The CSI Journal on
Computer Science and Engineering
Vol. 20, No. 1, 2026
Pages 98-112
doi.org/10.22034/jcse.2025.564479.1070
Regular Paper

MetaRecall: An Ensemble Classifier with Dynamic Base Classifier Selection and Ordering

Behnam Azarshab[1], Sanaz Ardeshiri[2], Sanaz Rostami[3]

[1,2,3]Computer Engineering Department, School of Engineering, Persian Gulf University, Bushehr, 75168, Iran.

Abstract:
The main criteria to evaluate different classifiers are accuracy of classification, time taken to build the classifier and classification time as well as generalizability of the classifier. In this paper, we propose a novel ensemble classifier, named MetaRecall, which exploits confusion matrix to automatically select its base classifiers to increase accuracy of ultimate classification. To do so, a set of base classifiers as well as the training data set is fed to the algorithm as its input and the output of the algorithm is a ensemble classifier which contains a subset of the given base classifiers. Each involved classifier in MetaRecall corresponds to a class in the given dataset and its task is to classify instances of its corresponding class. To evaluate performance of MetaRecall, we do extensive experiments on different well-known benchmark datasets. In addition, we compare MetaRecall with the most commonly used previous ensemble classifiers. The results show that MetaRecall outperforms the previous classifiers in terms of accuracy and execution time in many cases.

Keywords: Classification, Ensemble Classifier, Multiple Classifier System, Confusion Matrix, Automatic Base Classifier Selection and Ordering.

1 Introduction

Classification is the process of assigning a specific instance to predefined categories, based on the respective instance properties. Classification currently is used as one of the most important techniques in machine learning. The goal is to find a model that identifies the fittest relationship between a set of input records and the class label. Figure 1 illustrates general approach to build a classification model [34]. An algorithm which implements classification is called a classifier or supervised learner.

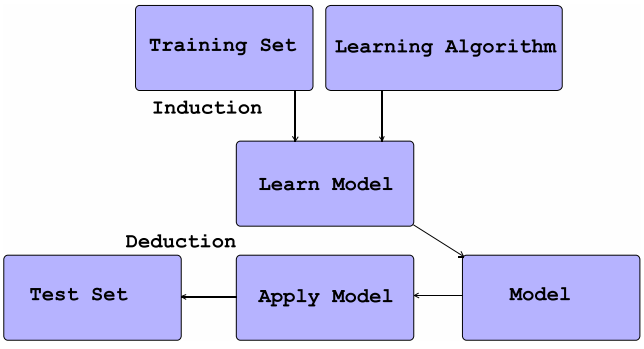

Figure 1: The general approach to build a Classification model.

Classifiers are generally trying to achieve three objectives; speed, accuracy and generalization. However, there is no classification method which on all data sets is better than all the other methods (No Free Lunch Theorem). So each method works better on a number of datasets. Some algorithms have more accuracy on some datasets and also

take more time. Achieving higher accuracy usually takes more time [34], [16]. Achieve higher accuracy has always been one of the most significant issues in classification. The acknowledgment that the classical approach to find the best individual classifier for a problem has some serious drawbacks, motivates the idea of Multiple Classifier System (MCS) that includes an ensemble of classifiers and a mechanism for parallel combination of outputs of the base classifiers [29], [3].

The idea of building ensemble (meta) classifiers is to produce a predictive model by integrating multiple classifiers [26], [36], [10], [8]. Generally ensemble methods are used for improving classification performance and accuracy [30].

In the rest of this section, we briefly review major classification techniques. The classifiers can be divided into two categories: base classifier methods and classifier combination methods (ensemble methods or meta-classifiers). In the following, we overview some major base classifiers.

In many applications, the relationship between attributes and class variable is non-deterministic. In the other words, the class label for the test records cannot be accurately predicted, even if attribute set is identical to the same training samples. Bayes classifiers is used for modeling probabilistic relationships between attribute sets and the class variable [34], [24], [22], [37].

Artificial neural networks and support vector machines are two samples of function based classifiers. Artificial neural networks are inspired by the human biological neural system. Recently, ensemble techniques have also been successfully applied to deep neural networks to enhance robustness [15]. An artificial neural network is composed of nodes and direct communications. In support vector machines, the goal is to find a hyper plane (decision boundary) that separate the data [20], [33], [25], [19], [18].

Another type of basic classifiers are lazy ones. In a lazy classifier the modeling process of the training data delays to the test data classification. One example of lazy classifiers is k nearest neighbors classifier.

In lazy learning methods, classification model is built locally for each dataset and can solve various problems and deal successfully with changes in the dataset. The disadvantage of lazy classifiers is need of large storage space to store all records of the training dataset. Lazy classifiers usually have good performance on datasets with few attributes. [2].

Rule-based classifiers process the training dataset and extract rules for building a classification model. This scheme is a technique for classifying records by using the "if ... then ..." rule sets. Each classification rule is expressed as follows: left-hand side of rule is called antecedent or precondition, and the right-hand side of the rule is called the rule consequent [12].

Another way to model relation between inputs and outputs of a dataset is tree-based classification. Decision trees are trees that classify samples by sorting them based on attribute values. Each node in a decision tree represents a feature in one sample that is supposed to be classified, and each branch represents a value of a node. The samples can be classified with starting from the root node and based on the attribute values are sorted [33].

Meta classifiers or ensemble classifiers aggregate predictions of all involved classifiers and improve classification accuracy in different applications [23], [8], [14], [32], [21], [39] . These classifiers use ensemble or classifier combination methods. Ensemble methods consist of a set of base classifiers in training data phase and classification done by voting on the predictions constructed by each base classifier. For example, boosting and bagging are ensemble classifiers [13], [4], [38], [27], [40].

In this paper, we propose a novel ensemble classifier based on extracting information from confusion matrix of the base classifiers with better classification accuracy in comparison of the most well-known ensemble classifiers.

The rest of paper is as follows: in section 2, we review the most related works to the proposed method. The proposed method is elaborated in section 3 and it is evaluated and compared with the most related works using well accepted benchmark datasets in section 4. Finally, we conclude the study in section 5.

2 Related work

The proposed method is an ensemble classifier. Therefore, we review the most well-known and prosperous ensemble classifiers. In addition, in our proposed method we order and select base classifiers dynamically according to train data, so the approaches which dynamically select or order their base classifiers are of our interest [9].

Bootstrap aggregating (Bagging) is designed to improve the accuracy and stability of classifiers. Bagging generates k new training sets, from a given dataset D, each of size less than size of D, by sampling from D uniformly and with replacement. So some instances may appear more than one time in a training set and some instances do not appear in any training set. Then we train the k classifiers with the k

generated training sets. Then any test instance is assigned to a class that has the maximum number of votes [4].

Another well known ensemble classifier is boosting. Boosting repeatedly applies adaptive changes to distribution of training instances and therefore base classifier focus on instances that are hard to classify. Boosting method, assigns weight to each instance and weight of each training instance at the end of each boosting round may change adaptively.

In Boosting we deal with two problems; first, update weights of the training samples at the end of each boosting round and second mix predictions of each classifier at the end of classification operation [13]. Modern boosting implementations, such as XGBoost, have further optimized this process for scalability and performance [7].

Random forest is another ensemble method that combines the prediction of multiple decision tree classifiers. All the trees in random forest are produced by an independent set of random vectors produced from a fixed probability distribution. Random forests are used in bagging in the case that randomness injected into model building phase and select instances randomly from the original training set, with replacement [5].

