Zhenbo Xie 216856239

Assignment 2 Classification

2023.3.20

1 Model introduction

In this paper, six classification models from weka are used.

C4.5: C4.5 is the algorithm used in the classification problem for machine learning and data mining. Its goal is supervised learning: given a dataset in which each element can be described by a set of attribute values, each element belongs to one of a series of mutually exclusive categories. the goal of C4.5 is to find, through learning, a mapping relation from attribute values to categories, and this mapping can be used to classify entities with new categories that are unknown. And the decision tree algorithm that uses the information gain rate as a metric for the current optimal decision attribute. The advantage is that the resulting classification rules are easy to understand and have a high accuracy rate; the disadvantage is that the process of constructing the tree requires several sequential scans and sorting of the data set, thus leading to inefficiency of the algorithm. In addition, C4.5 is only suitable for data sets that can reside in memory, so the program cannot run when the training set is too large to fit in memory, and the learning process is overly concerned with how to improve the correct classification of the training data, resulting in the construction of overly complex decision trees and thus overfitting the model. The corresponding code in weka is weka.classifiers.trees.J48

REPTree: Fast decision tree learner. Uses information gain rate to build decision trees and prunes them using error reduction pruning (with backward fit). Unlike C4.5, it only sorts the values of numerical attributes once, so its generation is faster than C4.5. The corresponding code in weka is weka.classifiers.trees.REPTree

Naïve Bayesian: Naïve Bayesian (NBC) is a method based on Bayes' theorem and the assumption of mutual independence of the feature conditions, which learns the joint probability distribution from the input to the output based on the learned model, input X to find the output Y that maximizes the posterior probability. Bayesian algorithm assumes that the data set attributes are independent of each other, so the logic of the algorithm is very simple and the algorithm is more stable, which means that when the relationship between the data set attributes is relatively independent, the plain Bayesian classification algorithm will have better results; then the condition of attribute independence is also the shortcoming of the plain Bayesian classifier. The independence of dataset attributes is difficult to be satisfied in many cases, because the attributes of the dataset are often interrelated with each other, and if this problem occurs in the classification process, it will lead to a significant reduction in the classification effect. The code corresponding to this in weka is weka.classifiers.trees.NaiveBayes.

Bayes Network: A Bayesian network is a directed acyclic graph consisting of nodes representing variables and directed edges connecting these nodes. The nodes represent random variables, and the directed edges between the nodes represent the interrelationships between the nodes, expressing the strength of the relationships in terms of conditional probabilities and the information in terms of prior probabilities if there is no parent node. Therefore, Bayesian networks have powerful ability to handle uncertainty problems. Bayesian networks express the correlations between individual information elements with conditional probabilities and are able to learn and reason under limited, incomplete, and uncertain information conditions. The code corresponding to it in weka is weka.classifiers.bayes.BayesNet.

k-Nearest Neighbor: k-Nearest Neighbor (KNN) is a basic classification and regression method, which belongs to supervised learning (with labels). The k-Nearest Neighbor in classification problems, where the input is each attribute of the instance and the output is the class of the instance, can take multiple classes. It is based on the principle of obedience to majority rule. In simple terms, it means that given a dataset in which the category of instances is determined, the k instances nearest to the target instance are found in the training dataset, and if most of these k instances belong to a certain category, the target instance is classified into that category. The advantage is that it is simple to implement, and when the sample distribution is regular and the number of samples is balanced, it can get a good classification effect; the disadvantage is that when the samples are not balanced, i.e., the number of samples in one category is much more than that in another category, which leads to classification errors when the samples of large volume classes occupy a larger number, although the training samples with small distance from the samples are small. The corresponding code in weka is weka.classifiers.lazy.IBK.

Neural Networks: Artificial neural network is a supervised learning algorithm, which tries to build a mathematical model by simulating the processing mechanism of the human brain's nervous system for complex information, and is developed from the neuron model. Its advantages are high accuracy of classification; strong parallel distributed processing ability, strong distributed storage and learning ability, strong robustness and fault tolerance to noisy nerves, and full approximation of complex nonlinear relationships; with the function of associative memory. The disadvantage is that the neural network requires a large number of parameters, such as the initial values of network topology, weights and thresholds; the learning process between observations cannot be observed, and the output results are difficult to interpret, which will affect the credibility and acceptability of the results; the learning time is too long, and may not even achieve the purpose of learning. The corresponding code in weka is weka.classifiers.functions.MutilayerPerceptron.

