ZAMBIA INFORMATION COMMUNICATION TECHNOLOGY (ICT) JOURNAL

Volume 3 (Issue 2) (2019) Pages 11-18

Feature Selection using Genetic Programming

George Mweshi

Department of Computer Science
University of Pretoria
Pretoria, South Africa

georgemweshi@yahoo.com

***Abstract* - Extracting useful and novel information from the large amount of collected data has become a necessity for corporations wishing to maintain a competitive advantage. One of the biggest issues in handling these significantly large datasets is the curse of dimensionality – a phenomenon where the performance of the data mining algorithms employed to mine these large datasets deteriorates due to the large search space created as a result of having irrelevant, noisy and redundant features in the dataset. Feature selection helps to tackle this problem by removing the unnecessary features. This in turn reduces the dimension of the data (as well as the search space) and consequently leads to an increase in the predictive accuracy and efficiency of the data mining algorithms. The study presented in this paper investigates the ability of Genetic Programming (GP), an Evolutionary Algorithm that is capable of automatically finding solutions in complex and large search spaces, to perform feature selection. A GP-based feature selection algorithm was implemented and tested on 5 benchmark classification datasets from the UCI repository. The feasibility of the approach was determined by examining the performance of four classifiers namely J48, Naives Bayes, PART and Random Forests using the GP selected features, all the original features and the features selected by the other commonly used feature selection techniques such as principal component analysis, information gain, relief-f and cfs. The experimental results show that GP not only selects a smaller number of features from the original features, the classifiers obtain better results when using the GP selected features than when using all the original features. Furthermore, when compared to the other feature selection techniques, GP obtains very competitive results.**

Keywords: curse of dimensionality, feature selection, genetic programming

I. INTRODUCTION

The recent technological advancements coupled with the reduction in the cost of computing hardware have allowed corporations to gather massive amounts of data for various purposes. Usually, this massive data contains both useful and irrelevant information that may not be essential in the data mining (DM) process since it may be redundant. The irrelevant data may not only mislead the data mining process but may also cause several problems for the DM algorithms such as slow processing time, overfitting [1] and poor accuracy.

Feature Selection (FS) [2] attempts to address these problems by selecting a smaller subset of features from the originally large number of features in the dataset. By filtering out the irrelevant, noisy and redundant data, FS enables the development of simpler, more comprehensible models that only consider relevant features. FS is considered a data pre-processing step [3] since it occurs before any analysis on the data can be performed. It has found many applications in problem domains such as online learning, clustering, classification, feature learning, and regression among others.

Although there have been many studies on FS and a wide range of literature on the strengths and weaknesses of many FS methods exists [4], [5], [6], applying these methods to datasets with a large number of features is still a challenging task due to the large solution search space which grows exponentially with the increase in the number of features in a dataset. For example, a dataset with k features has 2^k possible feature subsets to search from. This high complexity needs more powerful and efficient searching techniques. Heuristic based search strategies such as evolutionary algorithms have been proposed for this purpose.

Evolutionary Algorithms (EAs) have proved to be very effective at finding optimal solutions in complex and large search spaces [7]. For instance, Genetic programming (GP), a well-known evolutionary algorithm, has been succesfully used to find good solutions for various mathematical problems by automatically evolving mathematical models without any pre-defined template such as linear or non-linear.

The study presented in this paper aims to investigate the ability of GP to successfully perform FS on a dataset containing many features. In order to do this, we first develop and apply a GP algorithm on a FS task. We then compare the number of features selected by the developed GP algorithm with the original number of features as well as the number of features selected by other commonly used FS techniques such as CfsSubsetEval [8], Principal Component Analysis (PCA) [9], RELIEF-F(REFS-F) [10] and Information Gain (IG) [11]. Furthermore, the classification accuracy of four different classifiers, namely the J48 [12] decision tree, Naives Bayes (NB) [13], PART [14] and Random Forests (RF) [15] when using the GP selected features is compared with the accuracy of the classifiers when using all the features in the datasets and the features selected by the commonly used FS techniques. To be more specific, we investigate the following objectives:

1. Whether GP is capable of automatically selecting a smaller set of features that can be used to achieve a better classification accuracy as opposed to using all the features in the dataset.
2. Whether the J48, NB, PART and RF classifiers perform better when using the GP selected features than when using the features selected by PCA, REFS-F, CfsSubsetEval and IG.

The rest of the paper is organized as follows; Section II provides a brief background on FS and reviews some of works that have employed GP to perform FS in the literature. Section

III discusses the GP algorithm used to perform FS in this study. The experimental setup used to evaluate the GP algorithm is presented in section IV. The results obtained are presented and discussed in section V. The conclusion and future works are presented in section VI.

II. BACKGROUND AND RELATED WORK

A. *The curse of dimensionality*

Data is usually represented as a table consisting of examples (or instances) which are described by a fixed number of features (or attributes). For classification tasks [16], these features provide useful information for the class and one might assume that having more features helps to improve the class description. However, having more features reduces the performance of the classification algorithms (classifiers). This is because datasets with many features are more likely to contain irrelevant or redundant features which may mislead the classifiers. In addition, most classifiers are designed for low-dimensional space and as data becomes sparser (as is the case in high dimensional space) their performance reduces and they subsequently overfit [17]. This phenomenon is referred to as the curse of dimensionality. Dimensionality reduction techniques such as FS help to tackle this problem.

