Evaluating data mining algorithms using molecular dynamics trajectories

Vasileios A. Tatsis

Department of Engineering Informatics and
Telecommunications, Section of Applied Informatics,
University of Western Macedonia, Vermiou & Ligeris,
Kozani 50100, Greece
E-mail: btatsis@uowm.gr

Christos Tjortjis*

Department of Computer Science,
University of Ioannina,
Ioannina 45110, Greece
E-mail: tjortjis@cs.uoi.gr
*Corresponding author

Panagiotis Tzirakis

Department of Engineering Informatics and
Telecommunications, Section of Applied Informatics,
University of Western Macedonia, Vermiou & Ligeris,
Kozani 50100, Greece
E-mail: tzirakis@csd.uoc.gr

Abstract: Molecular dynamics simulations provide a sample of a molecule's conformational space. Experiments on the μs time scale, resulting in large amounts of data, are nowadays routine. Data mining techniques such as classification provide a way to analyse such data. In this work, we evaluate and compare several classification algorithms using three data sets which resulted from computer simulations, of a potential enzyme mimetic biomolecule. We evaluated 65 classifiers available in the well-known data mining toolkit Weka, using 'classification' errors to assess algorithmic performance. Results suggest that: (i) 'meta' classifiers perform better than the other groups, when applied to molecular dynamics data sets; (ii) Random Forest and Rotation Forest are the best classifiers for all three data sets; and (iii) classification via clustering yields the highest classification error. Our findings are consistent with bibliographic evidence, suggesting a 'roadmap' for dealing with such data.

Keywords: classification; data mining; evaluation; molecular dynamics simulations.

Reference to this paper should be made as follows: Tatsis, V.A., Tjortjis, C. and Tzirakis, P. (2013) 'Evaluating data mining algorithms using molecular dynamics trajectories', *Int. J. Data Mining and Bioinformatics*, Vol. 8, No. 2, pp.169–187.

Biographical notes: Vasileios A. Tatsis received a BSc Degree in Chemistry from the University of Ioannina in 2003 and a PhD in Bioinformatics in 2007. His current research interests include computational biophysics, Grid computing and data mining.

Christos Tjortjis is an Assistant Professor elect in Decision Support and Knowledge Discovery systems at the International Hellenic University, School of Science and Technology, and an adjunct Senior Lecturer at the University of Ioannina, Department of Computer Science and at the University of W. Macedonia, Department of Engineering Informatics and Telecoms. He holds an MEng Degree in Computer Engineering and Informatics from Patras, a BA Degree in Law from Thrace, an MPhil Degree in Computation from UMIST and a PhD in Informatics from Manchester. His research interests are in decision support knowledge discovery and software engineering.

Panagiotis Tzirakis graduated first in his class from the Department of Engineering Informatics and Telecoms, University of W. Macedonia, Greece. He has been awarded two scholarships by the Greek State Scholarships Foundation. Currently, he is a postgraduate student in the Computer Science Department, University of Crete, Greece. His research interests include data mining, database design and management.

1 Introduction

Data mining was introduced in the 1990s as a means for recovering knowledge from large databases (Fayyad et al., 1996). Its usage has since been extended to many different types of data, and it is being currently used as a tool for extracting knowledge from numerous types of large data sets (text, web, code, time series, graph, multimedia and biological data) (Denaxas and Tjortjis, 2008; Han and Kamber, 2006; Kanellopoulos et al., 2007).

A 'side effect' of this success is the plethora of algorithms and methods available for data mining. Arguably, one needs to go through an 'algorithm mining' phase first, before being able to mine the data. ''The challenge is to choose a suitable algorithm from the toolbox, since each algorithm will produce different results'' (Andreopoulos et al., 2009). In other words, experts in various fields need to establish which of the algorithms are likely to be more suitable for their scientific domain.

We identified this problem when we had to deal with data sets which resulted from biomolecular simulations. Given that clustering has been found to yield unsatisfactory results in similar domains (Rao and Karplus, 2010), we decided to use classification, a well-established technique (Browne et al., 2004; Haoudi and Bensmail, 2006; Lancashire et al., 2009; Pei, et al., 2010; Chung and Kim, 2011). Yet, the

well-known data mining toolkit Weka alone offers 65 different classification algorithms, each equipped with different configuration options (Hall et al., 2009). Facing the challenge of selecting a few algorithms with the potential for yielding good results, we decided to conduct a comprehensive set of experiments to evaluate algorithmic performance and construct a roadmap for selecting suitable classification algorithms for computer simulation data sets.

Computer simulation offers the possibility to study biomolecules and their dynamic behaviour in great detail, thereby complementing information that is accessible by experiment (Fersht and Daggett, 2002; van der Kamp et al., 2008). One of the most widely used simulation techniques is Molecular Dynamics (MD). It is a widely applied and well-developed method that provides valuable insight into the structure, dynamics, and interactions of biological macromolecules (Adcock and McCammon, 2006; Karplus and Kuriyan, 2005). Recent increases in computer power and enhancements in algorithms have enabled the execution of simulations of proteins on a large scale, resulting in huge amounts of data. One very useful way to analyse these data and extract important patterns is to cluster or classify molecular conformations into groups, according to the similarity of their conformations (as measured by an appropriate metric, for example the Root-Mean-Squared Distance (RMSD)) (Feher and Schmidt, 2001; Yona and Kedem, 2005).

In this work, we apply several classification algorithms, including trees learning, nearest neighbour, Bayesian classification and neural networks to three different data sets, collected from MD trajectories of a cyclic and branched oligopeptide (Stavrakoudis, et al., 2003; Tatsis, et al., 2008). The MD trajectories commence from three different initial peptide conformations, characterised as 'good', 'intermediate' and 'poor' according to their geometric resemblance to the natural enzyme (α -chymotrypsin). Our goal is to evaluate the performance of these algorithms when applied to these data. This is important because classification can expose hidden correlations among molecular conformations that can be hidden in the complexity of data (Jain et al., 2010). To the best of our knowledge, this is the first attempt to design a roadmap for applying classification algorithms to molecular simulation data sets in the literature.

The paper is organised as follows: in Background and Algorithms (Section 2) we present the basic classification algorithms and in Section 3 we describe how the MD trajectories were collected and the procedure we followed to rank the classification algorithms. In Section 4, we report the results of these algorithms, when applied to the three data sets. We review and discuss results and present related work on algorithmic performance in Section 5. Concluding remarks and perspectives are given in Section 6.