2 Introduction of datasets and data cleaning

Five UCI datasets as well as Kaggle datasets are used in this paper.

Ecoli: a protein localization dataset from the UCI database. There are Sequence Name columns representing the database login number, 7 numerical columns representing the features, and class columns representing the data categories. And it is easy to know that the first column will not play any role in the classification process, so the first column is selected for the weka.filter.unsupervised.attribute.Remove operation in the data cleaning process; in order to ensure that the data size will not have an impact on the prediction results, the other seven columns except the class column are removed using the weka.filter.unsupervised.attribute. Normalize. The distribution of each column in the data set before and after data cleaning is shown in Figure 2-1.

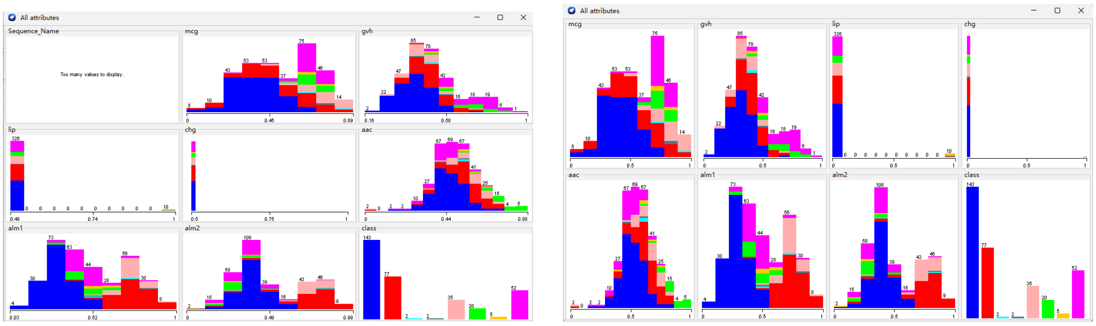

Figure 2-1

(a) Before data cleaning (b) After data cleaning

Figure 2-1 Distribution of columns before and after data cleaning for the Ecoli dataset

Iris: Iris data set from UCI database. Its feature column contains four numerical features and one class column representing the data category. To ensure that the data size does not affect the prediction results, the other 4 columns except the class column are normalized using weka.filter.unsupervised.attribute.Normalize. The distribution of each column in the data set before and after data cleaning is shown in Figure 2-2.

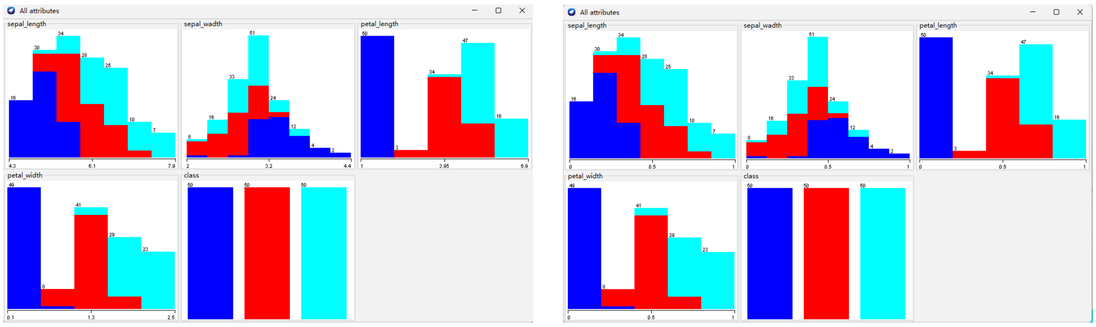

Figure 2-2

(a) Before data cleaning (b) After data cleaning

Figure 2-2 Distribution of columns before and after data cleaning for the Iris dataset

Glass: is the dataset from the UCI database. Its characteristic columns include 1 column of Index, 9 columns of numerical data and the last column of class. And it is easy to know that the first column will not play any role in the classification process, so the first column is selected for the weka.filter.unsupervised.attribute.Remove operation in the data cleaning process; in order to ensure that the data size will not have an impact on the prediction results, the other 9 columns except for the class column are cleaned using weka. Normalize; finally, since the class column is numeric, we run the weka.filters.unsupersivised.attribute.NumericToNominal filter to process the data. NumericToNominal filter. The distribution of the columns of the dataset before and after data cleaning is shown in Figure 2-3.