B. *Feature Selection*

There are many definitions of FS in the literature [18]. For example, Kira et al. [19] define FS as "*the problem of choosing a small subset of features that ideally is necessary and sufficient to describe the target concept*". Koller et al. [19] define FS as "*the process of finding a subset of features such that the resulting class distribution, given only the selected features, approximates the original class distribution as closely as possible*". FS essentially aims to remove the irrelevant, redundant and noisy features from a dataset in order to improve the performance, predictive accuracy and comprehensibility of the DM algorithms employed in the data mining process. It is a very important data preprocessing technique in data mining [20].

FS techniques can be categorized into two methods, namely feature subset selection and feature ranking, based on how the features are combined for evaluation [21]. Feature subset selection methods construct feature subsets using a search strategy and often use FS metrics such as correlation, consistency etc., to determine the best subset. However, the construction and evaluation of feature subsets makes these methods computationally expensive. Feature ranking methods on the other hand use simple FS metrics such as information gain [11], symmetric uncertainty [22], gain ratio [23], χ^2 [24], odds-ratio [25] etc. to assign a degree of relevance to the features. The top ranked features are selected as relevant features. This approach is computationally cheaper but does not deal with redundant data.

FS techniques may also be categorized into four algorithm types, namely wrapper, filter, embedded and hybrid, based on how machine learning algorithms (MLAs) are used in the FS process. In wrapper type algorithms, optimal feature subsets are adapted to specific MLAs adopted. The result is a high classification accuracy only for the MLA used. Furthermore, these algorithms do not generalize well, and the computational costs are very high. Filter type algorithms do not use MLAs in selecting features i.e. features are selected before learning. This reduces the computational costs and produces better generalization making them suitable for high-dimensional data. Embedded type algorithms use MLA's but with lower computational costs than wrappers. Hybrid methods combine wrapper and filter type algorithms. A few examples of filter based FS methods include: FOCUS algorithm [26], RELIEF algorithm [27] and its extension RELIEF-F [10].

The main problem with filter-based FS methods is their use of simple FS metrics which limit them to only search among individual features. As the search space grows, these methods not only become unreliable, they also fail to find other hidden and complex relationships between the predictive features and their target class labels [28]. Other more powerful heuristic searching strategies such as evolutionary algorithms (EAs) have been proposed to deal with these problems.

EAs are population-based random searching techniques that are able to find good enough or near-optimal solutions by evolving a population of candidate solutions using genetic operators [29]. The use of EAs to perform FS has been tackled by many researchers [30],[31],[32],[33]. Some of the commonly used EAs are Genetic Algorithm (GA), Genetic Programming (GP), Grammatical Evolution (GE) and Gene Expression Programming (GEP. This research will focus on GP.

C. *Genetic Programming*

GP, first introduced by Koza [34], evolves computer programs that perform a desired task using biologically inspired methods. In tree-based GP (which is also used in this study), computer programs are represented as syntax trees rather than lines of code. The syntax trees (see Fig. 1) consist of function and terminal nodes, where the function nodes are the inner nodes of the tree while the terminal nodes are leaf nodes. GP trees can also be expressed using a notation similar to Lisp where functions come before their arguments.

GP is a random and stochastic search technique. This means that it provides no guarantees that it will always find a solution. Nevertheless, it has been successfully used to solve in many real-world application domains such as cyber security[35], text mining [36] and medicine[37] to mention but a few.

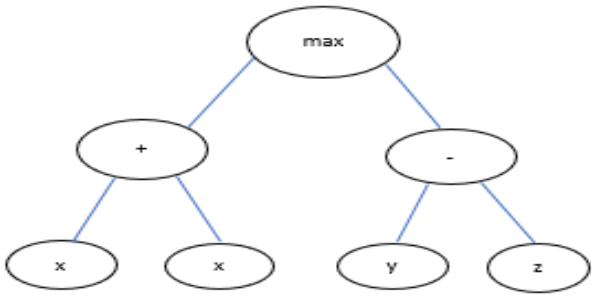

Fig. 1 GP tree for expression: max (x + x, y - z)

D. *GP Algorithm*

A typical GP algorithm starts by first creating an initial population of individuals (computer programs) from the functions and terminals defined for the problem. The fitness (or quality) of an individual is specified by a fitness value which is calculated using a fitness function. The fitness function determines how well the individual solves the problem being adrressed. Genetic operators such as crossover, reproduction and mutation are used to generate new offspring during evolution. The overall process is depicted in Fig.2

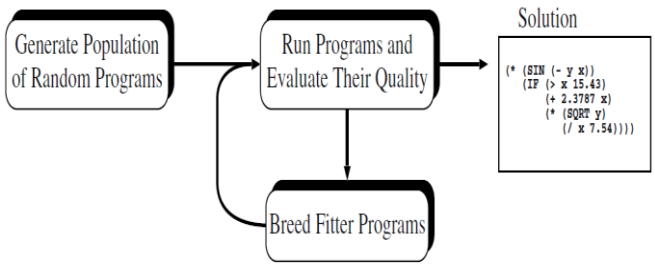

Fig. 2 GP Algorithm, extracted from [39]

Algorithmically, the main steps in GP are*:*

1. Initialize a random population of individuals from the specified functions and terminals;
2. Calculate and assign a fitness value (using the fitness function) to each individual in the population;
3. Repeat the following until the termination criteria is met:
   - Using a selection method, select one or two fittest individuals from the population;
   - Generate new individuals (offspring) for the next generation by applying genetic operators to the earlier selected individuals;
   - Assign fitness values to each new individual;
   - Replace the individuals in the old population with the new individuals;
4. Return the fittest individual as the best solution.