Rotation forest is another ensemble classifier based on feature extraction. In this method, training data is created by splitting the feature set into subsets and applying Principal Component Analysis (PCA) to each subset. To maintain the variable information in the data, it keeps all principal components. Thus, to create the new features for a base classifier, K axis rotations are accomplished [28].

Dagging generates some disjoint folds of data and feeds each fold of data to a copy of the base classifier. Majority vote makes the final prediction because base classifiers put their prediction into the vote ensemble classifier [35].

The next ensemble method is stacked generalization. The goal of the stacked generalization method is creating a global method that uses a high-level model for combining the prediction of lower-level models to achieve higher accuracy. Stacked generalization minimizes the generalization error rate of one or more generalizers. Stacked generalization according to offered training set works by deducing the generalizer(s) biases [38].

In all the previous ensemble classifiers in the first step, we should select some base classifiers to participate in the ensemble classifiers without any prior knowlege about performance compatiblity of them. Despite the previous work our proposed method (MetaRecall) automatically select the best base classifiers from a large set of classifiers without significant time overhead.

In [31], the authors proposed a dynamic overproduce-and-choose strategy to select base classifiers of an ensemble classifier. They combine optimization and dynamic selection in a two-level selection phase to allow the selection of the most confident subset of classifiers to label each test sample individually. The optimization level is intended to generate a population of highly accurate candidate classifier ensembles, while the dynamic selection level applies measures of confidence to reveal the candidate ensemble with the highest degree of confidence in the current decision.

In [6], the authors proposed a new dynamic fusion method named Localized Generalization Error Model Fusion Method (LFM) for multiple classifier systems. They provide a generalization error bound for unseen samples located within neighborhoods of testing samples. Base classifiers with lower generalization error bounds are assigned higher weights. LFM estimates the local competence of base classifiers not only using the information of training error but also the sensitivity of classifier outputs.

In [11], authors proposed a hierarchical and parallel branch-and-bound ensemble selection algorithm. It realizes layer-wise refinement of the selected ensemble solutions, which enables the classification performance of the selected ensembles to be improved in a layer-by-layer manner.

Despite this method, in our proposed method, after selecting base classifiers, we assign a classifier to each class and prioritize them to have the most correct results. Therefore, our proposed method differes from these methods in terms of base classifier selection, ordering and base classifier usage.

3 MetaRecall algorithm

As we mentioned before, there is no classifier which outperforms all the others in all situations and for all datasets. Even for a fixed given dataset, different classifiers may predict different classes in the dataset with different accuracy. The main idea of MetaRecall algorithm is to take advantages of classifiers and avoid disadvantages of them for any given dataset. To do so, to predict each class in the dataset a corresponding classifier is used (so the method is an ensemble classifier).

Each class in the dataset has a corresponding classifier, but a classifier may correspond to more than one classes. Assignment of a classifier to a class is done based on its recall. After choosing a classifier for each class, we de-

fine precedence of the classifiers based on their accuracy and then based on their precedence we sort them. The final ensemble classifier involves selected classifiers sorted by their precedence.

At the utilization time of the resulted ensemble classifier (MetaRecall), each input is examined against the first involved base classifier, if the output indicates that the given input belongs to its corresponding class, the process terminates and the resulted output is returned as the recognized class for the input by MetaRecall. Otherwise (if the involved classifier recognizes the input as a member of a class that is not correspond to it), the next involved classifier examines the input. If none of them recognize the input as their corresponding classes, a voting mechanism is employed for the final decision.

Before elaborating the proposed method, we should review the concepts of recall, precision and f-measure as we use them to develop the model. To do so, we should consider a confusion matrix of a classifier for a given problem. Evaluation of the performance of a classification model is based on the counts of testing instances correctly and incorrectly predicted by the model. The tabulated illustration of these counts is called confusion matrix [34]. In a confusion matrix, the predicted classes are displayed at the top of the matrix, and the observed classes down the left side. Each cell contains a number showing how many cases that were actually of the given observed class were assigned by the model to the given predicted class. For problems with two classes, the confusion matrix is represented as a table (Table 1).

Table 1: A typical confusion matrix to a two class problem.

	PREDICTED CLASS	
ACTUAL CLASS	Class= Positive	Class= Negative
Class=Positive	a	b
Class=Negative	c	d

The Precision, Recall and F-measure Concepts for problem of Table 1 is calculated using the following equations.

$$\text{Precision}(p) = \frac{a}{a+c}$$

$$\text{Recall}(r) = \frac{a}{a+b}$$

$$\text{F} - \text{measure(F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

$$\text{Accuracy} = \frac{a+d}{a+b+c+d}$$

In problems with more than two classes, a class is considered as positive class and the other classes as the negative class. We use the same concept in the proposed method to compute precision, recall and f-measure of the classifiers.

As the above equations show, higher value of recall, means fewer instances of the target class is classified as an instance of the other classes. On the other side, higher value of precision indicates that fewer instances of the other classes classified as an instance of the target class. F-measure is a combination of the precision and recall and that is a harmonic mean of both of them.

3.1 MetaRecall construction

As we mentioned before, the general idea of the proposed method (MetaRecall) is to assign a classifier to each class of a given dataset. To construct the final ensemble classifier, we should select some classifiers as the corresponding classifiers of classes of the dataset. To do so, we select a collection of classifiers as candidate set and refine it to reach the final classifiers involved in the resulted ensemble classifier. Figure 2 and Figure 3 show the general steps of the method.

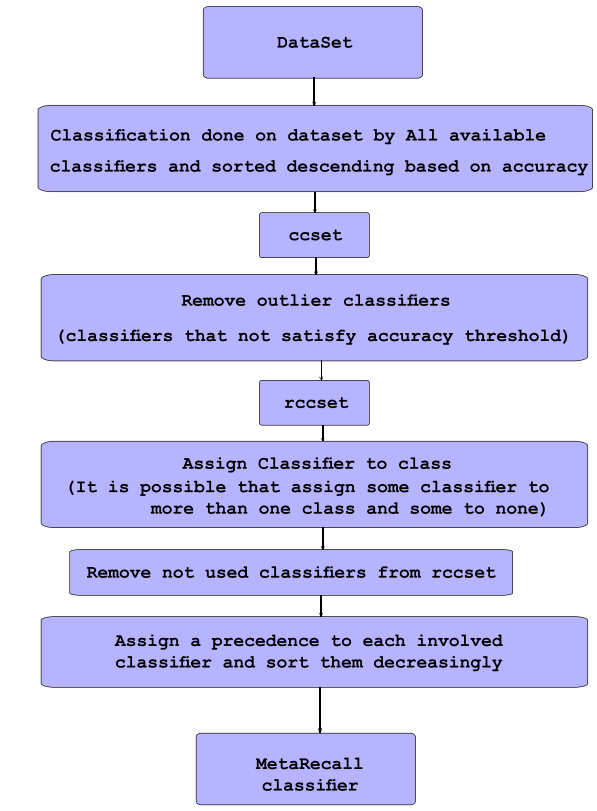

Figure 2: The general steps of MetaRecall construction algorithm. In this step, we assign a classifier to each class of a given dataset.

