of features will, in general, lead to the computation of a different distance matrix.

A complete, undirected and weighted graph $G(V,E,W)$ is used to represent the matrix H_{kxn} (considering only the k features), where the nodes in V represent each sample ($|V|= n$), E is a set of edges reflecting the relationship between samples, and W is a set of edge weights with $w_{ij} = d_{ij}$. Given the graph G described above, a sub-graph of G named *spanning tree* is one that links all the nodes together with $(n-1)$ edges. There are several possible spanning trees for a graph G. The MST is a spanning tree with the lowest total sum of weights of its $(n-1)$ edges (Graham & Hell, 1985), which we will denote is as $G_{MST}(V,E_{MST},W_{MST})$. The search for the MST is a combinatorial optimization problem and there are different efficient algorithms to solve this problem. Two well-known algorithms are Kruskal's (Kruskal, 1956) and Prim's (Prim, 1957). In this paper, Prim's algorithm is adopted due to the density of the graph and the minimal usage of memory (Graham & Hell, 1985). The quality of the subset of k features is evaluated based on its size, the number of edges connecting nodes (samples) from different classes (denoted by e) in the built MST, and a score calculated for the constructed MST (described later). An Evolutionary Algorithm, described in the next section, is applied for searching the best subset of features. The merit of different fitness functions is also evaluated.

3 FSMEC-Feature Selection Based on MST using Evolutionary Computation

Evolutionary Computation EC is used as a search strategy for finding a subset of k features. The evolutionary algorithm (EA) used in our approach is presented in Algorithm 1.

First, a population of p individuals is created by *initialisePop()*. An individual represents a solution to the problem, which in our case is a subset of features. It is represented by an array S of size m, where $s_i=1$ if feature i is selected and 0 if otherwise.

At each generation, the algorithm applies the following operators: parent selection (*selectParents (parent1, parent2)*), crossover (*crossover (parent1, parent2)*), and mutation (*mutation(offspring)*). We apply an update procedure *updatePop(P)* that replaces the worst individual with a new one if the latter has a better fitness than the former.

Algorithm 1: Pseudo-code of the Evolutionary Algorithm implemented for the FSMEC.

```
n: number of samples; m: number of features
p: number of individuals in the population.
P: population of individuals and their fitness values.
1  P ← initialisePop()
2  REPEAT
3      selectParents(parent1,parent2)
4      offspring ← crossover(parent1,parent2)
5      offspring ← mutation(offspring)
6      fitness(offspring)
7      population ← updatePop(P)
8  UNTIL max_numberOfGenerations or
numberOfGenerationWithoutImprovement
```

3.1 Population Initialisation

The population is structured as a list P of objects with each object containing an individual and its fitness value. The *initialisePop()* initialises the population of 40 individuals ordered by their fitness values. At each generation, a new individual is generated and may replace the worst individual in the population according to *updatePop(P)*. Since the population is ordered, a binary search procedure is used to locate the new individual in the population. It is worth mentioning that the time complexity of this process, $O(log(|S|))$, is significantly better than re-sorting the population for each new generation which would cost $O(|S|log(|S|))$. Two population initialisation strategies have been implemented in the EA:

Population initialisation strategy 1 - *5Bins*. In this strategy we divide P into five equal bins. Each bin b, $1 \le b \le 5$, for each individual, (b*10)% of features are randomly selected. For example, for b=1, 10% of features are randomly selected in each individual, for b=2, 20% of features are selected, and so forth.

Population initialisation strategy 2 (*5Bins-CM1*). It is similar to strategy 1 (5Bins), but uses the CM1 score to influence the probability of a feature to be chosen. Initially, we compute the CM1 score (described in Section 4) for each feature and normalise the CM1 values (CM1_norm). Similarly to strategy 1 (5bins), P is divided in 5 bins. In the first bin, 10% of features are chosen, but features with CM1_norm greater than 0.5 will have 90% chance to be chosen, while the features with CM1_norm score less than 0.5 will have 10% of probability to be chosen. In the second bin, 20% of features are chosen, but features with CM1_norm greater than 0.5 will have 80% chance to be chosen, while the features with CM1_norm score less than 0.5 will have 20% chance to be chosen. For the bins 3, 4 and 5, an equivalent strategy is employed.

3.2 Optimisation criteria - Fitness functions

In this work, we are considering only two classes. We have four fitness functions to deal with them:

Mine. Minimising the number of edges, *e*, connecting samples from different classes in the MST. (Min *e*)

MineMink. Minimising the summation of the normalised *e* (i.e. the number of inter-class edges in the MST connecting samples from different classes divide by the total number of edges in the MST) and normalised *k* (the number of selected features).

$$Min \frac{e}{n-1} + \frac{k}{m}$$

For the next two we need to define a score for the built MST (*MSTscore*). Consider a MST as a graph $G_{MST}(V,E_{MST},W_{MST})$. The set of nodes V has a bipartition in V_A and V_B; V_A are the set of samples that belong to class A and V_B are the set of samples that belong to class B. Using the weights of the edges that connect nodes from the same class, we can define a score for the MST given a partition of its set of vertices as follows:

$$MSTscore(A,B) = \frac{\frac{1}{|V_A|}\sum_{\substack{i\in V_A \\ j\in V_A}} w_{ij} - \frac{1}{|V_B|}\sum_{\substack{i\in V_B \\ j\in V_B}} w_{ij}}{1 + max_{\substack{i\in V_B \\ j\in V_B}}\{w_{ij}\} - min_{\substack{i\in V_B \\ j\in V_B}}\{w_{ij}\}}$$

Note that in general *MSTscore(A,B)* is different than *MSTscore(B,A)*. We can then define the other two functions:

MineMaxMSTscore: In this case the objective is to minimize the ratio between the number of edges, *e*, connecting samples from different classes in the MST and the score of the *MSTscore(A,B)*

$$Min \frac{e}{MSTscore(A,B)}$$

MineMinkMaxMSTscore: In this case the objective is to minimize the summation of normalised *e* and normalised *k*; divided by the *MSTscore(A,B)*

$$Min \frac{\frac{e}{n-1} + \frac{k}{m}}{MSTscore(A,B)}$$

3.3 Breeding

We refer to the process in which two "parent" solutions are selected and a new solution is created. It consists of four operations: *selecting* parent solutions from the population; creating a new offspring solution from the selected parents using *crossover* and *mutation* operations and (in case the new offspring has a better fitness function than the worst individual in the population) *replacing* the worst individual in the population with the new offspring. Each operation is described below.