In general, before GP can be used to solve a problem, the following must be specified: function set, terminal set, fitness function and other GP run control parameters such as the size of the population, method for generating the initial population, stopping criteria, selection method, mutation, reproduction and crossover application rates etc.

E. GP preparatory steps

One of the advantages of GP is in the way it evolves computer programs (or solutions). Unlike other EAs in which the solution structure remains the same, GP can automatically adjust both its parameters and the structure of solutions during evolution. Some of the most important parameters and components of a typical GP system are described below.

1) Function Set: This is a set of all the functions the GP individuals will use. These functions can range from simple arithmetic functions {+, -, *, /}, logical functions {AND, OR, NOT} to more complex functions. The functions to use are dependent on the problem one is trying to solve.

2) Terminal Set: This is the set of all the variables and constants that will be used to solve the problem. This set is also problem dependent. For example, in a FS problem such as ours, the terminal set consists of all the original features in the dataset and some randomly generated constants.

3) *Initial Population Generation Methods:* These methods are used to generate an initial population of individuals. The most commonly used methods in tree-based GP are grow, full and ramped-half-and-half [34].

4) Fitness function: A fitness function is a measure of how well a GP program solves a problem. It is user defined and depends on the type of problem one is trying to solve. A good fitness function guarantees better solutions by providing better chances of survival to the stronger individuals. An example of a fitness function, and the one used in this study is the classification accuracy of the classifiers when using the GP selected features.

5) Selection Methods: These methods are used to select individuals (parents) for reproduction. The most common selection methods employed in GP are tournament selection and fitness proportionate selection [34].

6) *Genetic Operators:* These operations are performed on the selected parents to produce children (offspring). The operators alter, combine or duplicate the genetic material of the parents in order to produce offspring that drive the population towards a solution. Crossover, reproduction and mutation are the most commonly used genetic operators [39].

F. GP for Feature Selection

Unlike Genetic Algorithms (GAs) [40],[41],[42], there have been few attempts at using GP for FS. Most of the research works in the domain of dimensionality reduction have employed GP for feature construction [43] rather than FS due to its flexible representation.

Among the earliest published works to use GP for FS was the work done by Sherrah et al. [44]. In the study, the authors utilized a generalized linear machine classifier to determine the fitness of the GP selected features. The approach produced results that were comparable with other classifiers. In later works, Neshatian et al. [7] used a variation of NB as a classifier for a wrapper type GP algorithm which used a combination of bit-mask encoding and set operators for find optimal feature subsets. The algorithm significantly reduced the dimensionality of the datasets and led to improvements in both the processing time and performance of certain classifiers. In [45], Hunt et al. also developed a wrapper typed GP based hyper-heuristics for adding and removing features. These wrapper type algorithms however suffered from high computational costs.

Friedlander et al. [46] proposed the use of a weight vector containing all feature weights and GP selected features based on this vector. However, this could only work for problems with small number of features. Other researchers such as Ahluwalia and Bull [47] proposed using automatic defined functions (ADFs[48] with the k-nearest neighborhood (K-NN) [49] classification algorithm to perform both feature extraction and selection. Gray et al. [50] decided on the features by analyzing GP classifiers for binary classification problems. No FS was performed during the GP evolution. In [33], Muni et al. used GP to simultaneously perform online FS and construct classifiers using the features selected.

In summary, GP has been quite successful in FS tasks even though there have been few attempts as compared to GAs. GP has also shown promising results in solving problem where they have been a large number of features but few examples. A comparative study of the various FS techniques is however a very difficult task because no specific guidelines regarding the strengths and weakness of alternative techniques exist. Hence, the performance of a FS technique is heavily influenced by the machine learning method used.

III. METHODOLOGY

Algorithms and experiments were chosen as the research approach. These were necessary in order to analyse the changes in the performance of classifiers when: using GP selected features; all the features; and features selected by FS techniques such as PCA, RELIEF, information gain etc. In this study, we particularly investigated whether the classifiers could achieve a higher classification accuracy by using the GP selected features.

The JGAP [51] package was used to implement the basic GP Algorithm. The WEKA java library [52] was used for the J48, PART, NaiveBayes and Random Forest classifiers as well as PCA, CfsSubsetEval, REFS-F and IG techniques.

A. Datasets

Five binary classification datasets from the publicly available UCI Machine Learning Repository[53] were used. The description of the datasets is presented in TABLE I.

TABLE I
UCI datasets for Classification tasks

Dataset	**# of features**	**# instances**	**# of classes**
breast-cancer-wisconsin	10	699	2
Arcene	10000	900	2
Gisette	5000	13500	2
Madelon	500	4400	2
Wdbc	30	569	2

B. GP Algorithm Parameters

In order to perform FS, a GP algorithm was first developed and implemented. The implemented algorithm was then used to find relationships between the predictive features and their target class. The best features appearing in the best individuals formed the feature subset. Feature ranking was achieved by counting the occurences of the features in the best individuals. The parameters used for the GP algorithm are summarised in Table II.