2 Background and algorithms

There are many kinds of classification algorithms; based on collective experience in the field of data mining and the maturity of the techniques, 65 prominent classification algorithms were selected to be included and made accessible to a wider audience via Weka. These are grouped into Bayesian classifiers, functions,

Lazy algorithms, meta algorithms, meta nested dichotomies, rules, trees and miscellaneous algorithms.

Bayesian Classifiers. These are statistical classifiers that predict class membership by probabilities. Several Bayes' algorithms have been developed, such as Bayesian networks and naïve Bayes. *Naïve Bayes* algorithms assume that the effect that an attribute plays on a given class is independent of the values of other attributes. In practice, dependencies often exist among attributes; hence *Bayesian networks* are graphical models, which, unlike naïve Bayesian classifiers, can describe joint conditional probability distributions. Bayesian classifiers have exhibited high accuracy and speed when applied to large databases, and are especially popular in the biochemical and medical domains, for example, which use Bayesian networks to analyse DNA hybridisation arrays, and in medical diagnosis; related material can be found in Fayyad and Irani, (1993); John and Langley, (1995); Kononenko, (1993).

Functions. The functions group includes classifiers that can be written as mathematical equations in a reasonably natural way. *SimpleLogistic* builds logistic regression models fitting them using LogitBoost with simple regression functions as base learners (Landwehr et al., 2005). *Logistic* is an alternative implementation for designing and using a multinomial logistic regression model with a ridge estimator to guard against overfitting by penalising large coefficients (Lecessie and Vanhouwelingen, 1992). *MultilayerPerceptron* (MLP) is a feed forward Artificial Neural Network model that maps sets of input data onto a set of appropriate outputs. It consists of multiple layers of nodes in a directed graph which is fully connected from one layer to the next. *RBFNetwork* implements a Gaussian Radial Basis Function Network, deriving the centres and widths of hidden units using K-means and combining the outputs obtained from the hidden layer using logistic regression if the class is nominal and linear regression if it is numeric. SMO implements the Sequential Minimal Optimisation algorithm for training a Support Vector classifier, using polynomial or Gaussian kernels (Keerthi et al., 2001).

Lazy classifiers. Lazy learners store the training instances and do no real work until classification time. *IB1* is a basic instance-based learner which finds the training instance closest in Euclidean distance to the given test instance and predicts the same class as this training instance (Aha et al., 1991). *IBk* is a k-nearest-neighbour classifier that uses the same distance metric. The number of nearest neighbours (default $k=1$) can be specified explicitly in the object editor or determined automatically using leave-one-out cross-validation, subject to an upper limit given by the specified value. *KStar* is a nearest neighbour algorithm with a generalised distance function based on transformations (Witten and Frank, 2005). *LWL* is a general algorithm for Locally Weighted Learning, and it assigns weights using an instance-based method and builds a classifier from the weighted instances.

Meta algorithms. Meta algorithms take classifiers and convert them into more powerful learners. *Bagging* bags a classifier to reduce variance (Breiman, 1996). It works for both classification and regression, depending on the base learner. In the case of classification, predictions are generated by averaging probability estimates,

not by voting. With *Classification via Clustering*, a user-defined cluster algorithm is built with the training data presented to the meta-classifier and then the mapping between classes and clusters is determined. This mapping is then used for predicting class labels of unseen instances and with the algorithm *Classification via Regression*, classification is performed using regression methods. *END* is a meta classifier for handling multi-class datasets with 2-class classifiers by building an ensemble of nested dichotomies (Dong et al., 2005). *Ensemble Selection* classifier combines several classifiers using the ensemble selection method. *Ordinal Class* is a meta classifier that allows standard classification algorithms to be applied to ordinal class problems. *Random Committee* builds an ensemble of base classifiers and averages their predictions (Witten and Frank, 2005). Each one is based on the same data, but uses a different random number seed. *Rotation Forest* is an ensemble method which generates classifier ensembles using the whole data set to independently train decision trees. Diversity between trees is promoted via feature extraction (Rodriguez et al., 2006).

Meta Nested Dichotomies. This group contains three classifiers: *Class Balanced ND*, *Data Near Balanced ND* and *ND* for handling multi-class datasets with 2-class classifiers by building a random class-balanced tree structure (Dong et al., 2005).

Misc. This group contains three classifiers: The Fuzzy Lattice Reasoning (FLR) classifier is presented for inducing descriptive, decision-making knowledge (rules) in a mathematical lattice data domain, including space RN (Kaburlasos et al., 2007). *Hyperpipes*, for discrete classification problems, records the range of values observed in the training data for each attribute and category and works out which ranges contain the attribute values of a test instance, choosing the category with the largest number of correct ranges (Witten and Frank, 2005). *VFI* (Voting Feature Intervals) constructs intervals around each class by discretising numeric attributes and using point intervals for nominal ones, records class counts for each interval on each attribute, and classifies test instances by voting (Demiroz and Guvenir, 1997).

Rules. Weka includes nine algorithms for generating rules. *DecisionTable* constructs a decision table majority classifier (Kohavi, 1995). It evaluates feature subsets using best-first search and can use cross-validation for evaluation. *JRip* implements RIPPER, including heuristic global optimisation of the rule set (Witten and Frank, 2005). *NNge* is a Nearest-neighbour method for generating rules using non-nested generalised exemplars. *OneR* is the 1R classifier with one parameter: the minimum bucket size for discretisation (Holte, 1993). *Part* obtains rules from partial decision trees. It builds the tree using C4.5's heuristics with the same user-defined parameters as J4.8 (Witten, 2005). *Ridor* implements the RIpple-DOwn Rule learner (Witten and Frank, 2005). It generates the default rule first and then the exceptions for the default rule with the least (weighted) error rate. Then it generates the 'best' exceptions for each exception and iterates until pure. *ZeroR* predicts the test data's majority class (if nominal) or average value (if numeric) (Witten and Frank, 2005).