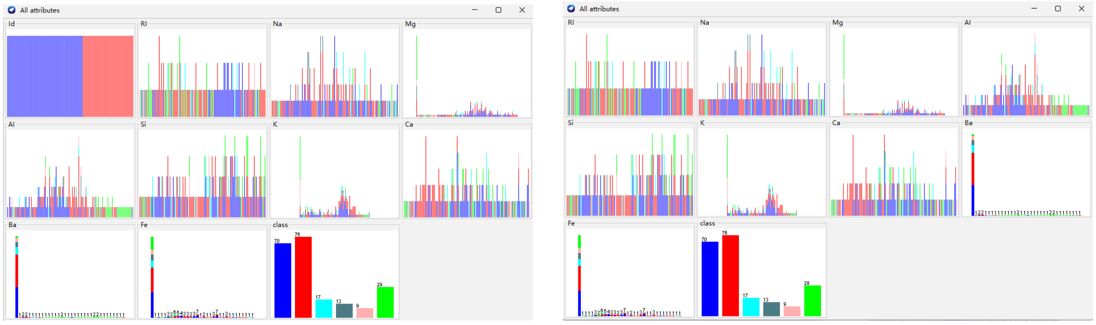

Figure 2-3

(a)Before data cleaning (b) After data cleaning

Figure 2-3 Column distribution of Glass dataset before and after data cleaning

Yeast: is the dataset from the UCI database. The feature column contains 1 column of Sequence Name, 8 columns of continuous data and the last column of class. And it is easy to know that the first column will not play any role in the classification process, so the first column is selected for the weka.filter.unsupervised.attribute.Remove operation in the data cleaning process; in order to ensure that the data size will not have an impact on the prediction results, the other 8 columns except the class column are cleaned using the weka. Filter.unsupervised.attribute.Normalize. The distribution of each column in the data set before and after data cleaning is shown in Figure 2-4.

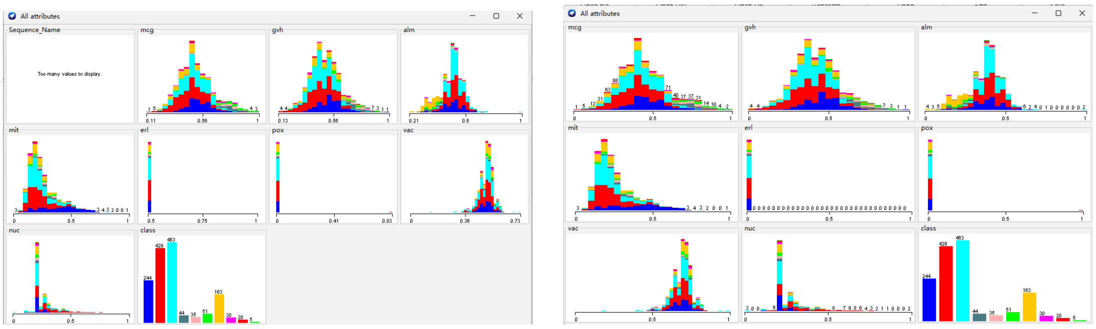

Figure 2-4

(a)Before data cleaning (b) After data cleaning

Figure 2-4 Distribution of columns before and after data cleaning for Yeast dataset

Gait: is the dataset from the UCI database. Its feature column contains 321 continuous features and a class column representing the data category. In order to ensure that the data size does not affect the prediction results, the other 321 columns except the class column are normalized using weka.filter.unsupervised.attribute.Normalize; finally, since the class column is numeric, we run weka.filters. NumericToNominal. The partial column distribution of the dataset before and after data cleaning is shown in Figure 2-5.

。

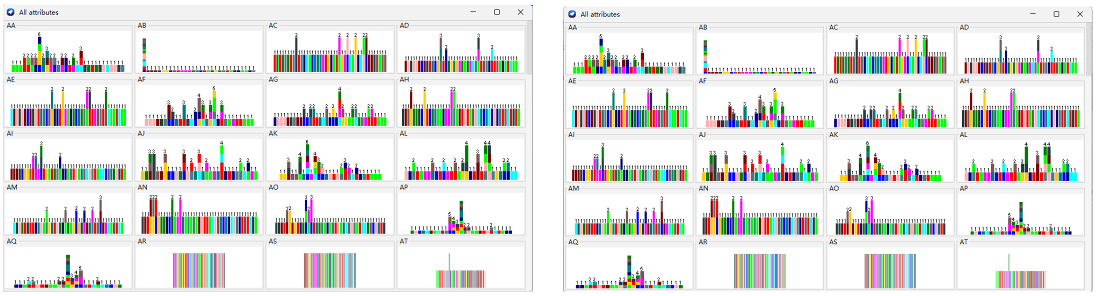

Figure 2-5

(a)Before data cleaning (b) After data cleaning

Figure 2-5 Distribution of columns in the Gait dataset before and after data cleaning