TABLE II
GP Algorihm Parameters

Parameter	**Value**
Terminal set	Dataset features, random constant values
Function set	+, -, *, /, √, if, max
Initial Population	Ramped Half and Half
Population Size	# of features * β
Max. generations	50
Max. Tree Depth	20
Selection Method	Tournament
Size of Tournament	7
Mutation	0.2
Crossover	0.8
Reproduction rate	0.2
Fitness weighting μ	0.95 - 0.98

The number of features *F* in the datasets varied from a few tens to thousands and as such the search spaces for each dataset were different. As a result, the size of the population was taken to be proportional to the number of features in the datasets and this was done so that GP could explore more areas in the search space. The population size *P* was set by multiplying *P* with a coefficient β where β = {3,1} if *F* less than 1000 or more than 5000 respectively. This was done because of computer memory limitations.

GP returns a single floating-point value as output and this value was used to predict the class of the output. If the GP program output was ≤ 0.0, the instance was classified as class1; otherwise as class2. The fitness of a GP expression was calculated as the percentage of outputs correctly predicted. The features in the individual with the best fitness were considered the best subset of features. If the two or more individuals had the same fitness value, the individual with fewer features was chosen as the best solution. To terminate the GP evolutions, a fitness weighting threshold μ was used in addition to setting the maximum number of generations. The weighting threshold was used to set the desired GP prediction accuracy for the best feature subset in order to speed up the time taken by GP to find solutions. The features in the best solution were used to create a new dataset consisting of only these selected features. The new dataset was then used by the J48, PART, NaiveBayes and Random Forest classifiers. Because GP is a random and stochastic algorithm, different subsets of features were selected on each run. As a result, 30 different GP runs were carried out for each dataset. The average classification accuracy and standard deviation of the classifiers were captured. All the experiments were run on a PC with Intel Quad Core i7-3615 CPU @ 2.3GHz, Windows 10 Pro and 8GB of RAM.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

As indicated earlier, 30 experiments were carried out for each dataset and the performance results shown in the tables represent the average values. Table IV shows the classification accuracy percentages (%) of the J48, NB, PART and RF classifiers on the five datasets using all the features (represented as None) verses the features selected by GP. The values in bold show the best classification accuracy for the classifier. Table V includes the classification accuracy achieved by the classifiers using the top features selected by PCA, CfsSubsetEval, IG and REFS-F techniques. For datasets with a larger number of features, only the top 50 ranked features were selected. This was empirically determined in order to maintain a high classifier performance since selecting a smaller number of features produced poor results.

TABLE III
Experimental Results

Classifier Accuracy for GP selected features vs Original features						
Dataset	FS Technique	Features	J48	NB	PART	RF
Breast	None	10	93.42	96.14	93.57	**96.57**
	GP	5	**94.86**	**96.28**	**95.15**	96.43
Arcene	None	10000	74.00	70.00	73.00	79.00
	GP	44	**84.90**	**83.00**	**81.00**	**91.80**
Gisette	None	5000	90.30	92.20	**88.60**	**93.80**
	GP	30	**92.80**	**93.80**	84.70	93.40
Madelon	None	500	70.00	68.00	69.00	78.00
	GP	6	**75.00**	**71.00**	**72.00**	**80.70**
Wdbc	None	30	93.14	92.97	93.49	96.49
	GP	7	**95.44**	**96.57**	**95.08**	**96.84**

From Table III, it can be seen that GP selects a smaller number of features (5, 44, 30, 6,7) when compared to the original features. The classifiers also achieve a better classification accuracy by using the GP selected features as expected. It can also be observed that the overall best classifier in terms of classification accuracy was RF. In Table IV, it can further be seen that the run times were significantly reduced. All the results observed point to the fact by removing the irrelevant and redundant features, the performance of the classifiers significantly improves. This phenomenon proves that not all the features in the dataset are useful.

TABLE IV
Experimental Results

Dataset	FS	Classifier	Accuracy+SD	Run Time(s)
Breast	None	J48	93.42 +3.95	4.03
		PART	93.57 +4.79	4.02
		NB	96.14 +2.12	4.01
		RF	**96.29 +4.05**	4.17
	GP	J48	94.86 +3.79	2.02
		PART	95.15 +4.48	2.01
		NB	96.28 +2.41	2.01
		RF	**96.43 +2.94**	2.04
Arcene	None	J48	74.00 +3.95	9.03
		PART	73.00 +4.79	10.02
		NB	70.00 +2.12	10.00
		RF	**79.00 +4.05**	9.13
	GP	J48	84.90 +3.54	4.01
		PART	81.00 +1.48	5.01
		NB	81.00 +1.11	4.01
		RF	**91.80 +1.84**	4.02
Gisette	None	J48	90.30 +2.97	5.12
		PART	88.60 +2.44	7.00
		NB	92.20 +1.78	6.41
		RF	**93.80 +1.33**	4.29
	GP	J48	92.80 +1.79	2.01
		PART	89.70 +1.48	3.20
		NB	93.80 +1.31	2.01
		RF	**94.40 +1.12**	2.12
Madelon	None	J48	70.00 +3.95	3.03
		PART	69.00 +4.79	3.02
		NB	68.00 +2.12	3.01
		RF	**78.00 +4.05**	2.40
	GP	J48	75.00 +3.79	1.20
		PART	72.00 +4.48	2.35
		NB	71.00 +2.41	1.40
		RF	**80.70 +2.94**	1.02
Wdbc	None	J48	93.14 +3.55	2.06
		PART	93.49 +3.93	2.40
		NB	92.97 +3.87	1.51
		RF	**96.49 +2.72**	1.26
	GP	J48	95.44 +2.74	1.03
		PART	95.08 +3.23	1.20
		NB	96.57 +3.90	1.01
		RF	**96.84 +3.12**	1.10