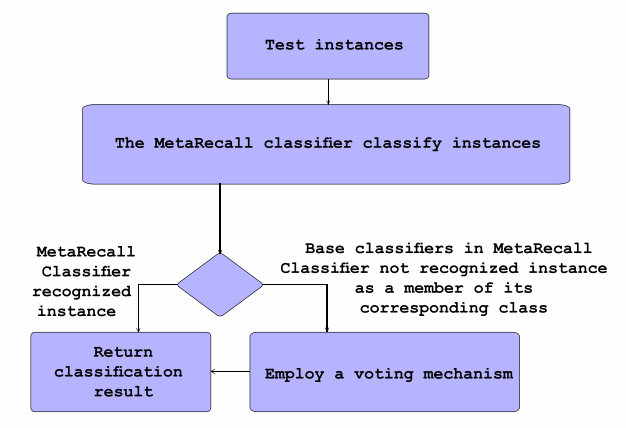

Figure 3: The general steps of MetaRecall utilization. In this setp MetaRecall base classifiers recognize the input as a member of its corresponding class or employ a voting mechanism to classify instances.

The criterion for choosing a classifier to be in the initial candidate set is its accuracy in classification of the given dataset. To do so, we examine some available classifiers against the dataset and sort the result descending based on accuracy as an initial candidate set.

We refine the initial candidate set by removing outlier classifiers based on their accuracy for the given dataset. By an outlier, we mean a classifier which its accuracy is less than a predefined τ threshold from the most accurate classifier in the initial candidate set. The reason to remove outlier classifiers is that they dominate other classifiers and mislead the final ensemble classifier (experiments confirm the same). In subsection 4.6, we seek for and show the optimal value of τ.

Formally, suppose that $C_1, C_2, C_3, \ldots, C_p$ are p available tuned distinct classifiers and the given data set, name it D, has n classes $c_1, c_2, c_3, \ldots, c_n$ (it is expected that $p > n$). The initial classifier candidate set ($CCset$), is built by selecting classifiers sorted descending based on accuracy for D as follows:

$$CCset = \underset{\text{Accuracy-On-D}}{\textit{Sorted-Descending-Based-On-Accuracy}}\{C_1, C_2, C_3, \ldots, C_p\}$$

The refined candidate ($RCCset$) set is obtained by eliminating outlier classifiers. So, the number of classifiers in the refined candidate set is less than or equal to the number of classes in the given data set.

$$RCCset = CCset - \{\textit{outlier-classifiers}\} = \{C_1, C_2, C_3, \ldots, C_m\}$$

The next step is to find a corresponding classifier for each class of the dataset from $RCCset$. As it has been mentioned before, a classifier may correspond to more than one class of the given dataset. A classifier in the refined candidate set may be assigned to more than one class and on the other side there may be some classifiers that are not assigned to any class. The later classifiers will be removed. In the rest of this section we describe the method in more detail.

If $P_{1c_1}, \ldots, P_{1c_n}, P_{2c_1}, \ldots, P_{2c_n}, \ldots, P_{mc_1}, \ldots, P_{mc_n}$ are the precisions of classifiers 1 to m (in refined candidate set) for classes 1 to n, respectively, $R_{1c_1}, \ldots, R_{1c_n}, R_{2c_1}, \ldots, R_{2c_n}, \ldots, R_{mc_1}, \ldots, R_{mc_n}$ are the recalls of classifiers 1 to m for classes 1 to n, respectively, and $F_{1c_1}, \ldots, F_{1c_n}, F_{2c_1}, \ldots, F_{2c_n}$, $F_{mc_1}, \ldots, F_{mc_n}$ are the F-measures of classifiers 1 to m for classes 1 to n, respectively, then C_i is the best classifier for class c, where $R_{ic} = \max\{R_{1c}, R_{2c}, R_{3c}, \ldots, R_{mc}\}$.

If $\max\{R_{1c}, R_{2c}, R_{3c}, \ldots, R_{mc}\} = \{R_{1c}, R_{2c}, \ldots, R_{kc}\}$ (It means that the recall amount of k classifiers has the same maximum value and equal and $2 \leq k \leq m$), then the best classifier for class c is C_j Where$F_{jc} = \max\{F_{1c}, F_{2c}, F_{3c}, \ldots, F_{kc}\}$.

If $\max\{F_{1c}, F_{2c}, \ldots, F_{kc}\} = \{F_{1c}, F_{2c}, F_{3c}, \ldots, F_{lc}\}$ ($2 \leq l \leq k$), then the classifier with the least runtime is chosen. The processes repeats for all classes of the dataset. Figure 4 illustrates the process and algorithm 1 shows the pseudo code of the method. The classifiers that do not assigned to any class are removed from the refined candidate set, also if a classifier corresponds to more than one class, it is repeated accordingly, and in this way the initial version of the final ensemble classifier is built. Suppose that the final ensemble classifier is $MetaClassifier = \{C_1^{'}, C_2^{'}, C_3^{'}, \ldots, C_n^{'}\}$ (n is number of classes and the classifiers may not be distinct). The final step to construct the ensemble classifier (MetaRecall) is to assign precedence to each involved base classifier and sort them decreasingly. To do so, we sort classifiers (if a classifier corresponds to more than one class, the same number of instances of that classifier appears in MetaRecall) involved in the final ensemble classifier based on their recall for their corresponding classes. Therefore, the final ensemble classifier is a decreasing sorted list of classifiers based on their recalls for the corresponded classes. Reason of sorting classifier based on the recall values is avoiding error propagation.

Algorithms 1 and 2 show detail of model construction in the proposed method. In algorithm 1, we create $ccset$ which contains p classifiers with the highest accuracy on data set D. Then we initialize $rccset$ which includes all $ccset$ clas-

Algorithm 1 Find corresponding classifiers

```
Dataset D
int m=Number of classifiers
int n=Number of classes in D
Set ccset={}
ccset=Descending sort classifiers by accuracy on D
Set rccset= ccset-(classifiers that do not satisfy accuracy
difference threshold)
Set MetaClassifier
for i = 0 to n − 1 do
  Set temp=Determine max recalls of rccset classifiers
  for Class[i]
  if temp.length ≥2 then
    Set tempF=Determine max F-measures in temp for
    Class[i]
    if tempF.length ≥2 then
      The fastest classifier in tempF add to MetaClassi-
      fier {classifier for class c[i] is the fastest classifier
      in tempF}
    else
      tempF[0] add to MetaClassifier{Classifier in
      tempF is corresponding classifier for class c[i]}
    end if
  else
    MetaClassifier.add(temp[0]){Classifier in temp is
    corresponding classifier for class c[i]}
  end if
end for
return MetaClassifier
```

Algorithm 2 Find order of corresponding classifiers.

```
Set findOrderOfCorrespondingClassifiers
Set CorrespondingClassi-
fiers=findCorrespondingClassifiers
Set res = {}
while CorrespondingClassifier is not Empty do
  max=Determine maximum recall of all classifies in cor-
  respondingClassifier set
  Add a classifier to res that max belongs to and remove
  from CorrespondingClassifier
end while
return res
```

sifiers except ones that do not satisfy the accuracy threshold. In for loop, we assign the classifiers to the classes. In this step, a classifier that has a maximum recall value of the specific class is assigned as corresponding classifier for that class.

If more than one classifier has a maximum recall value of that class, F-measure value is used to determine corresponding class and if more than one class has maximum value of the specific class, then minimum running time, determines the correspond classifier.

Algorithm 2 determines the execution order of corresponding classifiers. The execution order is important. Corresponding classifiers according to maximum amount of their recall values are ordered in descending order, which means that a classifier that has the highest value among the other recall values is the first classifier and so on.