Selection: selects two solutions as parents: the solution with the best fitness value in the population and another selected uniformly at random.

Crossover: a *uniform* crossover with a crossover rate of 40% is used.

Mutation: two mutation strategies have been applied in the EA and both are *1-flip mutation* with 5% mutation rate. In strategy 1 – **Mutation1** – 5% of each of the individual solutions are randomly mutated. In the second mutation strategy – **Mutation-CM1** – normalised scores (as used in the 5Bins-CM1 population initialisation) are used in the following way. If the feature is selected and the CM1_norm is less than 0.5, then the feature is discarded. If the feature is not selected and the CM1_norm is greater than 0.5, then it is selected.

Replacement. Finally, the mutated offspring is tested in the replacement strategy. If its fitness is better than the worst individual, then it is included in the population and the worst individual is discarded. A binary search is performed to locate the position of the offspring in the population. Two stopping criteria have been used in the FSMEC algorithm: the EA will stop after a predetermined *maximum number of generations* (10000) or the best individual is *unchanged for a fix number of generations* (1000).

4 Computational Results

For the purpose of evaluating the performance of our FSMEC algorithm, two datasets are used to carry out the computational experiments, which are described next. Our algorithm was coded in Python 2.7 and executed under Unix operating system in a machine with Dual Xeon 2.67 GHz, 8 cores and 32 GB RAM.

4.1 Datasets

The properties of the two datasets used are summarised in (Table 1) and described next.

Dataset Name		features	samples	Classes	
Shakespearean era plays and poems (Craig & Whipp, 2010)		220	256	Plays	202
				Poems	54
Alzheimer's disease (Ray et al., 2007)	Training	120	83	AD	43
				NDC	40
	Test	120	81	AD	42
				NDC	39

Table 1. Datasets used for evaluating the FSMEC algorithm.

Shakespearean era plays and poems dataset. This dataset contains 256 works of the Shakespearean era and they belong to two classes: plays (202) and poems (54) as samples and 220 *functional words* as features. The "frequency of use" of these 220 words have been extracted from a cohort of 66907 words previously analysed by Arefin et al. in (Arefin, Vimieiro, Riveros, Craig, & Moscato, 2014). The goal is then to identify 'a subset of functional words' that is able to group the texts into the two classes; plays and poems.

Alzheimer's Disease dataset. It consists of two sub-datasets: the training and test datasets from Ray et al. (Ray et al., 2007). The *training* dataset contains the relative abundances 120 proteins (z-scores, which will be used as features) measured on 83 people who have been classified into two classes: 43 Alzheimer's Disease (AD) samples and 40 Non Demented Control (NDC) samples. The *test* dataset contains 120 proteins and 81 patients classified into two classes - 42 AD samples and 39 NDC samples. The test dataset also contains 11 samples labelled as 'Other Dementia' OD samples, which were excluded from the analysis.

Different methods have been introduced to find an optimum molecular test for an earlier diagnosis of

Alzheimer's disease (Berretta et al., 2008; Ravetti & Moscato, 2008; Ray et al., 2007). They define the signatures that are able to distinguish between NDC and AD samples and hence predict the AD samples that already have a Mild Cognitive Impairment. In the same manner, the FSMEC is used to find a subset from the 120 proteins that can separate the AD patients from NDC ones.

4.2 Evolutionary Algorithm Analysis

The initial tests were conducted to evaluate and setup an initial configuration for the EA. All the results were analysed using the Wilcoxon test. We tested different crossovers operators, mutation rates, population sizes and stop criteria. Our preliminary tests indicate that the best results are achieved when we apply the uniform crossover (crossover rate 0.4), a 1-flip mutation (mutation rate 0.05), a population size of 40, and when the number of generations without improvements is equal to 1000.

EA	Population Strategy	Mutation Strategy
EA1	5Bins	Mutation1
EA2	5Bins-CM1	Mutation-CM1

Table 2. The configuration of the two Evolutionary Algorithms tested (according to population initialisation and mutation strategies).

Fitness function	EA	*k*	*e*	*Fit*	*Time*	*Gen*
Mine	EA1	64	2	**2.0**	333	1599
	EA2	101	3.2	3.2	303	1686
MineMink	EA1	2	96	**0.01**	320	1702
	EA2	5	40	0.02	422	1616
MineMaxMSTscore	EA1	112	2.8	**9.5**	1511	5847
	EA2	59.6	4	13.7	2363	9143
MineMinkMaxMSTscore	EA1	107	3	**0.036**	1541	5110
	EA2	60.8	3.6	0.052	2586	10000

Table 3. Average number of selected features *k*, inter-class edges *e*, fitness *Fit*, running *time*, and generations *gen* obtained by EA1 and EA2 using the Shakespeare era plays and poems dataset for each optimisation criteria.

Fitness function	EA	*k*	*e*	*Fit*	*Time*	*Gen*
Mine	EA1	43	5.9	5.9	18.3	1683
	EA2	53	5.8	**5.8**	19.5	1950
MineMink	EA1	53	5	**0.068**	25.6	2114
	EA2	35	7	0.086	24.2	1647
MineMaxMSTscore	EA1	55.3	7.6	44.3	103.1	5934
	EA2	31.4	6.5	**35.0**	115.3	7592
MineMinkMaxMSTscore	EA1	52.2	7.1	**0.98**	76.2	4385
	EA2	17.6	9.6	1.20	92.9	6587

Table 4: Average number of selected features *k*, inter-class edges *e*, fitness *Fit*, running *time*, and generation *gen* obtained by EA1 and EA2 using the Alzheimer disease training dataset for each optimisation criteria.

In the next set of experiments we tested the two different population initialisation strategies (*5Bins* and *5Bins-CM1*) and the two mutation strategies (*Mutation1* and *Mutation-CM1*) as depicted in Table 2. Tables 3 and 4 show the results obtained from the Shakespearean era plays and poems and Alzheimer's disease training datasets, respectively, for each optimisation criteria. Each row in these tables is the average of 10 executions. These tables highlight the best EA for each optimisation criteria. For the Shakespearean era plays and poems dataset (Table 3), the EA1 achieved the best average fitness score (*Fit*) for the four optimisation criteria. For the Alzheimer's disease training dataset, the EA1 was the best for two optimisation criteria (*MineMink* and *MineMinkMaxMSTscore*) and EA2 performed better for the other two optimisation criteria. EA1 was superior for Shakespearean era plays and poems dataset, and for the Alzheimer's disease training dataset there was a tie between EA1 and EA2.