Grade: The dataset from the Kaggle competition. Its feature columns contain 17 discrete features, 13 continuous features, and 1 class column representing the data category. In order to ensure that the data size does not affect the prediction results, the 13 continuous feature columns are normalized using weka.filter.unsupervised.attribute.Normalize. The partial column distribution of the dataset before and after data cleaning is shown in Figure 2-6

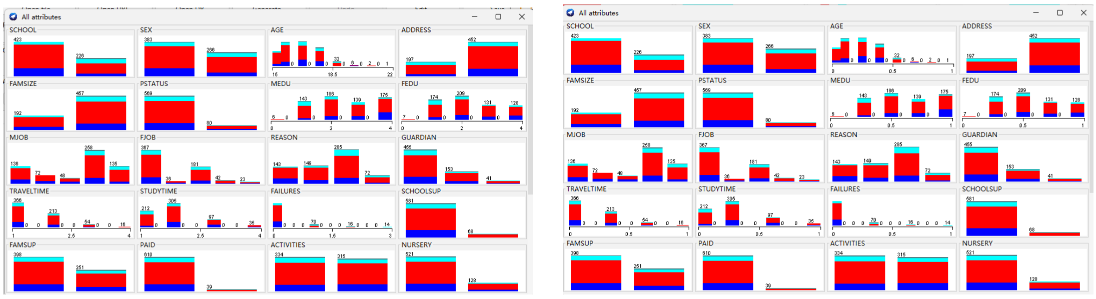

Figure 2-5

(a)Before data cleaning (b) After data cleaning

Figure 2-6 Distribution of columns before and after data cleaning of Grade data set.

3 Model training

3.1 C4.5

The ten-fold cross-validation was selected to train the model using C4.5 decision tree on the data set after data cleaning mentioned in Section 2, and the results obtained are shown in Table 3-1.

Table 3-1 C4.5 algorithm training results

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0327	0.1291	0.073555	0.273796
Iris	0.035	0.1586	0.078705	0.336353
Glass	0.2297	0.3466	0.931249	0.988888
Yeast	0.1013	0.2673	0.651445	0.959189
Gait	0.1178	0.2662	0.992842	1.085609

Grade	0.2197	0.4198	0.831393	1.156335
error_average	0.1227	0.2646	0.593198167	0.800028333

3.2 REPTree

The ten-fold cross-validation was selected to train the model using C4.5 decision tree on the cleaned data set mentioned in Section 2, and the results obtained are shown in Table 3-2.

Table 3-2 Training results of REPTree algorithm

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0635	0.1934	0.346953	0.640918
Iris	0.0563	0.1936	0.126749	0.410599
Glass	0.2237	0.3655	0.907769	1.042654
Yeast	0.1087	0.2465	0.698832	0.884562
Gait	0.1192	0.2464	1.004604	1.00497
Grade	0.2453	0.3639	0.92808	1.002451
error_average	0.136116667	0.268216667	0.668831167	0.831025667

3.3 Naïve Bayesian

Ten-fold cross validation was selected to train the model using C4.5 decision tree on the cleaned data set mentioned in Section 2, and the results obtained are shown in Table 3-3.

Table 3-3 Training results of Naïve Bayesian algorithm

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0429	0.1639	0.23461	0.543314
Iris	0.0361	0.1593	0.081248	0.338031

Glass	0.1816	0.3214	0.736499	0.917033
Yeast	0.1049	0.2398	0.674354	0.860213
Gait	0.033	0.1377	0.277852	0.561642
Grade	0.24	0.3915	0.907908	1.078581
error_average	0.106416667	0.2356	0.485411833	0.716469

3.4 Bayes Network

The ten-fold cross validation was selected to train the model using C4.5 decision tree on the cleaned data set mentioned in Section 2, and the results obtained are shown in Tables 3-4.

Table 3-4 Training results of Bayes Network algorithm

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.054	0.1768	0.295216	0.58612
Iris	0.0454	0.1828	0.102111	0.387793
Glass	0.1817	0.3334	0.75868	0.951099
Yeast	0.1102	0.2381	0.708256	0.854186
Gait	0.0282	0.1386	0.237683	0.565109
Grade	0.2356	0.3617	0.8914	0.996338
error_average	0.109183333	0.238566667	0.498891	0.723440833

3.5 k-Nearest Neighbor

Ten-fold cross-validation was chosen to train the model using C4.5 decision tree on the cleaned data set mentioned in Section 2, and the results obtained are shown in Tables 3-5.