3.2 MetaRecall utilization

At utilization time, as an input is fed to MetaRecall, it is given to the first involved classifier, if the classifier recognizes it as a member of its corresponding class, then the same class is returned as the result of MetaRecall. Otherwise, it is tested against the second involved classifier. The process continues until a classifier recognizes the input as a member of its corresponding class or we reach the end of the list of classifiers. In the latter case, we employ a voting mechanism as follows:

For each class c_i, we define a weight W_{c_i} as the sum of recalls (for classc_i) of classifiers that recognize the input as a member of class c_i. So,

$$W_{c_i} = \sum_{\substack{\text{classifier } j \text{ recognizes} \\ \text{input as member of } c_i}} R_{jc_i} \tag{1}$$

Where R_{jc_i} is recall value of classifier j for class i. The class with maximum weight is returned as output of MetaRecall. Algorithm 3 presents the pseudo code of the classification process. We should note that feeding inputs to the classifiers can be done in parallel to speed up the process. In the following section, we study the both serial and parallel versions of the algorithm.

4 Experiments

4.1 Experiment set up

The algorithms (MetaRecall algorithm and the other algorithms) are compared on five data sets. All data sets are

Algorithm 3 MetaRecall utilization algorithm

```
Set MetaClassifier=findCorrespondingClassifiers
double classifyInstance(Instance instance)
for i = 0 to MetaClassifier.length − 1 do
  if MetaClassifier[i] recognizes instance as a member of
  its corresponding class then
    return corresponding class
  end if
end for
WeightsArray=array [0.. numOfClasses-1]
for i = 0 to numOfClasses − 1 do
  for j = 0 to MetaClassifier.length − 1 do
    WeightsArray[i]=WeightsArray[i]+(recall of Meta-
    Classifier[j] for Class[i])
  end for
end for
return Class[k] where WeightsArray[k] is Max
```

Table 2: The test platform and hardware.

Component	Specification
Operating System	Microsoft Windows 7 Professional
System Type	x64-based PC
Processor	Intel(R) Core(TM) i7 CPU Q 740 @ 1.73GHz, 1734 MHz, 4 Core(s), 8 Logical
RAM	4.00 GB

available and downloadable [1].

Table 2, shows hardware configuration of the experiment platform. We employed weka-3-6-8 [17] as the base of our software platform. We extend weka by importing its libraries to our program. We use 10-fold cross-validation method with weka API to evaluate all the algorithms in the experiments.

4.2 Experimental Results

In the rest of this section, we compare performance of the algorithms in different datasets.

4.2.1 Anneal Data Set

This data set donated by David Sterling and Wray Buntine [1]. This dataset has includes 798 instances and 38 attributes. Figures 4, 9 and 14 show the results of experiments on the anneal dataset [1].

4.2.2 Car dataset

The car dataset is taken from a simple hierarchical decision model that originally was developed for DEX (expert systems for decision-making) [1]. This data set contains 1728 instances and 6 attributes and 4 classes. Figures 5, 10 and 15 show the results of experiments on the car dataset.

4.2.3 molecular biology promoters dataset

This dataset has been developed to help evaluate a "hybrid" learning algorithm ("KBANN") that uses examples to inductively refine preexisting knowledge. Using a "leave-one-out" methodology, the following errors were produced by various ML algorithms. [1]. This data set contains 106 instances and 59 attributes and 2 classes. Figures 6, 11 and 16 show the results of experiments on the Molecular Biology Promoters dataset.

4.2.4 spect train dataset

The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature pattern was created for each patient. The pattern was further processed to obtain 22 binary feature patterns. [1]. This data set contains 267 instances and 23 attributes and 2 classes. Figures 7, 12 and 17 show the results of experiments on the Spect Train dataset.

4.2.5 ionosphere dataset

This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts [1]. This dataset has includes 351 instances and 34 attributes. Figures 8, 13 and 18 show the results of experiments on the ionosphere dataset [1].

4.3 Effectiveness of MetaRecall

In this section, we evaluate effectiveness of the ensemble classifier, MetaRecall, in terms of accuracy. Figures 4,5,6,7 and 8 show accuracy comparison of MetaRecall in comparison with its employed base classifiers. The reported results are mean values of 25 runs. As it can be seen, by using MetaRecall, we can increase accuracy of classification at the expense of some time cost.

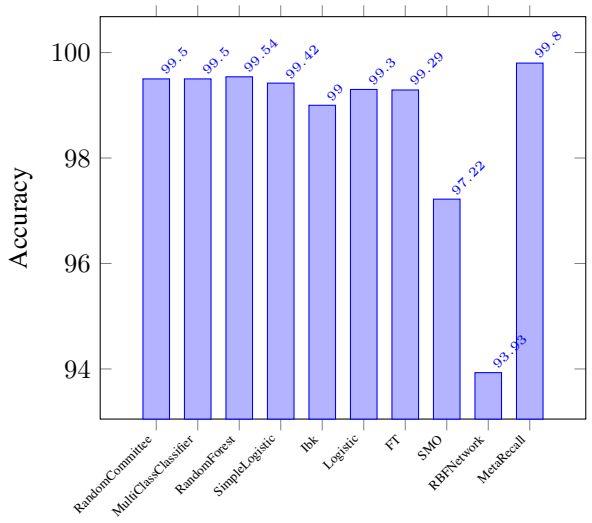

Figure 4: Comparison of MetaRecall with base classifiers on anneal dataset

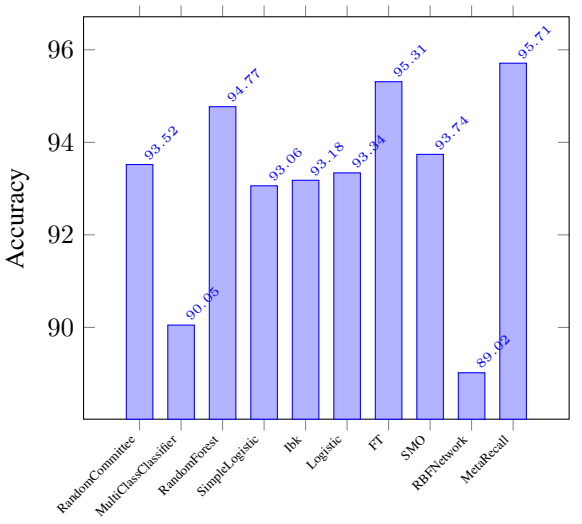

Figure 5: Comparison of MetaRecall with base classifiers on car dataset

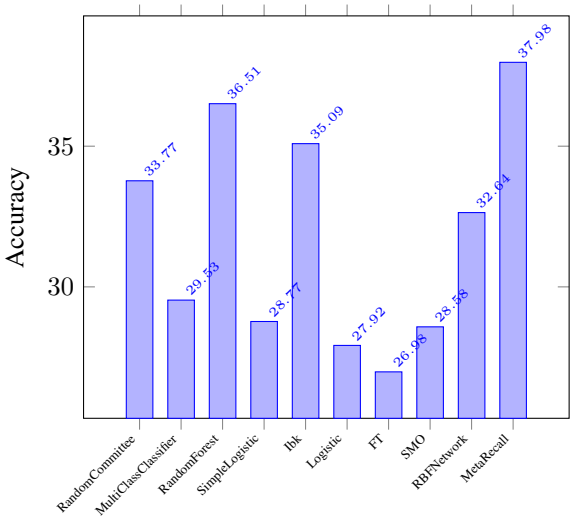

Figure 6: Comparison of MetaRecall with base classifiers on molecular biology promoters dataset