Trees. A decision tree is a tree-like structure, which starts from root attributes, and ends with leaf nodes. Generally, a decision tree has several branches consisting of

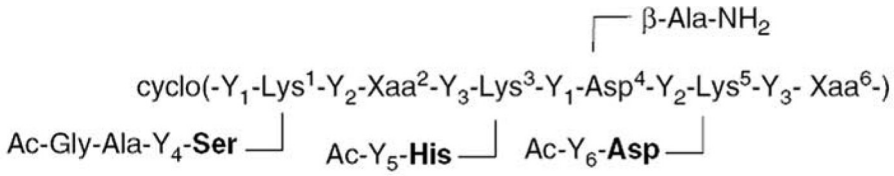
different attributes, the leaf node on each branch representing a class or a kind of class distribution. Decision tree algorithms describe the relationship among attributes, and the relative importance of attributes. Both the learning and classification steps of decision tree induction are generally fast. Algorithms like *J48* for generating a pruned or unpruned C4.5 decision tree, *FT* which is a first-decision tree classifier and *Cart* which implements minimal cost-complexity pruning, are included in this group. *RandomTree*, also a member of this group, chooses a test based on a given number of random features at each node, performing no pruning. *RandomForest* constructs random forests by bagging ensembles of random trees.

3 Methods

3.1 The data sets

Three data sets were used in this work, namely Ala54, Ala26 and Ala5, all produced by a bioinformatics application called Lysine based Trypsin Active Site (LysTAS) (Tatsis et al., 2006). LysTAS is an automated computer procedure that constructs peptidomimetic molecules satisfying a particular design, shown in Figure 1, and grades them against the active site's topology. Three initial conformations were subjected to MD simulations for 10 ns (following a 1 ns equilibration period), with a time step of 1 fs. Further computational details about the MD runs can be found in Tatsis et al. (2008). Structures were saved every 10 ps, so that 1000 conformations were stored and the last 900 were used for analysis.

Figure 1 Peptides modelled as serine protease mimetics. Active site residues Ser, His and Asp are in bold. Y_i denotes the D/L chirality. Xaa^2 and Xaa^6 accommodate the amino acids selected by the user. In this study, Xaa^2 and Xaa^6 were substituted with Ala



As a measure of resemblance to the natural trypsin active site, we measured the Root Mean Square Distances (RMSD) between the heavy atoms of the side chains of the catalytic triad residues (Asp102, His57, Ser195) of the enzyme and the corresponding side chain heavy atoms of the peptide to assign a similarity score.

$$\mathrm{RMSD}_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i^m - x_i^t\right)^2 + \left(y_i^m - y_i^t\right)^2 + \left(z_i^m - z_i^t\right)^2}. \quad (1)$$

The RMS distance between the atoms of the peptide (Figure 1) and the corresponding atoms of the enzyme (2ptn) is given by equation (1), where x^m, y^m, z^m are the Cartesian coordinates of the heavy atoms of the amino acids which comprise the

active site of the natural enzyme, and x^t, y^t, z^t are the Cartesian coordinates of the corresponding heavy atoms of the peptide.

The criterion (equation (2)) for a frame i of the trajectory to be counted as one with a favourable topology is (Wallace et al., 1996):

$$2\ \mathring{\mathrm{A}} \geq \mathrm{RMSD}_i = \mathrm{RMSD}\,[\mathrm{frame}_i^{\text{implicit solvation}} - \mathrm{TEMPLATE}]. \tag{2}$$

We have, also, calculated the Cartesian distances (d1 = His:HD1 – Asp:OD1, d2 = His:HD1 – Asp:OD2 and d3 = Ser:HG – His:NE2) between the atoms of the residues that are responsible for the formation of the hydrogen bond network in the active centre of the enzyme as an additional criterion of similarity. The distances between the C^a atoms of the active site's residues, d4 (His:C^{α}– Asp:C^{α}) , d5 (His:C^{α}– Ser:C^{α}) and d6 (Ser:C^{α}– Asp: C^{α}) show how close are the three residues that form the trypsin's active site.

3.2 Data pre-processing

Each data set contains 900 entries (instances), comprised eight attributes: the RMSD, the distances d1–d6 and the total potential energy of the MD trajectory's frames. More specifically, the RMSD was used as a class attribute, discretised to 1, 2, 3 and 4, according to whether its values ranged from: 0.5 to 1.0, 1.0 to 2.0, 2.0 to 5.0 or 5.0 to 10.0 respectively. The three data sets differ in the distribution of their values. For Ala54, the values of its attributes are more concentrated and the distribution is narrower, with smaller standard deviation (σ_T). For Ala26 and Ala5, the values distributions are wider and sparser, with larger standard deviation (σ_T). Based on this characteristic, Ala54 can be considered as 'good', Ala26 as 'intermediate' and Ala5 as 'poor'. These data sets were analysed using all the classification algorithms provided by the Weka 3.4.13 Data Mining toolkit, which was installed on a Windows machine equipped with 4 Intel Xeon 3.40 GHz processors and 8 GB RAM. We used 10-fold cross validation and the default Weka configuration settings. Using default settings was done for two reasons. Firstly, a classification scheme in Weka normally gets the default settings as described in the publication underlying that particular scheme. Secondly, there are no 'templates' for certain domains available, and changing the parameters, apart from presenting a 'combinatorial explosion' type of problem, would only produce a set of parameter values that optimise performance for the specific data sets at hand, rather than producing generic guidelines. A similar approach was followed in Janssen and Furnkranz (2011).

3.3 Ranking procedure

To assess the classification algorithms and their groups in Weka, we applied a simple ranking method. For every data set, the classifiers were sorted, in descending order, according to their classification error. Their ranking index for every data set was then summed and aggregated, resulting in the final ranks shown in Table 2.

4 Results

4.1 Overview

We used all 65 classification algorithms available in Weka, categorised in 8 groups, as shown in Table 1. Running NNge and BFTree algorithms on the Ala5 data set caused the system to run out of memory, even without using 10-fold cross-validation and after changing default memory usage to 1 GB, thus we exclude them from further discussion. We analysed statistically the classification error that these algorithms yielded, when the three data sets were used as input, and calculated the minimum, maximum and average classification error for each of the eight groups of algorithms. The results are summarised in Table 1 and detailed in Table 2. This subsection presents results in terms of groups of classification algorithms, as shown in Table 1.