4.3 Classification Performance

The next computational test aimed to evaluate the practical use of the set of features obtained by our FSMEC for a learning algorithm.

Type	Classifier	Type	Classifier
Bayes	BayesNet	Meta	RandomSub_Space
Bayes	NaiveBayes	Meta	RotationForest
Bayes	NaiveBayesUpdatable	Meta	ThresholdSelector
Function	Logistic	Mesc	HyperPipes
Function	SimpleLogistic	Mesc	VFI
Function	RBFNetwork	Rules	ConjunctiveRule
Function	SMO	Rules	DecisionTable
Function	SPegasos	Rules	Jrib
Function	VotedPerceptron	Rules	NNge
Lazy	IB1	Rules	OneR
Lazy	Kstar	Rules	Part
Lazy	LWL	Rules	Ridor
Meta	AdaBoost	Tree	ADTree
Meta	AttributeSelectedClassifier	Tree	BFTree
Meta	Bagging	Tree	FT
Meta	ClassificationViaRegression	Tree	LADTree
Meta	Dagging	Tree	LMT
Meta	Decorate	Tree	DecisionStump
Meta	END	Tree	J48
Meta	FilteredClassifier	Tree	J48graft
Meta	LogitBoost	Tree	RepTree
Meta	MultiBoostAB	Tree	NBTree
Meta	MultiClassClassifier	Tree	Random_Forest
Meta	OrdinalClass	Tree	RandomTree
Meta	RandomCommittee		

Table 5: List of classifiers associated with their types as categorised in WEKA (Witten et al., 2011) version 3.6.4.

For each row in the tables 3 and 4, the subset of features (out of 10 solutions) that has the best fitness value (*Fit*) was selected to be evaluated by a learning algorithm. We have used 49 machine learning algorithms from the well-known WEKA software package (Witten et al., 2011). Table 5 lists all of the classifiers considered, along with their respective types as categorised in WEKA (version 3.6.4). In each case, the average specificity, sensitivity,

classification accuracy, and the Matthews' correlation coefficient (MCC) have been calculated using 10-fold cross validation, which means that the original dataset is randomly divided into 10 equal subsets: 9 are used as training sets, and the remaining subsets are used as test sets. Evaluation is repeated 10 times, such that each subset is utilised exactly once for this purpose (Witten et al., 2011). The results are shown in Tables 6 and 7. Each row in these tables shows the evolutionary algorithm applied (EA1 or EA2), the size of the subset of features (k), the number of inter-class edges (e), and the fitness value (*Fit*) for the specific optimisation criterion. It also shows the average classification accuracy and the average MCC of 49 classifiers.

Optimisation Criteria	EA	*k*	*e*	*Fit*	ACC	MCC
Mine	EA1	84	1	1	**0.959**	**0.876**
	EA2	89	1	1	0.952	0.855
MineMink	EA1	94	1	0.018	**0.957**	**0.869**
	EA2	33	4	0.016	0.955	0.866
MineMaxMSTscore	EA1	117	2	6.7	0.954	0.861
	EA2	50	3	6.3	**0.959**	**0.876**
MineMinkMaxMSTscore	EA1	110	2	0.027	**0.960**	**0.880**
	EA2	52	1	0.028	0.956	0.868

Table 6: Average accuracy and Matthews' correlation coefficient (MCC) using the best subset of features from FSMEC for the Shakespeare era plays and poems dataset. Each row shows number of features (*k*), the number of inter-class edges (*e*), the fitness value (*Fit*), the average accuracy and MCC.

Optimisation Criteria	EA	*k*	*E*	*Fit*	ACC	MCC
Mine	EA1	56	5	5	0.859	0.720
	EA2	41	5	5	**0.861**	**0.725**
MineMink	EA1	4	57	0.045	0.860	0.721
	EA2	6	35	0.073	**0.867**	**0.737**
MineMaxMSTscore	EA1	55	6	35.4	0.862	0.725
	EA2	27	5	29.8	**0.875**	**0.752**
MineMinkMaxMSTscore	EA1	42	7	0.888	0.873	0.748
	EA2	15	8	0.795	**0.880**	**0.762**

Table 7. Average accuracy and Matthews' correlation coefficient (MCC) using the best subset of features from FSMEC for the Alzheimer's disease dataset. Each row shows number of features (*k*), the number of inter-class edges (*e*), the fitness value (*Fit*), the average accuracy ACC and MCC.

Table 6 illustrates the results for the Shakespearean era plays and poems dataset and Table 7 for the Alzheimer's disease training dataset. The FSMEC has been applied on the Alzheimer disease *training* dataset to find subsets of features, which in turn have been used to build a classification model over the Alzheimer disease *test* dataset. For each dataset and optimisation criterion, the best classification accuracy and MCC results are highlighted. In the previous experiment, EA1 performed better for the Shakespearean era plays and poems dataset (see Table 3). When we evaluate the classification performance, the results were similar, showing that EA1 obtained better results for 3 out of 4 optimisations criteria (see Table 6). In the case of the Alzheimer's disease dataset (see Table 7) the results showed a better performance of EA2. After analysing the different optimisation criteria, we found that the *MineMinkMaxMSTscore* obtained slightly better classification performance for both datasets, independently of the EA applied.

4.4 Benchmark Techniques

To examine the performance of the proposed method (FSMEC), two feature selection methods ((α,β)-k-Feature set and CM1 score) are used as benchmark techniques. The *(α,β)-k-Feature Set* is a supervised, multivariate filter method based on combinatorial optimization first proposed by Cotta et al. (Cotta, Sloper, & Moscato, 2004) and then used by many other applications (Berretta et al., 2008; Berretta, Mendes, & Moscato, 2005, 2007; de Paula, Ravetti, Berretta, & Moscato, 2011; Hourani, Berretta, Mendes, & Moscato, 2008; Ravetti, Rosso, Berretta, & Moscato, 2010). The task is to identify k features such that at least α features of these k can explain the dichotomy between samples that belong to different classes.In addition, those k features should satisfy that at least β features must explain the similarities between samples from the same class. A mathematical programming software called CPLEX has been used to obtain solutions using integer programming techniques (Berretta et al., 2008). For further details about the (α,β)-k-Feature Set problem we refer to (Berretta et al., 2008).