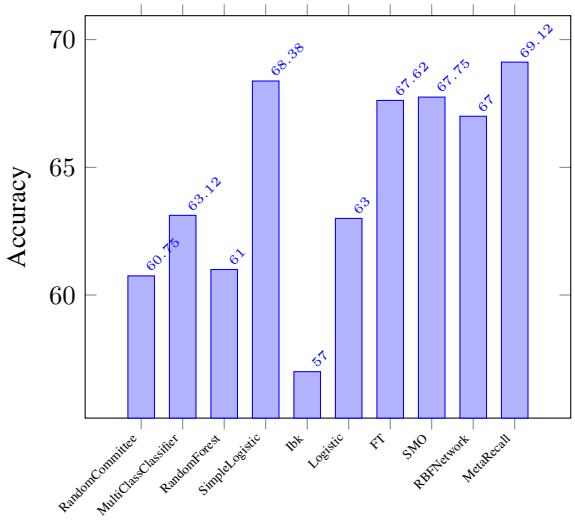

Figure 7: Comparison of MetaRecall with base classifiers on spect train dataset

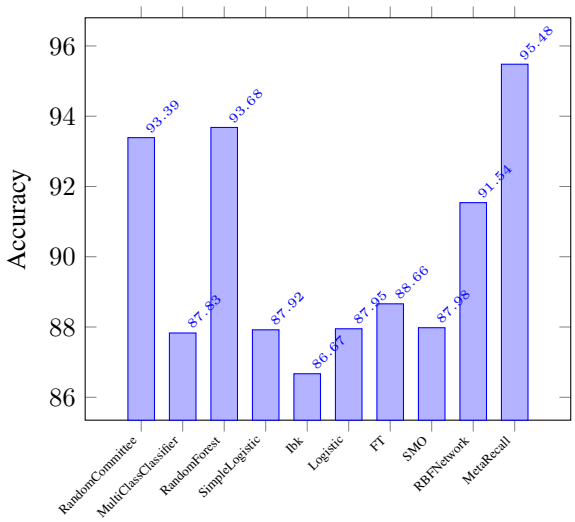

Figure 8: Comparison of MetaRecall with base classifiers on ionosphere dataset

4.4 Hypothesis testing of effectiveness of MetaRecall

In this subsection, with using hypothesis t-test, we evaluate effectiveness of MetaRecall in terms of accuracy in comparison with provided base classifiers...So, MetaRecall is separately compared with each base classifier. In these tests, the null hypothesis which is often denoted H_0, demonstrates that MetaRecall does not improve accuracy of classification in comparison with the base classifiers, and the alternative hypothesis which is often denoted H_a, demonstrates considerable difference between them. According to experimental results, MetaRecall improves accuracy of classifications sig-

nificantly.

The pooled estimate of standard deviation denoted S_{AB} is calculated by Eq 2. In this equation, n_A and n_B which are equal to 25, show the number of evaluations of MetaRecall and another compared algorithm, respectively. Also, S_A and S_B are standard deviations of MetaRecall and another compared classifier. Then, Eq 3 calculates the experimental t-value (t_{exp}). Finally, t_{exp} should be compared with the critical value ($t_{critical} = 2.407$). If t_{exp} is greater than $t_{critical}$, then H_0 is rejected; otherwise, it is retained.

Table 3 shows the results of this hypothesis test. We compare MetaRecall with its initial base classifier, before automatic selection among them, (RandomCommittee, MultiClassClassifier, RandomForest, SimpleLogistic, Ibk, Logistic, FT, SMO, RBFNetwork and MetaRecall) on Anneal, Car, Ionosphere, Molecular Biology Promoters and Spect Train datasets.

As it can be seen, in all cases (except MetaRecall/Logistic on Car dataset), H_0 is rejected and H_a is accepted.

$$S_{AB} = \sqrt{\frac{(n_A - 1)S_A^2 + (n_B - 1)S_B^2}{n_A + n_B - 2}} \tag{2}$$

$$t_{exp} = \frac{|\overline{f(x_A)} - \overline{f(x_B)}|}{S_{AB}\sqrt{\frac{1}{n_A} + \frac{1}{n_B}}} \tag{3}$$

	MR/RandomCommittee	MR/MultiClass	MR/RandomForest	MR/SimpleLogistic	MR/IBK	MR/Logistic	MR/FT	MR/SMO
Anneal	10.009	10.009	2.777	3.852	4.252	4.872	3.664	5.252
Car	3.802	9.825	4.283	10.102	7.100	5.915	**1.981**	4.920
Ionosphere	3.862	14.136	4.033	7.020	11.797	6.541	8.350	7.957
Molecular Biology	5.624	11.289	4.346	2.484	2.979	7.295	2.789	5.283
Spect Train	4.177	2.994	9.410	3.117	3.868	3.288	4.320	4.973

Table 3: Hypothesis testing of MetaRecall with base classifiers

In this part of paper, we compare performance of MetaRecall with some well-known ensemble classifiers. Voting, Stacking, Grading, MultiSchema and StackingC are those compare width MetaRecall on some datasets. We feed RandomComitee, Multiclassifier, RandomForest, SimpleLogistic, IBK, Logistic, FT and SMO with their default parameters as base classifiers of the ensemble classifiers (Voting, Stacking, Grading, MultiSchema, StackingC and MetaRecall). As we mentioned before, MetaRecall automatically selects a subset of fed base classifiers to be used in its classification processes depending on the training dataset. Tables 4, 5, 6, 7 and 8 show the selected base classifiers by MetaRecall for anneal, car, ionosphere, molecular biology promoters and spect train datasets respectly.

Class	Corresponding Base Classifier	Percentage Correct
0	RandomCommittee	99.777
1	RandomCommittee	99.777
2	RandomCommittee	99.777
3	RandomCommittee	99.777
4	RandomCommittee	99.777
5	MultiClassClassifier	99.666

Table 4: Selected base classifiers by MetaRecall on Anneal dataset.

Class	Corresponding Base Classifier	Percentage Correct
0	IBK	93.345
1	RandomForest	94.734
2	FT	95.602
3	FT	95.602

Table 5: Selected base classifiers by MetaRecall on Car dataset.

Class	Corresponding Base Classifier	Percentage Correct
0	RandomCommittee	94.017
1	RandomForest	93.732

Table 6: Selected base classifiers by MetaRecall on Ionosphere dataset.

Class	Corresponding Base Classifier	Percentage Correct
0	IBK	35.849
1	IBK	35.849
2	IBK	35.849
3	IBK	35.849

Table 7: Selected base classifiers by MetaRecall on molecular biology promoters dataset.

Class	Corresponding Base Classifier	Percentage Correct
0	SimpleLogistic	72.500
1	SimpleLogistic	72.500

Table 8: Selected base classifiers by MetaRecall on spect train dataset.