Table 1 The minimum (*min*), maximum (*max*) and average (*avg*) classification error for the eight groups of classifiers in Weka, when applied to the three data sets. The minimum value for the average classification error for each of the three data sets is in bold

		Classification Error (%)								
		Ala54			*Ala26*			*Ala5*		
Group of classifiers	*Rank*	*min*	*max*	*avg*	*min*	*max*	*avg*	*min*	*max*	*avg*
Bayes	7	0.24	0.42	0.36	11.48	25.94	16.74	14.32	32.21	19.64
Functions	2	0.09	0.26	0.13	7.77	10.91	8.70	10.99	15.17	13.81
Lazy	2	0.09	0.36	0.16	6.04	15.54	8.43	11.65	15.20	13.04
Meta	5	0.07	42.35	1.97	5.37	44.91	15.12	9.42	32.21	17.31
Meta nested Dichotomies	**1**	0.10	0.12	**0.11**	6.68	6.93	**6.79**	10.42	10.43	**10.42**
Misc	8	0.14	0.27	0.21	15.85	41.56	27.28	21.31	41.92	31.39
Rules	5	0.10	0.36	0.20	6.76	25.94	13.38	10.61	32.21	15.26
Trees	2	0.06	0.36	0.15	5.34	25.94	9.96	9.63	32.21	13.00

We applied four Bayesian classification algorithms to the Ala54 data set resulting to classification error ranging from 0.24% to 0.42%, with an average of 0.36%. For the Ala26 data set, the average classification error is 16.74%, ranging from 11.47% to 25.94%. The classification error for the Ala5 data set ranges from 14.32% to 32.31%, and its average is 19.64%.

We employed six algorithms from the *functions* group of classifiers. More specifically, for the Ala54 data set, the lower and upper boundaries of the classification error are 0.09% and 0.26%, respectively, and the average is 0.13%. The classification error for the Ala26 data set fluctuates between 7.77% and 10.91%, and its average is 8.70%. The Ala5 data set's classification error minimum approximates the Ala26 maximum and reaches a maximum at 15.17%. Its average is 13.80%.

The four *Lazy* classification algorithms behave quite similarly to the *functions* group. More specifically, for the Ala54 data set, the average classification error is 0.16%, whilst the minimum and maximum is 0.09% and 0.36%, respectively. The classification error for Ala26 ranges from 6.04% to 15.54%, and its average is

Table 2 Total ranking and classification error for the three data sets Ala54, Ala26 and Ala5, for all 65 Weka classifiers

				Classific. error (%)	
Group	*Classifier*	*Rank*	*Ala54*	*Ala26*	*Ala5*
Bayes	BayesNet	41	0.24	11.48	14.32
	DMNBtext	54	0.36	25.94	32.21
	NaiveBayes	49	0.42	14.79	16.02
	NaiveBayesUpdateable	47	0.42	14.76	16.02
Functions	Logistic	37	0.10	8.14	15.17
	MultilayerPerceptron	27	0.09	7.77	10.99
	RBFNetwork	40	0.26	10.91	14.24
	SimpleLogistic	32	0.09	7.98	14.02
	SMO	36	0.11	8.72	14.62
Lazy	IB1	14	0.09	6.04	12.65
	IBk	14	0.09	6.04	12.65
	KStar	13	0.10	6.11	11.65
	LWL	44	0.36	15.54	15.20
Meta	AdaBoostM1	43	0.22	25.94	15.00
	AttributeSelectedClassifier	25	0.12	6.58	13.21
	Bagging	4	**0.10**	**5.68**	**9.42**
	ClassificationViaClustering	63	42.35	44.91	18.11
	ClassificationViaRegression	5	**0.08**	**6.01**	**9.81**
	CVParameterSelection	54	0.36	25.94	32.21
	Dagging	42	0.29	13.14	14.70
	Decorate	25	0.44	5.99	11.73
	END	8	**0.10**	**6.26**	**10.41**
	EnsembleSelection	6	**0.09**	**6.22**	**9.64**
	FilteredClassifier	28	0.14	7.94	11.58
	Grading	54	0.36	25.94	32.21
	LogitBoost	31	0.09	8.00	13.56
	MultiBoostAB	51	0.36	25.94	15.21
	MultiClassClassifier	39	0.11	11.28	15.16
	MultiScheme	54	0.36	25.94	32.21
	OrdinalClassClassifier	9	**0.10**	**6.58**	**10.41**
	RacedIncrementalLogitBoost	37	0.25	9.15	13.76
	RandomCommittee	3	**0.07**	**5.64**	**9.62**
	RandomSubSpace	18	0.13	6.68	11.46
	RotationForest	2	**0.08**	**5.37**	**9.47**
	Stacking	54	0.36	25.94	32.21
	StackingC	54	0.36	25.94	32.21
	Vote	54	0.36	25.94	32.21
Meta nested	**ClassBalancedND**	12	**0.10**	**6.68**	**10.42**
Dichotomies	DataNearBalancedND	17	0.10	6.75	10.43
	ND	19	0.12	6.93	10.42
Misc	FLR	53	0.27	15.85	41.92
	HyperPipes	44	0.14	24.42	30.93
	VFI	47	0.20	41.56	21.31
Rules	ConjunctiveRule	46	0.36	25.94	15.10
	DecisionTable	34	0.13	10.09	12.78
	DTNB	33	0.12	9.67	12.79

Table 2 Total ranking and classification error for the three data sets Ala54, Ala26 and Ala5, for all 65 Weka classifiers (continued)

				Classific. error (%)	
Group	*Classifier*	*Rank*	*Ala54*	*Ala26*	*Ala5*
	JRip	19	0.11	6.76	10.61
	NNge	0.13	6.99		
	OneR	49	0.33	20.57	15.55
	PART	21	0.10	7.10	10.75
	Ridor	29	0.14	7.40	12.28
	ZeroR	54	0.36	25.94	32.21
Trees	BFTree		0.14	7.17	
	DecisionStump	51	0.36	25.94	15.21
	FT	9	**0.08**	**6.73**	**9.83**
	J48	14	**0.09**	**6.78**	**10.38**
	J48graft	9	0.14	6.52	10.28
	LADTree	35	0.11	9.91	13.85
	LMT	7	0.12	6.32	9.69
	NBTree	23	0.12	6.89	10.82
	RandomForest	1	**0.06**	**5.34**	**9.63**
	RandomTree	30	0.12	7.30	13.24
	REPTree	23	0.12	7.29	10.53
	SimpleCart	21	0.14	7.36	10.33
	UserClassifier	54	0.36	25.94	32.21

8.43%. The corresponding values for the Ala5 data set are 11.65%, 15.20% and 13.6%, respectively.

We used 26 classification algorithms of the *Meta* group and their range of classification error was broad. For the first data set (Ala54) the maximum is 42.35% and the minimum is 0.07%, whilst the average is closer to the minimum and equal to 1.97%. The minimum error for the Ala26 data set is 5.37%, it reaches a maximum of 44.91% and its average is 15.12%. For the Ala5, the classification error fluctuates in narrower amplitude, between 9.42% and 32.21%, and it averages at 17.3%.