In contrast, the *CM1* score (Marsden et al., 2013) is a supervised, univariate filter method. It works by individually ranking the features according to their expression values in order to identify features presenting differentiation between samples from a target class and samples from the outclass. Consider V_A and V_B are a partition of samples (V) in the dataset of interest (i.e. $V_A \cup V_B = V$ and $V_A \cap V_A = \emptyset$), such that V_A is a set of all samples that belong to one class and V_B is a set of all samples that are not labelled as class A. . The CM1 score for a feature k can then be defined as

$$CM1_k = \frac{\frac{1}{|A|}\sum_{i \in A} h_{ki} - \frac{1}{|B|}\sum_{i \in B} h_{ki}}{1 + \max_{i \in V_B}\{h_{ki}\} - \min_{i \in V_B}\{h_{ki}\}}$$

where h_{kxi}, as described before, holds the value of the feature k in the sample i.

Table 8 summarises the size of the features' sets obtained by the benchmark techniques and the FSMEC. For CM1, we are following the same approach of [13], we select the 20 highest and 20 lowest CM1 markers for both words in the Shakespeare era plays and poems dataset and features in the Alzheimer's disease dataset.

Table 9 summarises the average MCC obtained by the benchmark techniques and the FSMEC. It shows the average MCC results achieved using all (*ALL*) the features *(ALL),* the benchmark methods ((α,β)-k-Feature set and CM1) and the results obtained by the four FSMEC's optimisation criteria.

	Dataset							
	Shakespeare				Alzheimer's			
All	220				120			
(α,β)-*k*-FEATURE SET	140				10			
CM1	40				40			
	EA1		EA2		EA1		EA2	
	k	*e*	*k*	*e*	*k*	*e*	*K*	*e*
Mine	84	1	89	1	56	5	41	5
MineMink	94	1	33	4	57	4	35	6
MineMaxMSTscore	117	2	50	3	55	6	27	5
MineMinkMaxMSTscore	110	2	52	1	42	7	15	8

Table 8. The size of resulting features' subsets (*k*) obtained by the benchmark techniques and the number of inter-class edges *e* and the value of *k* for the FSMEC for each optimisation criteria and considering the EA1 and EA2.

	Dataset							
	Shakespeare				Alzheimer's			
	ACC		MCC		ACC		MCC	
All	0.949		0.849		0.848		0.698	
(α,β)-*k*-FEATURE SET	0.952		0.856		**0.891**		**0.784**	
CM1	0.941		0.824		0.860		0.722	
	EA1		EA2		EA1		EA2	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Mine	0.959	0.876	0.952	0.855	0.859	0.720	0.861	0.725
MineMink	0.957	0.869	0.955	0.866	0.86	0.721	0.867	0.737
MineMaxMSTscore	0.954	0.861	0.959	0.876	0.862	0.725	0.875	0.752
MineMinkMaxMSTscore	**0.960**	**0.880**	0.956	0.868	0.873	0.748	0.880	0.762

Table 9. The best average MCC and ACC (accuracy) results achieved from the FSMEC' four optimisation criteria and the two benchmark methods for the dataset under study. The (All) means all features *m* without applying a feature selection.

It also shows that most of FSMEC's optimisation criteria demonstrated their superiority in terms of the MCC and the accuracy over the other methods in case of the Shakespearean era plays and poems dataset. If we compare our four FSMEC's optimisation criteria, the *MineMinkMaxMSTscore* achieved the best MCC results with 0.880 and the best accuracy with 96.0% compared with *Mine*, *MineMink* and *MineMaxMSTscore*. Notably, the value of *e* is 2 in case of the best *MineMinkMaxMSTscore* while the value of *e* is 1 in case of both *Min e* and the *MineMink* with both providing very close MCC values.

In case of the Alzheimer's disease training dataset, the (α,β)-*k*-Feature Set provided the best average MCC result (0.784), however our proposed method is highly competitive with *MineMinkMaxMSTscore* attaining a MCC of 0.762. Additionally, the FSMEC obtained better results than CM1 and using all features (*ALL*).

Notably, the *MineMinkMaxMSTscore* in case of both the Shakespeare era plays and poems dataset and Alzheimer's disease dataset using EA1 or EA2 always obtained better classification performance compared with other optimisation criteria. However, there is no clear winner between EA1 and EA2.

Next, we selected the five best performing WEKA classifiers in each experiment from Tables 8 and 9. These results are organised in Tables 10-13. Tables 10 and 11 show the results achieved for the Shakespeare era plays and poems dataset using EA1 (Table 10) and EA2 (Table 11). Tables 12 and 13 show the results for the Alzheimer's disease dataset using EA1 and EA2, respectively. Each table shows the number of features (*k*), ACC and MCC achieved by each classifier. We also show the average and median of the results for each method in the last two rows. Note that the list of classifier methods in each table is the union of the five best performing methods in each experiment. In each row of each table we highlighted the best result(s).

Analysing the results in Tables 10 and 11 we note that the *MineMinkMaxMSTscore* optimisation criterion continues to lead the results by achieving 14 best results (6 using EA1 and 8 using EA2). It also achieves the best average and median results. The *MineMaxMSTscore* optimisation criterion also obtained good results, achieving 8 best results (3 for EA1 and 5 for EA2). Figure 1 shows the MST that has been constructed from the selected features and using the *MineMinkMaxMSTscore* optimisation criterion for Shakespeare era plays and poems dataset, using EA2, with 52 features. In the case of the Alzheimer's disease dataset (see Table 12 and 13), the *MineMinkMaxMSTscore* optimisation criterion also achieves excellent results. When we compare using median, the best results are achieved by *MineMinkMaxMSTscore* and (α,β)-*k*-Feature Set method.

In the next section, we show the effect of the different optimisation criteria in the classification performance.

4.5 Effect of the number of fitness functions on the MCC

An empirical study has been made in order to investigate the effect of the four optimisation criteria on the classification performance. We run EA2 five times for each optimisation criterion using Alzheimer's disease training and the Shakespearean era plays and poems datasets. Each 10 generations of the EA2, we saved the best individual of the population in a set. Then, for each solution in this set, we used the same 49 classification algorithms and calculated the average of MCC values.

The results of this experiment are reported in Figures 2 and 3. Figure 2 illustrates the results for the Shakespearean era plays and poems dataset while Figure 3 for Alzheimer's disease dataset, with the performance evaluated in terms of the MCC. The horizontal line (*x*-axis) shows the fitness value for each solution during EA2 execution every 10 generations, while the average MCC values for each of the corresponding solution are shown in the vertical line (*y*-axis).