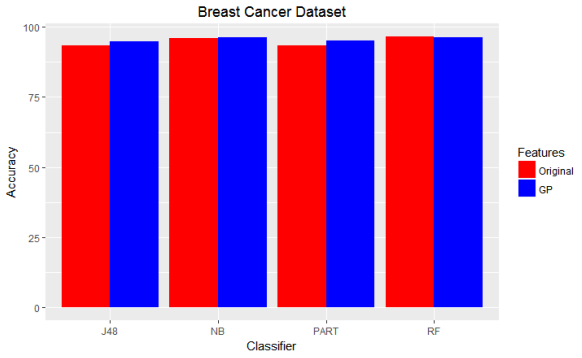

Fig. 3 Classification Performance for Breast Cancer Dataset

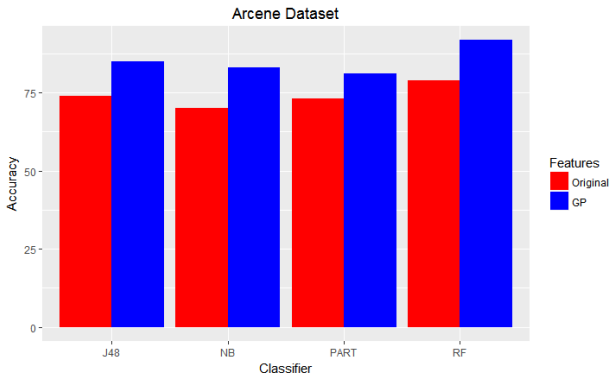

Fig. 4 Classification Performance for Arcene Dataset

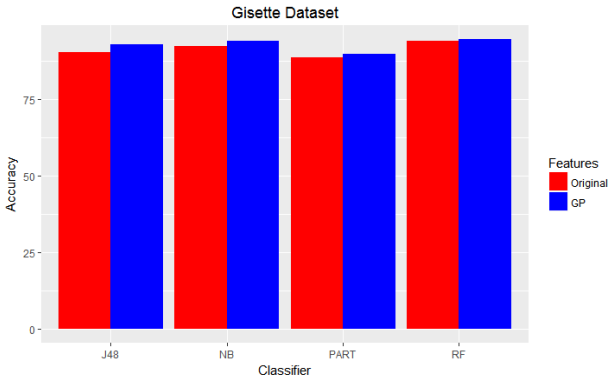

Fig. 5 Classification Performance for Gisette Dataset

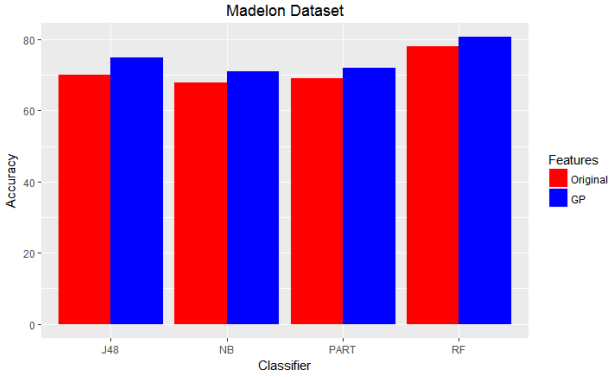

Fig. 6 Classification Performance for Madelon Dataset

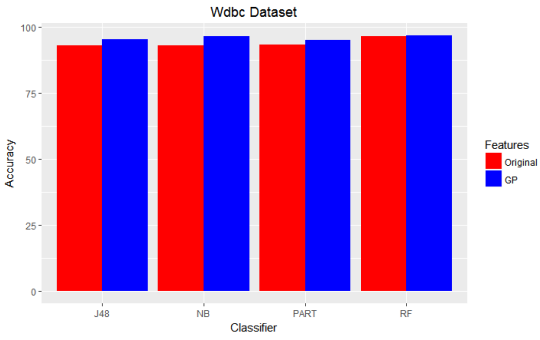

Fig. 7 Classification Performance for Wdbc Dataset

TABLE V
Combined Experimental Results

Comparing Classification Accuracy for FS Techniques						
Dataset	FS Technique	Features	J48	NB	PART	RF
Breast	None	10	93.42	96.14	93.57	96.57
	PCA	8	**95.56**	94.85	**95.85**	96.70
	CfsSubset	9	93.84	95.85	94.70	96.28
	IG	7	94.13	**96.42**	94.13	**96.85**
	REFS-F	7	94.42	**96.42**	93.13	96.57
	GP	5	94.86	96.28	95.15	96.43
Arcene	None	10000	74.00	70.00	73.00	79.00
	PCA	50	81.50	82.47	84.90	92.68
	CfsSubset	53	81.00	**93.00**	80.00	**93.00**
	IG	50	81.00	77.00	**85.00**	91.00
	REFS-F	50	80.00	77.00	80.00	79.00
	GP	44	**84.90**	83.00	81.00	91.80
Gisette	None	5000	90.30	92.20	88.60	93.80
	PCA	50	92.60	90.85	92.10	**94.58**
	CfsSubset	68	91.20	93.30	**92.30**	94.30
	IG	50	87.80	87.90	88.10	91.80
	REFS-F	50	91.00	86.10	89.00	93.70
	GP	30	**92.80**	**93.80**	89.70	94.40