Three algorithms from the *Meta nestedDichotomies* group were evaluated in classifying the three data sets. For the first data set (Ala54), the minimum and maximum classification error is 0.10% and 0.12%, respectively. The same applies to the other two data sets (Ala26 and Ala5). More specifically, for the Ala26, the classification error varies from 6.68% to 6.93% and for the Ala5 from 10.42% to 10.43%.

The FLR, HyperPipes and VFI classifiers, members of the *Misc* groups, were also evaluated. For the first data set (Ala54), the minimum classification error is equal to 0.14%, and it reaches a maximum of 0.27%; its average is 0.21%. The minimum classification error for the Ala26 is 15.85%, the maximum 41.56% and its average 27.27%. For the Ala5 data set these statistical measures are increased, as expected and noticed in the previous algorithms. The average value of the classification error is 31.3% and it varies between 21.31% and 41.92%.

The nine classification algorithms of the *Rules* group were also assessed on how well they can classify data sets resulting from MD studies of a biomolecule. For the

first data set (Ala54), the minimum error is 0.10%, the maximum is 0.36% and its average value is equal to 0.20%. For the Ala26 data set, the upper and lower boundaries of the classification error are 25.94% and 6.76%, respectively. The average value of the classification error is 13.38%. The classification error for the third data set (Ala5) ranges from 10.61% to 32.21%, and its average is 15.2%.

Trees was the last group of classifiers that was evaluated. This group comprises 15 algorithms, available in Weka. For the first data set (Ala54) the lowest value of the classification error is 0.06%, which is the 'global' minimum for all classification algorithms that have been used. The maximum error of this algorithm is 0.36% and it averages at 0.15%. For the second data set the minimum and maximum are 5.34% and 25.94%, respectively. The trees classification algorithms when applied to the third data set (Ala5) showed the average of the classification error as equal to 13.00%.

It appears that the *Lazy* classification algorithms behave quite similarly to the *functions* algorithms. We noticed that the classification error for the *Meta nestedDichotomies* group fluctuates in a narrow amplitude ($\sim$0.10%). All in all, Bayesian classifiers performed worse than the other algorithms, in contrast with the meta group which exhibited the best performance, when applied to MD data sets. Furthermore, the classification algorithms' performance is consistent with the profile of the data sets, with regards to the shape of the distribution (standard deviation, σ_T) of their attributes' values. More specifically, the classifiers' error is lower for the 'good' data set (Ala54) and it increases for the next two datasets, 'intermediate' (Ala26) and 'poor' (Ala5).

4.2 Results evaluation

In this subsection, we present results for all 65 Weka classifiers, when applied to the three MD data sets, and focus on the best algorithms. Specifically, we consider here as '*best*' classifiers these which resulted in a classification error lower than the best group average error, for all three data sets shown in Table 1. In other words, we identify and review all the algorithms that yielded lower classification error than 0.11% for Ala54, 6.79% for Ala26 and 10.42% for Ala5 (produced by the *Meta nestedDichotomies* group). The classification error for all 65 Weka classifiers is shown in Table 2, and the best classifiers are indicated in bold font.

All in all, there are eleven (11) classification algorithms that, for all the data sets, result in classification error rate lower than 0.11%, 6.79% and 10.42%, respectively (see Table 2). These are: Bagging, ClassificationViaRegression, END, EnsembleSelection, OrdinalClassClassifier, RandomCommittee, RotationForest from the meta group, ClassBalancedND from the *Meta nestedDichotomies* group, FT, J48 and RandomForest from the *trees* group. We discuss these algorithms in their respective groups.

Moreover, we have implemented a simple method for ranking the classification algorithms, which is described in Section 3.3. These rankings are also shown in Table 2. We note that 8 out of the top 10 ranking algorithms are also amongst the 'best' classifiers, as discussed above. LMT and J48graft, ranking seventh and ninth respectively, are not amongst the 'best' classifiers. On the other hand, two classifiers belonging to the 11 best ones, did not make the top 10 in terms of rankings: ClassBalancedND and J48, in 12th and 14th place respectively.

None of the algorithms in the *bayes* group yield lower classification error than the lowest average group values (0.11%, 6.79% and 10.42%, respectively for the three data sets). The BayesNet classifier results in the lowest classification error for all three data sets, and is positioned in the 41st place, in the overall ranking (see Table 2).

None of the algorithms in the *functions* group yields lower classification error than the lowest average group values. MultilayerPerceptron is the best performing classifier in the group, holding the 27th position in the overall ranking.

None of the algorithms in the *lazy* group yields lower classification error than the lowest average group values. KStar scores the lowest classification error for this group, occupying the 13th place in the overall ranking.

Seven of the twenty-four classification algorithms of the meta group yield a classification error rate lower than the average error rate of the *Meta nestedDichotomies* group. These algorithms are the following: Bagging, Classification via Regression, END, Ensemble Selection, Ordinal Class Classifier, Random Committee and Rotation Forest. Rotation Forest and Random Committee succeed in classifying the data sets with the lowest error rate, in this group, and they are positioned in the second and third place, respectively.

The Class Balanced ND, from the *Meta nestedDichotomies*, results in a classification error rate lower than the average error rate of this group. Its error is lower than the group's average for Ala54 and Ala26, and equal to the group's average (10.42%) for Ala5, and it ranks in the 12th position.

Hyperpipes, the top algorithm for the *misc* group, is in the 44th place in the overall ranking. Its error is however higher than the *Meta nestedDichotomies* average error.

JRip and PART algorithms classified the three data sets with an error rate lower than the average of their group (*rules*), but higher than that of the *Meta nestedDichotomies*. They are ranked in the 19th and 21st place, respectively.

From the group of trees classifiers, the classification error rate of FT, J48 and RandomForest is lower than the average classification error of the meta nestedDichotomies group. RandomForest had the best performance compared to the other 62 classifiers, and is ranked first in the overall ranking. It results in the lowest classification error when applied to any of the three data sets (0.06%, 5.34% and 9.63% for the Ala54, Ala26 and Ala5, respectively).

In summary, seven *meta* classifiers performed better than the *Meta nestedDichotomies* average error, with all 7 of them being in the top ten of the overall ranking. One *Meta nestedDichotomies* and three tree classifiers performed better than the *Meta nestedDichotomies* average error, but only two tree classifiers made the top ten, with Random Forest topping the ranks. In total, there are nine classifiers (7 *meta* and 2 *tree* classifiers) which achieved both 'best' (i.e., better than the *Meta nestedDichotomies* average error) and 'top ten' status.