The majority of the results indicate that minimising any one of the fitness functions under study generally results in maximising the average MCC.

5 Conclusion

This work presents a new method based on the Minimum Spanning Tree to find a solution to the feature selection problem. Accordingly, four fitness functions (based on this criterion) have been tested: *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore*. An evolutionary algorithm (EA) has been used to address the combinatorial optimisation problem. Two sorts of experiments have been made on two real life datasets in order to select the best performing EAs (parameters, operators, and fitness functions). The first test is used to tune the population size, maximum number of generations, mutation rate, and crossover operator. The results of this experiment have been analysed by the Wilcoxon test and accordingly the best performing operators and parameters were selected. In the second test, two EAs have been implemented to investigate whether the CM1 score can improve the solutions evolved by the EA (i.e. EA2) or not (i.e. EA1). First, the results were varied (i.e. EA1 performed better for the Shakespearean era plays and poems dataset while the EA2 attained better results for the Alzheimer's disease).

Next, we used 49 machine learning algorithms from the WEKA software package to evaluate the set of features obtained by our FSMEC. Moreover, we selected the best five performing classifiers for each of the methods to better analyse the results. The results show that the proposed method can be successfully used to reduce the number of features and increase the classification performance. In other words, the FSMEC has produced improved classification performance, when compared against all the features before applying our method. We have also compared our method with two state of the art feature selection methods on two real world datasets.

In case of the Shakespeare era plays and poems dataset the FSMEC' four optimisation criteria were managed to outperform the others in terms of MCC using 49 Weka classifiers and even though using the best five performing classifiers. Our method did not attain the best MCC results in case of the Alzheimer's disease using 49 Weka classifiers but using the best five performing classifiers our Mine*M*in*k*Max*MSTvalue* achieved the best results for both datasets. Finally, an investigation has been made to evaluate the effect of fitness function on the MCC values. Most of the results in this investigation drew an upward trend-line to increase the MCC values when the fitness value minimises.

In the future, two-way improvement will be considered. In the first direction, we will continue improving the EA by employing Memetic Algorithms (Moscato et al., 1989). In addition, we will test our method using different proximity graphs.

6 References

Arefin, A. S., Vimieiro, R., Riveros, C., Craig, H., & Moscato, P. (2014). An Information Theoretic Clustering Approach for Unveiling Authorship Affinities in Shakespearean Era Plays and Poems. *PLoS ONE, 9*(10), e111445. doi: 10.1371/journal.pone.0111445

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*: IOP Publishing Ltd.

Berretta, R., Costa, W., & Moscato, P. (2008). Combinatorial optimization models for finding genetic signatures from gene expression datasets *Bioinformatics* (pp. 363-377): Springer.

Berretta, R., Mendes, A., & Moscato, P. (2005). *Integer programming models and algorithms for molecular classification of cancer from microarray data*. Paper presented at the Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38.

Berretta, R., Mendes, A., & Moscato, P. (2007). Selection of discriminative genes in microarray experiments using mathematical programming. *Journal of Research and Practice in Information Technology, 39*(4), 287-299.

Cotta, C., Sloper, C., & Moscato, P. (2004). Evolutionary search of thresholds for robust feature set selection: application to the analysis of microarray data *Applications of Evolutionary Computing* (pp. 21-30): Springer.

Craig, H., & Whipp, R. (2010). Old spellings, new methods: automated procedures for indeterminate linguistic data. *Literary and Linguistic Computing, 25*(1), 37-52.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent data analysis, 1*(3), 131-156.

de Paula, M. R., Ravetti, M. G., Berretta, R., & Moscato, P. (2011). Differences in abundances of cell-signalling proteins in blood reveal novel biomarkers for early detection of clinical Alzheimer's disease. *PloS one, 6*(3), e17481.

ElAlami, M. E. (2009). A filter model for feature subset selection based on genetic algorithm. *Knowledge-Based Systems, 22*(5), 356-362.

Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence* (Vol. 1): John Wiley & Sons.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing, 7*(1), 43-57.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157-1182.

Hourani, M. a., Berretta, R., Mendes, A., & Moscato, P. (2008). Genetic signatures for a rodent model of Parkinson's disease using combinatorial optimization methods *Bioinformatics* (pp. 379-392): Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1–2), 273-324. doi: http://dx.doi.org/10.1016/S0004-3702(97)00043-X

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem.

Proceedings of the American Mathematical society, 7(1), 48-50.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on, 17*(4), 491-502.

Marsden, J., Budden, D., Craig, H., & Moscato, P. (2013). Language Individuation and Marker Words: Shakespeare and His Maxwell's Demon. *PloS one, 8*(6), e66813.

Moscato et al., P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report, 826*, 1989.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell system technical journal, 36*(6), 1389-1401.

Ravetti, M. G., & Moscato, P. (2008). Identification of a 5-protein biomarker molecular signature for predicting Alzheimer's disease. *PloS one, 3*(9), e3111.

Ravetti, M. G., Rosso, O. A., Berretta, R., & Moscato, P. (2010). Uncovering molecular biomarkers that correlate cognitive decline with the changes of hippocampus' gene expression profiles in Alzheimer's disease. *PloS one, 5*(4), e10153.

Ray, S., Britschgi, M., Herbert, C., Takeda-Uchimura, Y., Boxer, A., Blennow, K., . . . Karydas, A. (2007). Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins. *Nature medicine, 13*(11), 1359-1362.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics, 23*(19), 2507-2517.

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques.

Wu, Y.-L., Tang, C.-Y., Hor, M.-K., & Wu, P.-F. (2011). Feature selection using genetic algorithm and cluster validation. *Expert Systems with Applications, 38*(3), 2727-2732.

Yang, S., Yuan, L., Lai, Y.-C., Shen, X., Wonka, P., & Ye, J. (2012). *Feature grouping and selection over an undirected graph.* Paper presented at the Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.

Yu, L., & Liu, H. (2003). *Feature selection for high-dimensional data: A fast correlation-based filter solution.* Paper presented at the ICML.