Madelon	None	500	70.00	68.00	69.00	78.00
	PCA	50	66.15	60.48	68.94	77.54
	CfsSubset	8	65.95	59.94	60.20	74.58
	IG	50	65.30	59.50	66.45	70.40
	REFS-F	50	62.55	59.66	67.90	72.10
	GP	6	**75.00**	**71.00**	**72.00**	**80.70**
Wdbc	None	30	93.14	92.97	93.49	96.49
	PCA	10	94.55	91.56	93.67	95.78
	CfsSubset	11	94.02	94.55	94.55	96.48
	IG	10	92.09	92.97	92.44	94.90
	REFS-F	10	94.72	94.38	94.73	96.13
	GP	7	**95.44**	**96.57**	**95.08**	**96.84**

From Table V, we can see that GP performs considerably well in comparison to the other FS techniques. For example, we can see that for the Madelon and Wdbc datasets, GP outperforms all the FS techniques. We can also observe that on average, with respect to the classification accuracy, GP outperformed IG and REFS-F for all four classifiers. When the number of features was small, there was little variation between the FS techniques. This further showed that GP is able to select the best features. When the number of features was large, the performance of PCA, IG, REFS-F significantly reduced. This behaviour was expected as these techniques fail to handle the complex relationships between the features and the class

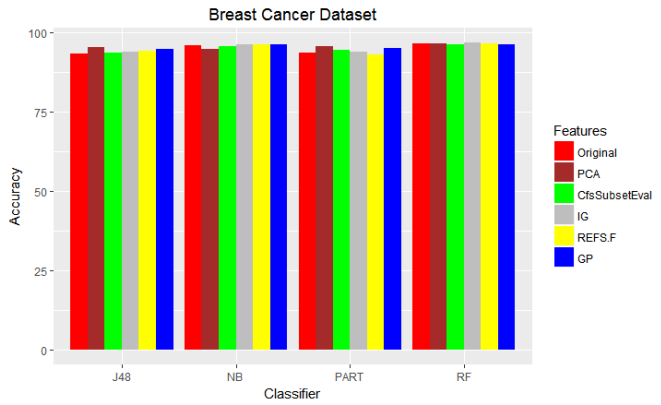

Fig. 8 Classification Performance for Breast Cancer Dataset

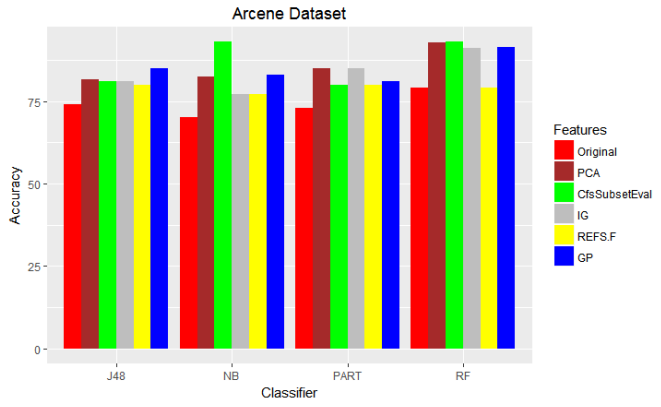

Fig. 9 Classification Performance for Arcene Dataset

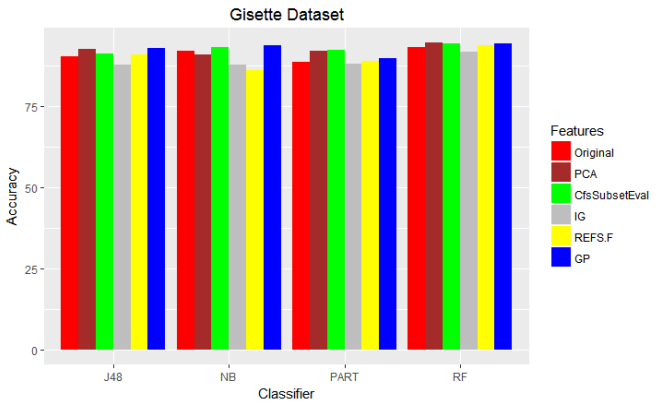

Fig. 10 Classification Performance for Gisette Dataset

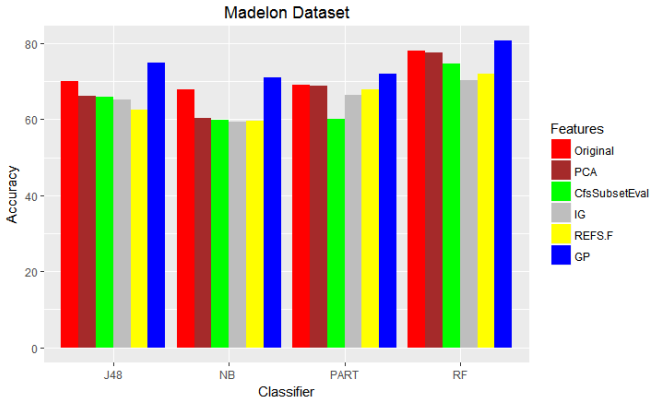

Fig. 11 Classification Performance for Madelon Dataset

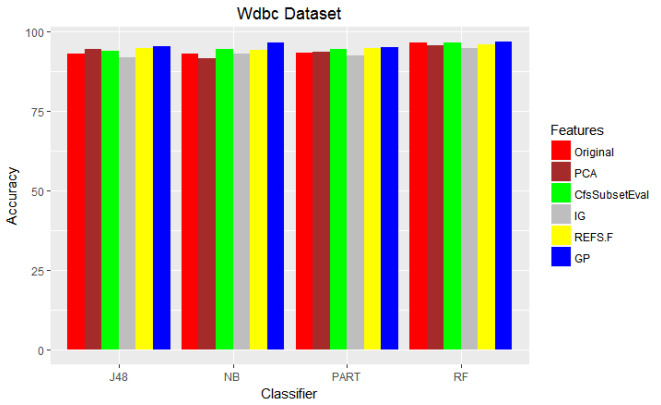

Fig. 12 Classification Performance for Wdbc Dataset

A. GP Feature Selection Performance

From the results obtained, it can be seen that the performance of the classifiers using the GP selected features was generally better than the performance of the classifiers using all the original features. In comparison to the other FS techniques, we can see that even though GP selected a fewer number of features, the performance of the classifiers using the GP selected features in most cases was better than all of other FS techniques. In the other instances were GP did not outperform the other FS techniques, most likely due to the parameter settings used in the research, we can see that it was not far off.

By further comparison, we can also see that in most cases, using features selected by the other FS techniques also improved the classification performance when compared to using the original features. This shows that the classifiers perform better with a smaller number of features as opposed to thousands.

V. CONCLUSION AND FUTURE WORKS

The main aim of this paper was to investigate the feasibility of performing FS using GP. We successfully achieved this aim by implementing a basic GP algorithm which was able to select a smaller number of relevant features from the datasets consisting of a large number of features. The results show that using the GP selected features resulted in a better classification performance by the J48, RF, NB, PART classifiers than using all the original features. The classifiers computational time was also significantly reduced. Even though this was a basic GP algorithm, in comparison to PCA, IG, Cfs and REFS-F, the performance was very competitive despite using a smaller number of features. In fact, for FS using the madelon and wdbc datasets, GP outperformed all the FS techniques. GP can therefore improve the performance of data mining algorithms as a pre-processing technique.