5 Discussion

5.1 Classification algorithms' comparison

In this work, we evaluated 65 classification algorithms available in Weka, using three computer simulation data sets. These data sets represent the time series of geometric and energetic characteristics of a cyclic-branched peptide, with potential enzyme

mimetic activity. The performance of the classifiers was assessed according to their classification error, when applied to these data sets.

The algorithms are categorised by Weka in 8 groups. Following our ranking procedure, explained in Section 3 and shown in Table 1, the *Meta nestedDichotomies* group qualifies for the first place. The *Functions*, *Lazy* and *Trees* groups share the second place. Seven out of the top eleven algorithms belong to the *meta* group of classifiers, whilst the other four top 11 classifiers are members of the *Trees* group. ClassBalanceND, which belongs to the *Meta nested Dichotomies* group, holds the 12th position.

If one was to choose from the 65 algorithms available in Weka for classifying MD simulations data, *meta* and *tree* classifiers would be a good place to start. In particular, our experiments indicated that the leading performers are those algorithms which featured in both the top ten and the list of 11 'best' algorithms that, for all the data sets, result in classification error rate lower than the average classification error of the *Meta nestedDichotomies* group. This leaves 7 meta and 2 tree classifiers with the potential for excellent performance.

This is not very surprising, given that meta and ensemble classifiers, combining more than one individual classifier, have been shown to produce high accuracy at the expense of potentially higher computational burden, especially for classifier training. We review related work in the following subsection. In addition, the selection of base classifiers influences the performance of ensemble algorithms greatly (Das and Sengur, 2010). Depending on the choice of base classifiers, an ensemble algorithm can outperform a single classifier.

Nine out of the top ten algorithms result in smaller classification error than the lowest average classification error of the meta nested Dichotomies group. LMT and J48.graft do not satisfy this criterion.

We also note that the classification error for the algorithms used increases from the Ala54 data set to Ala26. The main difference amongst these data sets is their values' distribution. The Ala54 data set has a smaller standard deviation (σ_T) than the other data sets, and its values' distribution is narrow. Ala5' values exhibit a sparser and wider distribution. The properties of the Ala26 data set are in between of those of Ala5 and Ala54.

All in all, Random Forest, a member of the *trees* group, is the best classifier, achieving for the first two data sets (Ala54 and Ala26) lower error than Rotation Forest, the second best classifier, whilst for the last data set (Ala5) it achieves higher error.

BFTree and NNge did not produce results for Ala54, whilst Classification via Clustering resulted in the highest classification error for the three data sets. The average classification error for the three data sets (Ala5, Ala26 and Ala54) was 0.84%, 13.22% and 16.17%, respectively.

5.2 Related work

Rodriguez et al. (2006) compared the Rotation Forest with the standard implementations of J48, Bagging, AdaBoost, and Random Forest available in Weka. The experimental results with 33 data sets from the UCI Machine Learning Repository showed that Rotation Forest outperformed all the other four algorithms by a large margin (Rodriguez et al., 2006). It has been found that

Rotation Forest has a similar diversity-accuracy pattern to Bagging, but is more accurate and slightly more diverse than Bagging.

Random Forest classifier, on the other hand, yields results competitive with boosting and adaptive bagging (Breiman, 2001).

A comparison of eight different classifiers on the problem of distinguishing all major protein subcellular location patterns in both 2D and 3D fluorescence microscope images has been performed (Huang and Murphy, 2004). The algorithms that were applied were: neural networks, support vector machines with linear, polynomial, radial basis and exponential radial basis kernel functions, and ensemble algorithms such as: AdaBoost, Bagging, and Mixtures-of-Experts. The highest accuracy (95.3%), for a specific data set, was achieved using neural networks.

A comparative study of four classification methods (Neural Networks, DMneural, Regression and Decision Tree), for effective diagnosis of Parkinson's disease, was carried out recently by Das (2010). The neural network classifier yielded the best score: 92.9% classification accuracy.

A comparative study of three ensemble classification algorithms: bagging, boosting and Random Subspace, was performed by using a data set containing 215 samples of valvular heart disease (Das and Sengur, 2010). Their main finding is that boosting, with base classifier SVM, improves performance, achieving accuracy of 98.4%.

A study of the performance of 21 classification algorithms, including bagging, Boosting and Random Forest, using seven gene expression data sets was carried out (Lee et al., 2005). They found that aggregating classifiers such as bagging, boosting and Random Forest improves the performance of the CART (Classification and regression tree) significantly. Also, Random Forest is the best tree algorithm (Bagging, Boosting and Random Forest) when the number of classes is moderate.

Naive Bayes, C4.5 and Random Forest were applied to galaxy classification (de la Calleja and Fuentes, 2004). Experimental results indicate that Random Forest results in higher accuracy, when considering three, five and seven galaxy types.

Finally, Random Forest and Rotation Forest were shown to perform well with Gene promoters data (Gupta et al., 2010) and the SCOP database (Dehzangi et al., 2010). Table S1 summarises classification algorithms' accuracy results reported in the recent literature.

6 Conclusions

This paper presented a comprehensive, comparative empirical study on the performance of all 65 classification algorithms available in Weka when applied to three MD data sets. Our aim was to explore the pros and cons of the classifiers and design a suitable 'roadmap' for applying classification to data sets that result from MD simulations of biomolecules.

We focused on classification error and excluded speed and memory requirement considerations from our evaluation, given that only classification error is technology independent; speed and memory requirements depend on the implementation and the platform used and are likely to improve along with improving hardware availability. We used default settings not only because these

are expected to be the best candidates for producing higher accuracy but mainly because 'over-engineering' settings could make results harder to generalise to other data sets. We found that Random Forest and Rotation Forest perform favourably, amongst the other algorithms, in terms of classification error.

Seven out of the leading nine classification algorithms are members of the meta group, whilst the other two belong to the trees group. These are, in ascending order of average error: Random Forest, Rotation Forest, Random Committee, Bagging, Classification via Regression, Ensemble Selection, END, Ordinal Class Classifier, and FT. Our findings are partly consistent with the literature. Notably, experiments reported elsewhere have also shown superior performance for Random Forest, Bagging, Rotation Forest (*Lee et al., 2005; de la Calleja and Fuentes, 2004; Gupta et al., 2010; Dehzangi et al., 2010*). However, SVM, NN, boosting, and Mix of Experts, reported to have demonstrated superior performance in the literature, did not do as well when applied to MD simulations data (*Das and Sengur, 2010; Das, 2010; Huang and Murphy, 2004*).