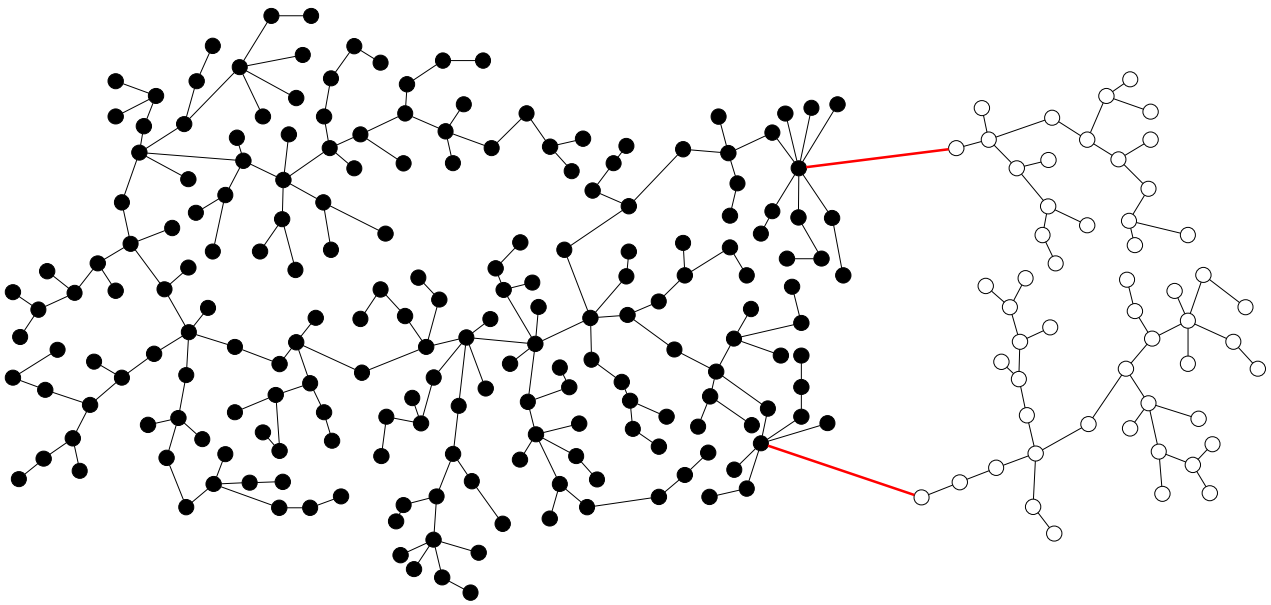

Figure 1. The MST constructed from our *MineMinkMaxMSTscore* optimisation criterion for the Shakespearean dataset using EA2. The size of resulted features is 52. The nodes represent the works, which are classified into plays (202 nodes in black) and poems (54 nodes in white). The red edges show the inter-class edges.

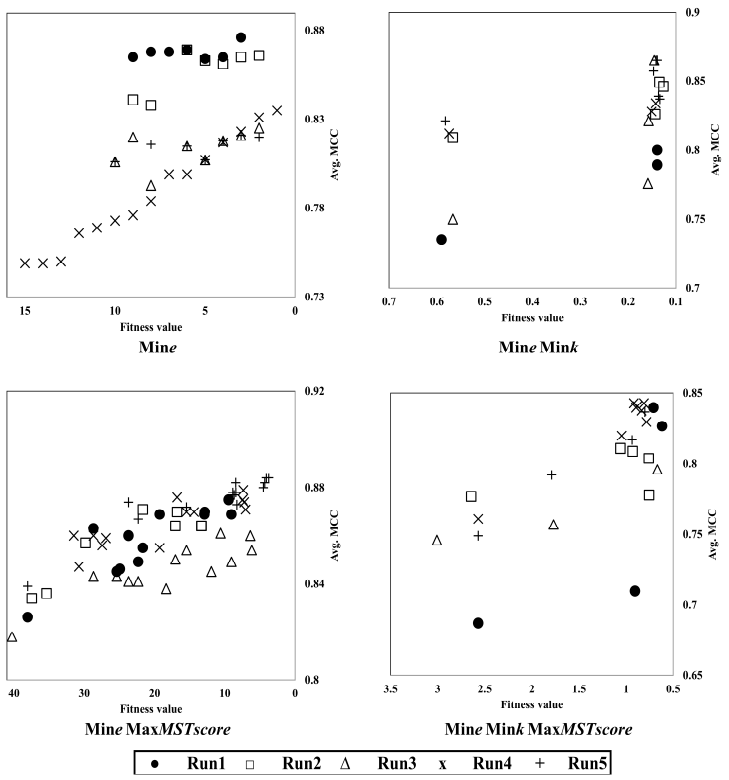

Figure 2. Effect of the Fitness functions (in x-axis): *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore* optimisation criteria on the average MCC value (in y-axis) in case of the Shakespearean era plays and poems dataset.

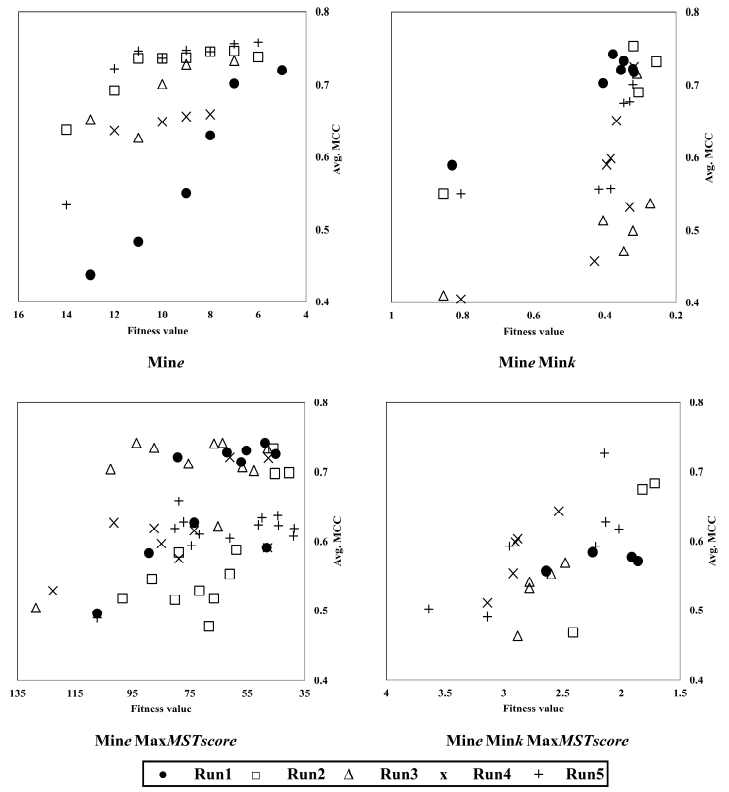

Figure 3. Effect of the Fitness functions (in x-axis): *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore* optimisation criteria on the average MCC value (in y-axis) in case of the Alzheimer's disease dataset.