The random nature of GP makes it produce different results on each run. This presents many challenges in the sense that the features selected by GP are always different on each run. In some of the experiments, the performance of classifiers using GP selected features fluctuated. This was not the same for the other FS techniques. Therefore, in future, it would be wonderful to investigate ways in which to make the results obtained by GP consistent i.e. make GP select the same features on each run. It would also be interesting to investigate the application of other versions of GP such as grammatical evolution to FS.

REFERENCES

[1] G. I. Webb, "Overfitting," in *Encyclopedia of Machine Learning and Data Mining*, 2017.

[2] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.*, 2011.

[3] I. Kononenko and M. Kukar, "Data Preprocessing," in *Machine Learning and Data Mining*, 2013.

[4] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowl. Inf. Syst.*, vol. 34, no. 3, pp. 483–519, 2013.

[5] Z. M. Hira and D. F. Gillies, "A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data," *Adv. Bioinformatics*, 2015.

[6] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.

[7] K. Neshatian and M. Zhang, "Dimensionality reduction in face detection: A genetic programming approach," in *Image and Vision Computing New Zealand, 2009. IVCNZ'09. 24th International Conference*, 2009, pp. 391–396.

[8] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[9] J. M. Kinser, "Principle Component Analysis," in *Image Operators*, 2018.

[10] R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, "Relief-based feature selection: Introduction and review," *Journal of Biomedical Informatics*. 2018.

[11] B. Azhagusundari and A. S. Thanamani, "Feature Selection based on Information Gain," *Int. J. Innov. TechnolB Azhagusundari Antony Selvadoss Thanamani. 2013. Featur. Sel. based Inf. Gain. Int. J. Innov. Technol. Explor. Eng. (IJITEE), 2(2)18–21.ogy Explor. E*, 2013.

[12] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on j48 algorithm for data mining," *Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 6, 2013.

[13] J. P. Kharoufeh, P. Keskinocak, J. J. Cochran, L. A. Cox, M. Goldszmidt, and J. C. Smith, "Bayesian Network Classifiers," in *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[14] K. Elekar, M. M. Waghmare, and A. Priyadarshi, "Use of rule base data mining algorithm for intrusion detection," in *2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015*, 2015.

[15] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble Machine Learning: Methods and Applications*, 2012.

[16] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[17] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[18] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," in *Data Classification: Algorithms and Applications*, 2014.

[19] D. Koller and M. Sahami, "Toward optimal feature selection," 1996.