These results allow for an initial comparison amongst classification techniques when applied to MD data sets. In addition, the results provide insight into selecting the most appropriate classification algorithm for data sets that were derived from computer simulations. Having comprehensively evaluated classification algorithms in Weka, a promising area for future research would be to compare the performance of *clustering* algorithms, using a similar approach, on data sets with the same 'texture'.

As for the roadmap for selecting suitable classification algorithms for computer simulation data sets, one can consider algorithms which not only performed very well in this evaluation, but also demonstrated top performance in the literature, such as Random Forest, Bagging, and Rotation Forest. Algorithms with poor performance either in this evaluation or in similar evaluations in the literature do not make suitable candidates for selection.

We also note that groupings used in Weka do not lead to safe conclusions regarding algorithmic performance; for instance, both the top six and the bottom six algorithms are grouped in the meta team.

In the future, we plan to take in consideration metrics like scalability (runtime and memory requirements should not explode on large, high dimensional datasets), robustness (ability to detect outliers that are distant from the rest of the samples), and mixed data types.

Acknowledgements

The authors thank the anonymous reviewers for the constructive feedback, which helped us to improve the paper.

References

Adcock, S.A. and McCammon, J.A. (2006) 'Molecular dynamics: Survey of methods for simulating the activity of proteins', *Chem. Rev.*, Vol. 106, pp.1589–1615.

Aha, D.W., Kibler, D. and Albert, M.K. (1991) 'Instance-based learning algorithms', *Mach. Learn.*, Vol. 6, pp.37–66.

Andreopoulos, B., An, A., Wang, X. and Schroeder, M. (2009) 'A roadmap of clustering algorithms: finding a match for a biomedical application', *Briefings in Bioinformatics*, Vol. 10, pp.297–314.

Breiman, L. (1996) 'Bagging predictors', *Mach. Learn.*, Vol. 24, pp.123–140.

Breiman, L. (2001) 'Random forests', *Mach. Learn.*, Vol. 45, pp.5–32.

Browne, A., Hudson, B.D., Whitley, D.C., Ford, M.G. and Picton, P.D. (2004) 'Biological data mining with neural networks: implementation and application of a flexible decision tree extraction algorithm to genomic problem domains', *Neurocomputing*, Vol. 57, pp.275–293.

Chung, D. and Kim H. (2011) 'Robust classification ensemble method for microarray data', *Int. J. Data Min. Bioin.*, Vol. 5, pp.504–518.

Das, R. (2010) 'A comparison of multiple classification methods for diagnosis of Parkinson disease', *Expert Syst. Appl.*, Vol. 37, pp.1568–1572.

Das, R. and Sengur, A. (2010) 'Evaluation of ensemble methods for diagnosing of valvular heart disease', *Expert Syst. Appl.*, Vol. 37, pp.5110–5115.

de la Calleja, J. and Fuentes, O. (2004) 'Automated classification of galaxy images', *Lect. Notes Artif. Int.*, Vol. 3215, pp.411–418.

Dehzangi, A., Phon-Amnuaisuk, S., Manafi, M. and Soodabeh Safa, S. (2010) 'Using rotation forest for protein fold prediction problem: an empirical study', Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. *Lecture Notes in Computer Science,* Vol. 6023/2010, pp.217–227.

Demiroz, G. and Guvenir, H.A. (1997) 'Classification by voting feature intervals', *Machine Learning : Ecml-97*, Vol. 1224, pp.85–92.

Denaxas, S.C. and Tjortjis, C. (2008) 'Scoring and summarising gene product clusters using the Gene Ontology', *Int. J. Data Min. Bioin.*, Vol. 2, pp.216–235.

Dong, L., Frank, E. and Kramer, S. (2005) 'Ensembles of balanced nested dichotomies for multi-class problems', *Knowledge Discovery in Databases: Pkdd 2005*, Vol. 3721, pp.84–95.

Fayyad, U.M. and Irani, K.B. (1993) 'Multi-interval discretization of continuous-valued attributes for classification learning', *Proc. 13th Int'l Joint Conf. on Uncertainty in AI*, Morgan Kaufmann, pp.1022–1027.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P. (1996). 'From data mining to knowledge discovery: an overview', *Advances in Knowledge Discovery and Data Mining* American Association for Artificial Intelligence, pp.1–34.

Feher, M. and Schmidt, J.M. (2001) 'Metric and multidimensional scaling: Efficient tools for clustering molecular conformations', *J. Chem. Inf. Comp. Sci.*, Vol. 41, pp.346–353.

Fersht, A.R. and Daggett, V. (2002) 'Protein folding and unfolding at atomic resolution', *Cell*, Vol. 108, pp.573–582.

Gupta, R., Wikramasinghe, P., Bhattacharyya, A., Perez, F.A., Pal, S. and Davuluri, R.V. (2010) 'Annotation of gene promoters by integrative data-mining of ChIP-seq Pol-II enrichment data', *BMC Bioinformatics*, Vol. 11. p.S65

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009) 'The WEKA data mining software: an update', *ACM SIGKDD Explorations Newsletter*, Vol. 11, pp.10–18.

Han, G. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers. San Francisco.

Haoudi, A., and Bensmail, H. (2006) 'Bioinformatics and data mining in proteomics', *Expert Rev. Proteomic*, Vol. 3, pp.333–343.

Holte, R.C. (1993) 'Very Simple classification rules perform well on most commonly used datasets', *Mach. Learn.*, Vol. 11, pp.63–91.

Huang, K. and Murphy, R.F. (2004) 'Boosting accuracy of automated classification of fluorescence microscope images for location proteomics', *BMC Bioinformatics*, Vol. 5. p.78

Jain, A., Hegger, R. and Stock, G. (2010) 'Hidden complexity of protein free-energy landscapes revealed by principal component analysis by parts', *The Journal of Physical Chemistry Letters*, pp.2769–2773.

Janssen, F. and Furnkranz, J. (2011) 'Heuristic rule-based regression via dynamic reduction to classification', *Proc. 22nd Int'l Joint Conf. on Artificial Intelligence*, pp.1330–1335.

John, G. and Langley, P. (1995) 'Estimating continuous distributions in Bayesian classifiers', *Proc. 11th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp.338–345.

Kaburlasos, V.G., Athanasiadis, I.N. and Mitkas, P.A. (2007) 'Fuzzy lattice reasoning (FLR) classifier and its application for ambient ozone estimation', *Int. J. Approx. Reason.*, Vol. 45, pp.152–188.