Figures 9, 10, 11, 12 and 13 show accuracy of the methods on data sets. Also, Figures 14, 15, 16, 17 and 18 show time performance of the methods on the same datasets. On anneal dataset, MetaRecall has the best accuracy (Figure 9) and the

third time cost after voting and MultiScheme (Figure 14). On car dataset, MetaRecall has the third accuracy performance (Figure 10) and third time cost (Figure 15). MetaRecall has the best accuracy on Molecular Biology Promoters dataset (Figure 11) and second time cost among the other classifiers (Figure 16). On Spect-train dataset, MetaRecall has the best accuracy (Figure 12) and the third time cost (Figure 17). Finally, on ionosphere dataset, again MetaRecall has the best accuracy (Figure 13) and in time cost performance has third place (Figure 18). As it can be seen, MetaRecall has the best accuracy on four datasets out of five. In addition, in most cases it has the second or third time cost among six well-known ensemble classifiers.

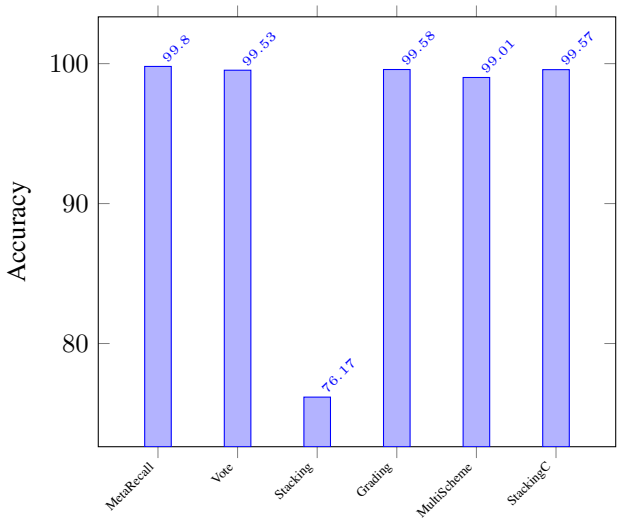

Figure 9: Comparison of MetaRecall with other ensemble classifiers on anneal dataset

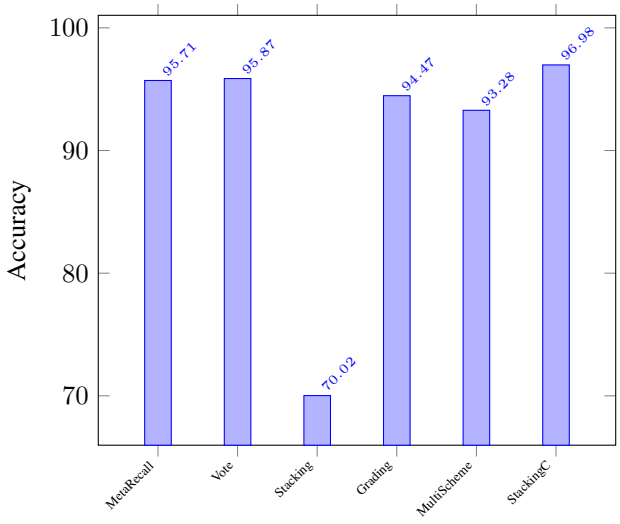

Figure 10: Comparison of MetaRecall with other ensemble classifiers on car dataset

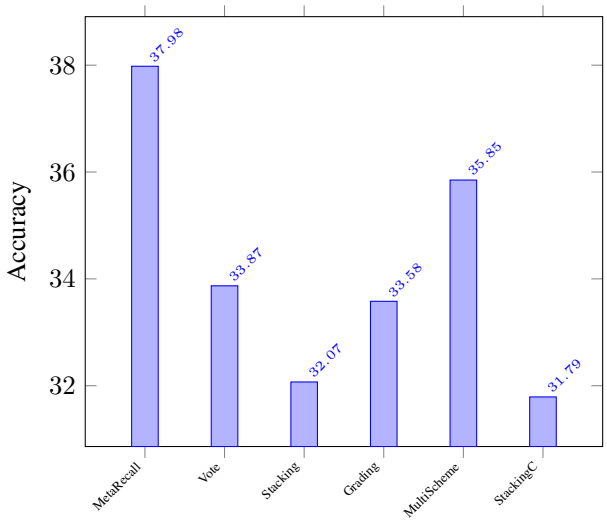

Figure 11: Comparison of MetaRecall with other ensemble classifiers on molecular biology promoters dataset

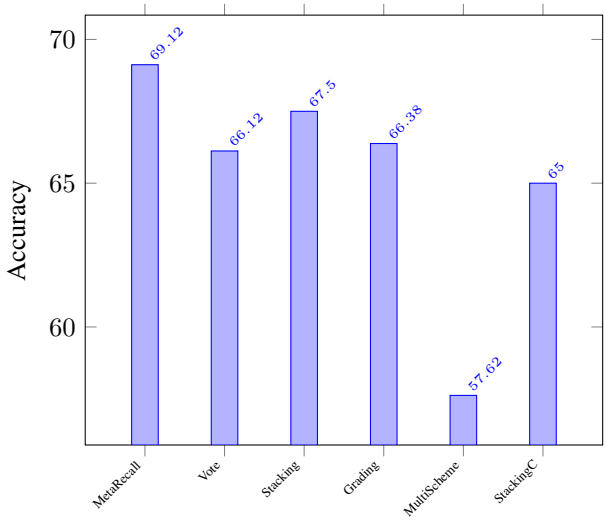

Figure 12: Comparison of MetaRecall with other ensemble classifiers on spect train dataset

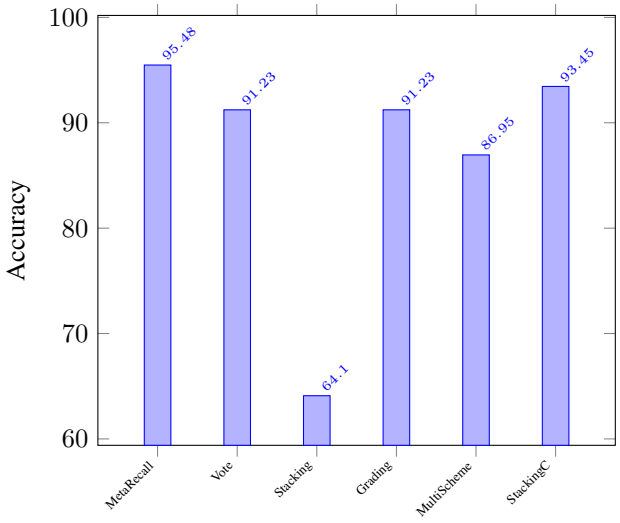

Figure 13: Comparison of MetaRecall with other ensemble classifiers on ionosphere dataset

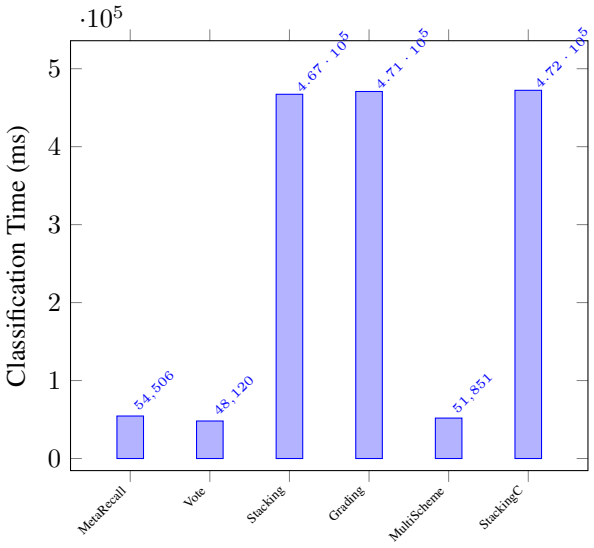

Figure 14: Compare time performance of MetaRecall with other ensemble classifiers on anneal dataset