[20] A. Kalousis, J. Prados, and M. Hilario, "Stability of feature selection algorithms: a study on high-dimensional spaces," *Knowl. Inf. Syst.*, vol. 12, no. 1, pp. 95–116, 2007.

[21] D. Asir, S. Appavu, and E. Jebamalar, "Literature Review on Feature Selection Methods for High-Dimensional Data," *Int. J. Comput. Appl.*, vol. 136, no. 1, pp. 9–17, 2016.

[22] B. Singh, N. Kushwaha, and O. P. Vyas, "A Feature Subset Selection Technique for High Dimensional Data Using Symmetric Uncertainty," *J. Data Anal. Inf. Process.*, 2014.

[23] P. P. R., V. M.L., and S. S., "GAIN RATIO BASED FEATURE SELECTION METHOD FOR PRIVACY PRESERVATION," *ICTACT J. Soft Comput.*, 2016.

[24] N. Spolaôr and G. Tsoumakas, "Evaluating feature selection methods for multi-label text classification," in *CEUR Workshop Proceedings*, 2013.

[25] D. Mladenić, "Feature subset selection in text-learning," 2005.

[26] H. Almuallim and T. G. Dietterich, "Learning With Many Irrelevant Features.," in *AAAI*, 1991, vol. 91, pp. 547–552.

[27] R. P. L. DURGABAI and R. B. Y, "Feature Selection using ReliefF Algorithm," *IJARCCE*, 2014.

[28] K. Neshatian and M. Zhang, "Pareto front feature selection: using genetic programming to explore feature space," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 1027–1034.

[29] A. Purohit, N. S. Chaudhari, and A. Tiwari, "Construction of classifier with feature selection based on genetic programming," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1–5.

[30] F. Viegas *et al.*, "A Genetic Programming approach for feature selection in highly dimensional skewed data," *Neurocomputing*, 2018.

[31] M. G. Smith and L. Bull, "Feature Construction and Selection Using Genetic Programming and a Genetic Algorithm," 2007.

[32] P. L. Lanzi, "Fast feature selection with genetic algorithms: a filter approach," 2002.

[33] D. P. Muni, N. R. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 36, no. 1, pp. 106–117, 2006.

[34] J. R. Koza, *Genetic programming II, automatic discovery of reusable subprograms*. MIT Press, Cambridge, MA, 1992.

[35] J. V. Hansen, P. B. Lowry, R. D. Meservy, and D. M. McDonald, "Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection," *Decis. Support Syst.*, 2007.

[36] R. Isele and C. Bizer, "Active Learning of Expressive Linkage Rules Using Genetic Programming," *SSRN Electron. J.*, 2018.

[37] W. La Cava, S. Silva, K. Danai, L. Spector, L. Vanneschi, and J. H. Moore, "Multidimensional genetic programming for multiclass classification," *Swarm Evol. Comput.*, 2019.

[38] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[39] W. B. Langdon, R. Poli, N. F. McPhee, and J. R. Koza, "Genetic programming: An introduction and tutorial, with a survey of techniques and applications," in *Computational intelligence: A compendium*, Springer, 2008, pp. 927–1028.

[40] I. S. Oh, J. S. Lee, and B. R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2004.

[41] C. F. Tsai, W. Eberle, and C. Y. Chu, "Genetic algorithms in feature and instance selection," *Knowledge-Based Syst.*, 2013.

[42] R. Leardi, "Genetic Algorithms in Feature Selection," in *Genetic Algorithms in Molecular Modeling*, 2007.

[43] K. Neshatian, M. Zhang, and M. Johnston, "Feature Construction and Dimension Reduction Using Genetic Programming," in *AI 2007: Advances in Artificial Intelligence*, 2007.

[44] J. Sherrah, R. E. Bogner, and B. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *Intelligent Information Systems, 1996., Australian and New Zealand Conference on*, 1996, pp. 284–287.

[45] R. Hunt, K. Neshatian, and M. Zhang, "A genetic programming approach to hyper-heuristic feature selection," in *Asia-Pacific Conference on Simulated Evolution and Learning*, 2012, pp. 320–330.

[46] A. Friedlander, K. Neshatian, and M. Zhang, "Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 2011, pp. 941–948.

[47] M. Ahluwalia and L. Bull, "Coevolving functions in genetic programming," *J. Syst. Archit.*, vol. 47, no. 7, pp. 573–585, 2001.

[48] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Stat. Comput.*, vol. 4, no. 2, pp. 87–112, 1994.

[49] P. Cunningham and S. J. Delany, "k-Nearest neighbour classifiers," *Mult. Classif. Syst.*, vol. 34, pp. 1–17, 2007.

[50] H. F. Gray, R. J. Maxwell, I. Mart\'\inez-Pérez, C. Arús, and S. Cerdán, "Genetic programming for classification and feature selection: analysis of 1H nuclear magnetic resonance spectra from human brain tumour biopsies," *NMR Biomed.*, vol. 11, no. 4–5, pp. 217–224, 1998.

[51] K. Meffert, N. Rotstan, C. Knowles, and U. Sangiorgi, "Jgap-java genetic algorithms and genetic programming package," *URL http//jgap. sf. net*, 2012.

[52] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[53] E. Dua, D. and Karra Taniskidou, "UCI (University of California Irvine) Machine Learning Repository," *Repository*, 2017. .