Kanellopoulos, Y., Makris, C. and Tjortjis, C. (2007) 'An improved methodology on information distillation by mining program source code', *Data Knowl. Eng.*, Vol. 61, pp. 359–383.

Karplus, M. and Kuriyan, J. (2005) 'Molecular dynamics and protein function', *P. Natl. Acad. Sci. USA*, Vol. 102, pp. 6679–6685.

Keerthi, S.S. Shevade, S.K., Bhattacharyya, C. and Murthy, K.R.K. (2001) 'Improvements to Platt's SMO algorithm for SVM classifier design', *Neural Comput*, Vol. 13, pp.637–649.

Kohavi, R. (1995) 'The power of decision tables', *Machine Learning: Ecml-95.*, Vol. 912, pp.174–189.

Kononenko, I. (1993) 'Inductive and Bayesian learning in medical diagnosis', *Appl. Artif. Intell.*, Vol. 7, pp.317–337.

Lancashire, L.J., Lemetre, C. and Ball, G.R. (2009) 'An introduction to artificial neural networks in bioinformatics – application to complex microarray and mass spectrometry datasets in cancer studies', *Briefings in Bioinformatics*, Vol. 10, pp.315–329.

Landwehr, N., Hall, M. and Frank, E. (2005) 'Logistic model trees', *Mach. Learn.*, Vol. 59, pp.161–205.

Lecessie, S. and Vanhouwelingen, J.C. (1992) 'Ridge Estimators in Logistic-Regression', *Appl. Stat.-J. Roy. St. C*, Vol. 41, pp.191–201.

Lee, J.W., Lee, J.B., Park, M. and Song, S.H. (2005) 'An extensive comparison of recent classification tools applied to microarray data', *Comput. Stat. Data An.*, Vol. 48, pp.869–885.

Pei B., Rowe D.W., and Shin D.G. (2010) 'Learning Bayesian networks with integration of indirect prior knowledge', *Int. J. Data Min. Bioin.*, Vol. 4, pp.505–519.

Rao, F. and Karplus, M. (2010) 'Protein dynamics investigated by inherent structure analysis', *P. Natl. Acad. Sci. USA*, Vol. 107, pp.9152–9157.

Rodriguez, J.J., Kuncheva, L.I. and Alonso, C.J. (2006) 'Rotation forest: a new classifier ensemble method', *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, pp.1619–1630.

Stavrakoudis, A., Makropoulou, S., Tsikaris, V., Sakarellos-Daitsiotis, M., Sakarellos, C. and Demetropoulos, I.N. (2003) 'Computational screening of branched cyclic peptide motifs as potential enzyme mimetics', *J. Pept. Sci.*, Vol. 9, pp.145–155.

Tatsis, V.A., Stavrakoudis, A. and Demetropoulos, I.N. (2006) 'LysinebasedTrypsinActSite (LysTAS)': A configurational tool of the TINKER software to evaluate Lysine based branched cyclic peptides as potential chymotrypsin-mimetics', *Mol. Simulat.*, Vol. 32, pp.643–644.

Tatsis, V.A., Stavrakoudis, A. and Demetropoulos, I.N. (2008) 'Molecular Dynamics as a pattern recognition tool: an automated process detects peptides that preserve the 3D arrangement of Trypsin's Active Site', *Biophys. Chem.*, Vol. 133, pp.36–44.

van der Kamp, M.W., M.W. Shaw, K.E., Woods, C.J. and Mulholland A.J. (2008) 'Biomolecular simulation and modelling: status, progress and prospects', *J. R. Soc. Interface*, Vol. 5, pp.S173–S190.

Wallace, A.C., Laskowski, R.A. and Thornton, J.M. (1996) 'Derivation of 3D coordinate templates for searching structural databases: application to Ser-His-Asp catalytic triads in the serine proteinases and lipases', *Protein Science*, Vol. 5, pp.1001–1013.

Witten, I.H. and Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., Morgan Kaufmann. San Francisco.

Yona, G. and Kedem, K. (2005) 'The URMS-RMS hybrid algorithm for fast and sensitive local protein structure alignment', *J. Comput. Biol.*, Vol. 12, pp.12–32.

Table S1 The accuracy of various classification algorithms on several data sets found in literature are summarised in this table

	Dataset									
Classification algorithm	*Valvular heart disease (Das and Sengur, 2010)*	*Gene promoters (Gupta, et al., 2010)*	*SCOP database (Dehzangi, et al., 2010)*	*Parkinson's disease (Das, 2010)*	*Microarray data (Lee, et al., 2005)*	*SLF13 (2D DNA) (Huang and Murphy, 2004)*	*SLF8 (2D DNA) (Huang and Murphy, 2004)*	*SLF10 (3D DNA) (Huang and Murphy, 2004)*	*SLF14 (3D DNA) (Huang and Murphy, 2004)*	*Galaxy images (de la Calleja and Fuentes, 2004)*
(κ-NN/MLP/SVM) Single	91.1 – 95.1 –92.7									
(κ-NN/MLP/SVM) Bg	92.6 – 95.1 –95.1									
(κ-NN/MLP/SVM) ABs	92.6 – 95.9 – **98.4**									
(κ-NN-MLP-SVM) RS	94.2 – 95.1 –92.7									
Bg		**96.49**			89.00 - 79.00 - 94.00	88.90	87.20	89.40	82.20	
Bs					91.00 - 80.00 - 94.00					
LB		96.34			93.00 - 77.00 - 81.00					
RaF		96.46			**99.00 - 82.00 -99.00**					**91.29**
RoF		95.10	**62.4**							
NN				**92.9**	78.00 - 63.00 - 80.00	87.80	86.10	**95.30**	88.40	
DMNeural				84.3						
Rg				88.6						
DT				84.3						
CART										
SVM (rbf - exprbf)						89.40	88.10	95.20	**89.10**	
ABs						88.90	**88.20**	93.20	87.70	
MoE						**89.70**	87.00	92.20	83.8	
NB										80.68
C4.5										87.39

Abbreviations used: **κ-NN**: **κ**-Nearest Neighbour, **MLP**: network of simple neurons called perceptrons, **SVM**: Support Vector Machine, **LB**: LogitBoost, **RaF**: Random Forest, **RoF**: Rotation Forest, **NN**: Neural Network, **NB**: Naïve Bayes, **DT**: Decision Tree, **Bg**: Bagging, **Bs**: Boosting, **ABs**: AdaBoosting, **Rg**: Regression, **RS**: Random Subspaces, **MoE**: Mix-of-Experts