	ALL		**(α,β)-*k*-FEATURE SET**		**CM1**		***Mine***		***MineMink***		***MineMax MSTscore***		***MineMinkMax MSTscore***	
EA							EA1		EA1		EA1		EA1	
k	220		140		40		84		44		117		110	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	0.977	0.934	0.934	0.977	0.957	0.883	0.973	0.922	**0.984**	**0.955**	0.969	0.915	**0.984**	**0.955**
Simple_Logistic	0.984	0.955	0.905	0.969	0.957	0.872	**1.000**	**1.000**	0.965	0.892	0.984	0.953	0.988	0.965
RBF_Network	0.977	0.934	**0.945**	**0.980**	0.969	0.910	0.977	0.932	0.977	0.934	0.973	0.924	0.980	0.945
SMO	**0.996**	**0.988**	0.977	0.992	0.980	0.943	0.977	0.930	0.984	0.953	0.992	0.977	0.992	0.976
S_Pegasos	0.965	0.892	0.905	0.969	0.953	0.856	0.977	0.930	0.988	0.965	0.969	0.905	**0.996**	**0.988**
IBK	0.945	0.832	0.856	0.953	0.973	0.917	0.973	0.917	**0.977**	**0.929**	0.957	0.868	0.961	0.880
LWL	0.930	0.792	0.815	0.938	0.926	0.779	0.930	0.795	0.922	0.762	0.937	0.818	**0.941**	**0.828**
Dagging	0.977	0.929	**0.942**	**0.980**	0.957	0.869	0.980	0.941	0.969	0.905	0.980	0.941	0.977	0.929
Decorate	0.961	0.883	0.954	0.984	0.965	0.895	0.973	0.918	**0.996**	**0.988**	0.984	0.954	0.977	0.930
Random_Committee	0.980	0.945	0.918	0.973	0.937	0.805	0.969	0.906	**0.988**	**0.965**	0.980	0.941	0.977	0.930
OneR	0.910	0.722	0.749	0.918	0.910	0.722	0.918	0.749	0.926	0.771	0.926	0.771	**0.937**	**0.815**
FT	0.977	0.930	0.918	0.973	0.961	0.881	**0.988**	**0.965**	0.965	0.892	0.973	0.917	**0.988**	**0.965**
LMT	0.984	0.955	0.905	0.969	0.957	0.872	**1.000**	**1.000**	0.965	0.892	0.984	0.953	0.988	0.965
NBTree	0.988	0.965	**0.977**	**0.992**	0.961	0.884	0.984	0.954	0.980	0.941	0.961	0.883	0.969	0.908
Avearge	0.968	0.904	0.907	0.969	0.955	0.863	0.973	0.919	0.970	0.910	0.969	0.909	**0.975**	**0.927**
Median	**0.977**	**0.930**	0.918	0.973	0.957	0.872	**0.977**	**0.930**	**0.977**	0.929	0.973	0.924	**0.977**	**0.930**

Table 10. Performance results for the top five WEKA models for *ALL* (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxkMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy and MCC achieved for Shakespeare era plays and poems dataset, using EA1.

	ALL		**(α,β)-*k*-FEATURE SET**		**CM1**		***Mine***		***MineMink***		***MineMax MSTscore***		***MineMinkMax MSTscore***	
EA							EA2		EA2		EA2		EA2	
k	220		140		40		89		53		50		52	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	**0.977**	**0.934**	0.977	0.934	0.957	0.883	0.973	0.924	0.973	0.920	**0.977**	**0.934**	0.973	0.924
Naive_Bayes	0.977	0.932	0.977	0.932	0.945	0.860	0.977	0.932	0.969	0.908	**0.988**	**0.966**	0.977	0.932
Simple_Logistic	0.984	0.955	0.969	0.905	0.957	0.872	0.957	0.869	0.953	0.859	0.977	0.929	**0.992**	**0.977**
RBF_Network	0.977	0.934	0.980	0.945	0.969	0.910	0.980	0.945	0.973	0.920	**0.988**	**0.966**	0.973	0.922
S_Pegasos	0.965	0.892	0.969	0.905	0.953	0.856	0.973	0.917	0.941	0.821	0.984	0.953	**0.996**	**0.988**
IBK	0.945	0.832	0.953	0.856	0.973	0.917	0.957	0.869	0.965	0.892	**0.980**	**0.941**	0.969	0.905
Dagging	0.977	0.929	0.980	0.942	0.957	0.869	0.980	0.941	0.953	0.855	0.977	0.929	**0.984**	**0.953**
Decorate	0.961	0.883	**0.984**	**0.954**	0.965	0.895	0.977	0.931	0.977	0.930	0.984	0.953	0.984	0.954
Logit_Boost	0.969	0.905	0.961	0.883	0.953	0.859	0.969	0.910	0.973	0.917	0.969	0.905	**0.980**	**0.941**
Hyper_Pipes	0.953	0.877	0.969	0.915	0.953	0.870	0.957	0.886	0.969	0.908	0.973	0.920	**0.977**	**0.934**
FT	0.977	0.930	0.973	0.918	0.961	0.881	0.949	0.845	0.957	0.874	0.984	0.953	**0.992**	**0.977**
LMT	0.984	0.955	0.969	0.905	0.957	0.872	0.957	0.869	0.953	0.859	0.977	0.929	**0.992**	**0.977**
NBTree	0.988	0.965	**0.992**	**0.977**	0.961	0.884	0.977	0.932	0.973	0.917	0.969	0.905	0.953	0.861
Random_Forest	0.973	0.918	0.973	0.917	0.957	0.869	**0.977**	**0.932**	0.957	0.868	0.973	0.917	0.973	0.918
Avearge	0.972	0.917	0.973	0.921	0.958	0.878	0.969	0.907	0.963	0.889	0.979	0.936	**0.980**	**0.940**
Median	0.977	0.930	0.973	0.918	0.957	0.872	0.973	0.921	0.967	0.900	0.977	0.932	**0.979**	**0.938**

Table 11. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy and MCC achieved for Shakespeare era plays and poems dataset, using EA2.