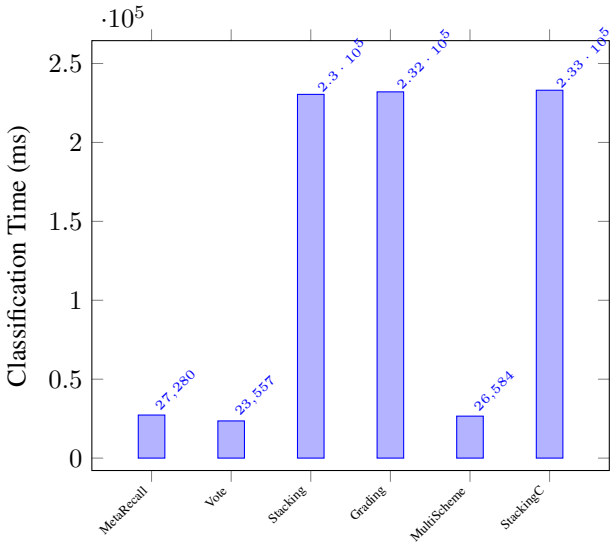

Figure 15: Compare time performance of MetaRecall with other ensemble classifiers on car dataset

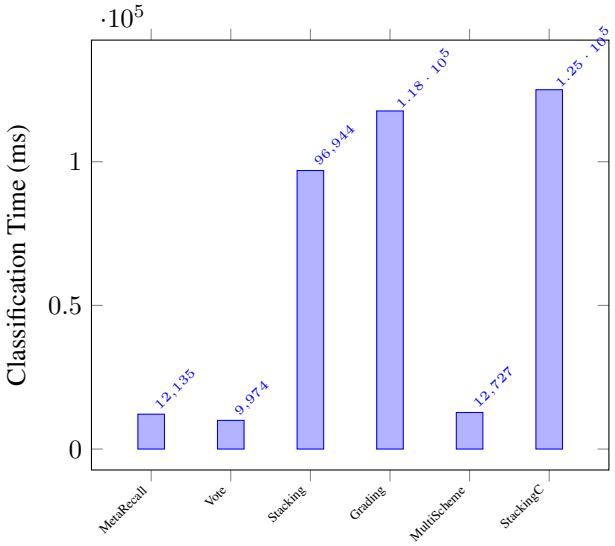

Figure 16: Compare time performance of MetaRecall with other ensemble classifiers on molecular biology promoters dataset

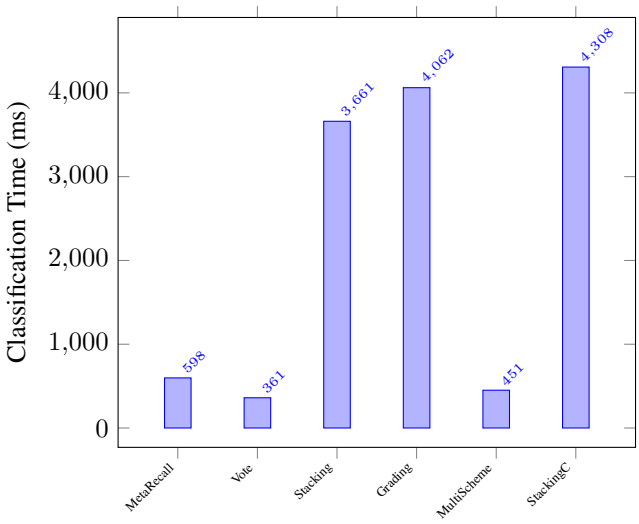

Figure 17: Compare time performance of MetaRecall with other ensemble classifiers on spect train dataset

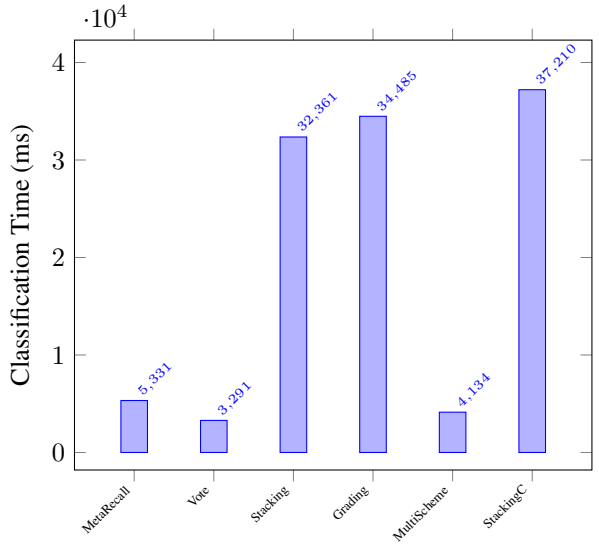

Figure 18: Compare time performance of MetaRecall with other ensemble classifiers on ionosphere dataset

As the results show, MetaRecall shows the highest accuracy among the examined classifiers on different datasets. We should note that the based classifiers used in the ensemble classifiers are selected by algorithm 1 which is part of MetaRecall. So, it may seem that the chosen base classifiers are biased to MetaRecall. But the other ensemble classifiers have no algorithm to choose the base classifiers and the chosen base classifiers are those with the highest accuracy. So, it is not expected that the studied ensemble classifiers show better accuracy using some other base classifiers.

4.5 Hypothesis testing of comparison of MetaRecall and the other ensemble classifiers

In this subsection, with using hypothesis t-test, we evaluate comparison results of MetaRecall with the other ensemble classifiers in terms of accuracy. So, MetaRecall is separately compared with each ensemble classifier. In these tests, the null hypothesis witch is often denoted H_0, demonstrates that MetaRecall and the ensemble classifier compared with it have no difference in accuracy, and the alternative hypothesis witch is often denoted H_a, demonstrates considerable difference between them.

Again we use Eq 2 and Eq 3 ($t_{critical}$=2.407) to do hypothesis t-test.

Table 9 shows the results of this hypothesis test. We compare MetaRecall with some well-known ensemble classifiers (Voting, Stacking, Grading, MultiScheme, StackingC) on Anneal, Car, Ionosphere, Molecular Biology Promoters and Spect Train datasets. As it can be seen, in all cases H_0 is rejected and H_a is accepted.

	MR/Vote	MR/Stacking	MR/Grading	MR/MultiScheme	MR/StackingC
Anneal	3.550	14.108	2.106	3.197	9.256
Car	**0.690**	106.774	2.961	4.347	4.150
Ionosphere	5.138	18.129	3.616	17.696	2.787
Molecular Biology	7.211	13.508	4.520	5.414	3.267
Spect Train	2.642	5.339	3.232	19.097	2.840

Table 9: Hypothesis testing of MetaRecall with the other ensemble classifiers

4.6 Effect of τ on performance of MetaRecall

As it is mentioned in the previous section, a parameter that influences base classifier selection is the accuracy difference threshold (τ).

Figure 19 shows the effect of τ on the accuracy of MetaRecall. As it can be seen, by increasing the value of τ, the accuracy of the classifier generally decreases. The best value of τ, for the given datasets, is in the interval $[0, 3]$ and around 2.4012.