	ALL		(α,β)-k-FEATURE SET		CM1		*Mine*		*MineMink*		*MineMax MSTscore*		*MineMinkMax MSTscore*	
EA							EA1		EA1		EA1		EA1	
k	120		10		40		56		23		55		42	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Logistic	0.840	0.679	0.901	0.803	0.827	0.658	0.827	0.654	0.827	0.655	**0.914**	**0.827**	0.802	0.604
Simple_Logistic	0.889	0.779	0.926	0.852	0.914	0.829	0.889	0.779	0.889	0.779	0.914	0.827	**0.951**	**0.901**
SMO	0.864	0.728	0.951	0.902	0.926	0.852	0.889	0.778	0.926	0.852	0.951	0.902	**0.963**	**0.926**
S_Pegasos	0.852	0.708	**0.914**	**0.830**	0.852	0.703	0.827	0.655	0.901	0.804	**0.914**	**0.827**	0.840	0.685
Classification_Via_Regression	0.926	0.852	0.889	0.778	0.877	0.753	0.914	0.829	**0.938**	**0.877**	0.864	0.728	**0.938**	**0.878**
Dagging	0.877	0.760	**0.951**	**0.901**	0.914	0.829	0.840	0.685	0.827	0.658	0.864	0.731	0.864	0.741
Decorate	0.877	0.757	0.889	0.780	0.889	0.778	**0.951**	**0.902**	0.938	0.877	0.901	0.802	**0.951**	**0.902**
Logit_Boost	0.852	0.703	0.889	0.778	0.877	0.753	**0.926**	**0.852**	0.889	0.778	0.901	0.802	0.901	0.804
Multi_Class_Classifier	0.840	0.679	0.901	0.803	0.827	0.658	0.827	0.654	0.827	0.655	**0.914**	**0.827**	0.802	0.604
Rotation_Forest	0.926	0.853	**0.951**	**0.902**	0.914	0.827	0.938	0.878	0.926	0.855	0.926	0.852	0.914	0.830
NNge	0.877	0.753	0.926	0.852	0.914	0.827	**0.951**	**0.902**	0.926	0.853	0.889	0.780	0.926	0.853
LMT	0.889	0.779	0.926	0.852	0.914	0.829	0.889	0.779	0.889	0.779	0.914	0.827	**0.951**	0.901
Random_Forest	0.753	0.505	0.889	0.779	**0.914**	**0.827**	0.815	0.630	0.840	0.685	0.827	0.663	0.877	0.762
AVERAGE	0.866	0.733	**0.916**	**0.832**	0.889	0.779	0.883	0.767	0.888	0.777	0.899	0.800	0.898	0.799
Median	0.877	0.753	**0.914**	**0.830**	**0.914**	**0.827**	0.889	0.779	0.889	0.779	**0.914**	0.827	**0.914**	**0.830**

Table 12. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy ACC and MCC achieved for Alzheimer's disease dataset, using EA1.

	ALL		(α,β)-k-FEATURE SET		CM1		*Mine*		*MineMink*		*MineMax MSTscore*		*MineMink MaxMSTscore*	
EA							EA2		EA2		EA2		EA2	
k	120		10		40		41		31		27		15	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	0.852	0.708	0.889	0.780	0.889	0.780	0.852	0.708	0.877	0.757	**0.901**	**0.802**	**0.901**	**0.804**
Naive_Bayes	0.827	0.654	**0.926**	**0.852**	0.877	0.754	0.889	0.778	0.864	0.728	0.877	0.753	0.901	0.802
Simple_Logistic	0.889	0.779	0.926	0.852	0.914	0.829	**0.938**	**0.877**	**0.938**	**0.878**	0.901	0.805	0.877	0.753
SMO	0.864	0.728	**0.951**	**0.902**	0.926	0.852	0.901	0.802	0.914	0.827	**0.951**	**0.902**	0.926	0.855
IBK	0.914	0.827	0.901	0.802	0.827	0.654	0.877	0.756	0.889	0.779	0.914	0.829	**0.951**	**0.905**
LWL	0.889	0.783	0.889	0.783	0.889	0.783	0.864	0.729	**0.914**	**0.833**	0.889	0.788	0.901	0.810
Classification_Via_Regression	0.926	0.852	0.889	0.778	0.877	0.753	0.889	0.778	**0.938**	**0.877**	0.877	0.753	0.877	0.753
Dagging	0.877	0.760	**0.951**	**0.901**	0.914	0.829	0.901	0.807	0.901	0.807	0.901	0.802	0.926	0.852
Decorate	0.877	0.757	0.889	0.780	0.889	0.778	0.926	0.852	0.926	0.852	**0.938**	**0.877**	0.914	0.827
Filtered_Classifier	0.827	0.654	0.827	0.654	**0.840**	**0.679**	0.815	0.632	0.778	0.556	0.815	0.63	0.815	0.630
Logit_Boost	0.852	0.703	0.889	0.778	0.877	0.753	**0.901**	**0.802**	0.877	0.754	0.877	0.753	**0.901**	**0.802**
Random_Committee	0.84	0.687	0.914	0.833	0.889	0.783	0.889	0.780	0.889	0.791	0.864	0.728	**0.938**	**0.877**
Random_Sub_Space	**0.914**	**0.827**	0.877	0.753	0.901	0.802	0.901	0.805	0.840	0.678	**0.914**	**0.827**	0.901	0.803
Rotation_Forest	0.926	0.853	**0.951**	**0.902**	0.914	0.827	0.901	0.802	0.926	0.852	0.938	0.877	0.926	0.852
Decision_Table	0.815	0.630	0.778	0.559	0.802	0.609	**0.827**	**0.654**	0.802	0.604	0.753	0.506	0.815	0.630
NNge	0.877	0.753	**0.926**	**0.852**	0.914	0.827	0.889	0.778	0.914	0.827	**0.926**	**0.852**	**0.926**	**0.852**
ADTree	0.864	0.728	0.852	0.703	0.864	0.728	0.864	0.728	0.852	0.708	**0.877**	**0.753**	**0.877**	**0.753**
BFTree	0.915	0.830	0.915	0.830	0.902	0.804	0.915	0.855	0.864	0.728	0.927	0.854	**0.940**	**0.879**
J48graft	0.864	0.729	**0.901**	0.802	0.864	0.729	0.84	0.678	0.864	0.728	0.889	0.779	**0.901**	**0.803**
AVERAGE	0.875	0.751	0.889	0.779	0.881	0.762	0.881	0.765	0.875	0.751	0.884	0.769	**0.897**	**0.793**
Median	0.877	0.753	0.889	0.780	0.889	0.778	0.889	0.780	0.877	0.754	0.889	0.779	**0.901**	**0.803**

Table 13. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy ACC and MCC achieved for Alzheimer's disease dataset, using EA2.