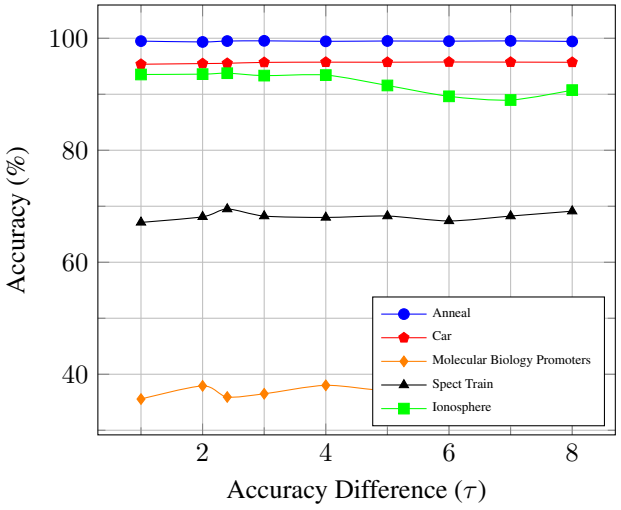

Figure 19: Accuracy decreases with increasing accuracy difference (τ) of MetaRecall corresponding classifier.

The reason that increasing τ causes a decrement in the accuracy of the classifier is that although the recall values of some corresponding classifiers for some classes are better and can predict instances of the corresponding classes more accurately, the weakness of prediction in the other classes leads to a negative impact on the accuracy of MetaRecall.

5 Conclusion

In this paper, we presented a novel meta classifier (ensemble classifier). The idea of MetaRecall is that each class of a given dataset has a corresponding base classifier in the

ensemble classifier. MetaRecall has a systematic and automated approach to choose its base classifiers and assigning them to classes of the given dataset. We compared the MetaRecall with different ensemble classifiers on different benchmark datasets. Results show that by using MetaRecall, we can increase accuracy of classification in comparison with base classifiers. In addition, in the most datasets examined MetaRecall has better classification accuracy in comparison of the most well-known ensemble classifiers with reasonable time cost.

References

[1] A. Asuncion, D.N.: UCI machine learning repository (2007). URL `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`

[2] Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6**(1), 37–66 (1991)

[3] Biglari, M., A., S., Hassanpour, H.: Pattern Anal Applic (2017). DOI 10.1007/s10044-017-0637-4

[4] Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)

[5] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[6] Chan, P.P.K., Yeung, D.S., Ng, W.W.Y., Lin, C.M., Liu, J.N.K.: Dynamic fusion method using localized generalization error model. information sciences **217**, 1–20 (2012). DOI 10.1016/j.ins.2012.06.026. URL `http://dx.doi.org/10.1016/j.ins.2012.06.026`

[7] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)

[8] Chen, Z., Lin, T., Chen, R., Xie, Y., Xu, H.: Creating diversity in ensembles using synthetic neighborhoods of training samples. Applied Intelligence **47**(2), 570–583 (2017). DOI 10.1007/s10489-017-0922-3

[9] Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C.: Dynamic classifier selection: Recent advances and perspectives. Information Fusion **41**, 195–216 (2018)

[10] Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C.: Prototype selection for dynamic classifier and ensemble selection. Neural Computing and Applications **29**(2), 447–457 (2018). DOI 10.1007/s00521-016-2458-6

[11] Dai, Q., Yao, C.: A hierarchical and parallel branch-and-bound ensemble selection algorithm. Applied Intelligence **46**(1), 45–61 (2017). DOI 10.1007/s10489-016-0817-8

[12] Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization pp. 144–151 (1998)

[13] Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)

[14] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets. Information Sciences **354**, 178 – 196 (2016)

[15] Ganaie, M.A., Hu, M., Malik, A.K., Tanveer, M., Suganthan, P.N.: Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence **115**, 105,151 (2022)

[16] Gorunescu, F.: Data mining concepts, models and techniques, vol. 12. Springer, Berlin (2011)

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)

[18] Hastie, T., Tibshirani, R.: Classification by pairwise coupling. The annals of statistics **26**(2), 451–471 (1998)

[19] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to platt's smo algorithm for svm classifier design. Neural Comput. **13**(3), 637–649 (2001)

[20] Le Cessie, S., Van Houwelingen, J.: Ridge estimators in logistic regression. Applied Statistics **41**(1), 191–201 (1992)

[21] Li, Y., Bai, C., Reddy, C.K.: A distributed ensemble approach for mining healthcare data under privacy constraints. Information Sciences **330**, 245 – 259 (2016). SI Visual Info Communication

[22] McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization, vol. 752, pp. 41–48. Citeseer (1998)

[23] Meo, R., Bachar, D., Ienco, D.: Lode: A distance-based classifier built on ensembles of positive and negative observations. Pattern Recognition **45**(4), 1409–1425 (2012)

[24] Mera, D., Fernández-Delgado, M., Cotos, J.M., Viqueira, J.R.R., Barro, S.: Comparison of a massive and diverse collection of ensembles and other classifiers for oil spill detection in sar satellite images. Neural Computing and Applications **28**(1), 1101–1117 (2017). DOI 10.1007/s00521-016-2415-4

[25] Platt, J.C.: Advances in kernel methods pp. 185–208 (1999)

[26] Rahman, A., Verma, B.: Ensemble classifier generation using non-uniform layered clustering and genetic algorithm. Know.-Based Syst. **43**, 30–42 (2013)

[27] Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning **87**(3), 357–380 (2012)

[28] Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**(10), 1619–1630 (2006)

[29] Roli, F.: Multiple Classifier Systems, pp. 981–986. Springer US, Boston, MA (2009). DOI 10.1007/978-0-387-73003-5_148. URL https://doi.org/10.1007/978-0-387-73003-5_148

[30] Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(4), e1249 (2018)

[31] Santos, E.M.D., Sabourin, R., Maupin, P.: A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. Pattern Recognition **41**(10), 2993 – 3009 (2008)

[32] Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R.Y., Liu, F.: Dynamic clustering forest: An ensemble framework to efficiently classify textual data stream with concept drift. Information Sciences **357**, 125 – 143 (2016)

[33] Sumner, M., Frank, E., Hall, M.: Speeding up logistic model tree induction. Knowledge Discovery in Databases: PKDD 2005 **3721**, 675–683 (2005)

[34] Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005)

[35] Ting, K.M., Witten, I.H.: Stacking bagged and dagged models. pp. 367–375 (1997)

[36] Wang, Z., Jie, W., Chen, S., Gao, D.: Random projection ensemble learning with multiple empirical kernels. Know.-Based Syst. **37**, 388–393 (2013)

[37] Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive bayes: Aggregating one-dependence estimators. Machine Learning **58**(1), 5–24 (2005)

[38] Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)

[39] Zhou, L., Fujita, H.: Posterior probability based ensemble strategy using optimizing decision directed acyclic graph for multi-class classification. Information Sciences pp. – (2017)

[40] Zou, P.C., Wang, J., Chen, S., Chen, H.: Bagging-like metric learning for support vector regression. Knowl.-Based Syst. **65**, 21–30 (2014)

Behnam Azarshab received his BSc and MSc degrees in software engineering.

Email: b.azarshab@email.com

Sanaz Ardeshiri is currently a student at Persian Gulf University.

Email: sanaz.ardeshiri18@gmail.com

Sanaz Rostami is currently a student at Persian Gulf University.

Email: sanstarrs@gmail.com

Paper Handling Data:
Corresponding author: Sanaz Ardeshiri
Computer Engineering Department, School of Engineering, Persian Gulf University of Bushehr, Bushehr 75168, Iran.