Table 3-5 Training results of k-Nearest Neighbor algorithm

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0535	0.2189	0.29238	0.725574
Iris	0.0399	0.1747	0.89763	0.370695
Glass	0.1675	0.3178	0.679099	0.90671
Yeast	0.096	0.3078	0.61749	1.10423
Gait	0.0678	0.1949	0.57141	0.794891
Grade	0.2321	0.4791	0.878168	1.319849
error_average	0.109466667	0.2822	0.6560295	0.870324833

3.6 Neural Networks

The ten-fold cross-validation was selected to train the model on the data set after data cleaning mentioned in Section 2 using C4.5 decision tree, and the empty positions in the table are the data set with training crash, which is guessed to be too large data dimension, resulting in the failure of neural network training. The obtained results are shown in Table 3-6.

Table 3-6 Training results of Neural Networks algorithm

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0483	0.1704	0.263973	0.564742
Iris	0.0327	0.1291	0.073555	0.273796
Glass	0.2507	0.4814	1.016456	1.373414
Yeast	0.1017	0.2385	0.653926	0.855591
Gait	/	/	/	/
Grade	0.218	0.4243	0.824811	1.168889
error_average	0.13028	0.28874	0.5665442	0.8472864

3.7 Summary

As shown in Table 3-7, the average error of several algorithms, of which the bolded plain Bayesian algorithm and Bayesian network algorithm are the top two algorithms with better performance: both are Bayesian-based methods, and the reason for the good model training effect at this time is that the correlation between features has been shaved off in the process of selecting the features of the data set, which meets the requirements of the model. The underlined algorithms are the poorly performing algorithms: distance-based kNN and neural network. The kNN algorithm only considers the distance between samples when estimating.

Table 3-7 Average error of the algorithm

	C4.5	REPTree	Naïve Bayesian	Bayes Network	k-Nearest Neighbor	Neural Networks
MAE	0.1227	0.1361	**0.1064**	**0.1092**	0.1095	0.1302
RMAE	0.2646	0.2682	**0.2356**	**0.2386**	0.2822	0.2887
RAE	0.5932	0.6688	**0.4854**	**0.4989**	0.6560	0.5665
RRSE	0.8000	0.8310	**0.7165**	**0.7234**	0.8703	0.8472

4 Parameter tuning

For the plain Bayesian algorithm, the parameters that can be adjusted are

useKernelEstimator: use the kernel estimator for numerical attributes instead of the normal distribution.

numDecimalPlaces: the number of decimal places to use for the numbers in the output model.

batchSize: the preferred number of instances to process if a batch prediction is being performed. More or fewer instances can be provided, but this gives the implementation the opportunity to specify the preferred batch size.

Debug: if set to true, the classifier may output additional information to the console.

displayModelInOldFormat: Outputs the model using the old format. The old format is better when there are many class values. The new format is better when there are fewer classes and more attributes.

doNotCheckCapabilities: if set, do not check classifier functions before building the classifier (use with care to reduce runtime).

useSupervisoredDiscretization: converts numeric attributes to nominal attributes using supervised discretization.

Therefore useKernelEstimator will play a role in the accuracy of the model, adjust it to True and the result is shown in Table 4-1

Table 4-1 Run results after using the kernel estimator

	mean absolute error	Root mean aquared error	relative absolute error	root relative squared error
Ecoli	0.0459	0.158	0.250727	0.523586
Iris	0.0388	0.1527	0.087366	0.323828
Glass	0.1816	0.3214	0.736499	0.917033
Yeast	0.1057	0.2335	0.679582	0.837743
Gait	0.033	0.1377	0.277852	0.561642
Grade	0.2174	0.372	0.822463	1.024618
error_average	0.103733	0.229217	0.475748	0.698075

For the Bayesian network algorithm, the parameters that can be adjusted are

numDecimalPlaces: the number of decimal places used to output the numbers in the model.

batchSize - the preferred number of instances to process if a batch prediction is being performed. More or fewer instances can be provided, but this gives the implementation the opportunity to specify the preferred batch size.

estimator: selects the estimator algorithm used to find the conditional probability table of the Bayesian network.

debug: if set to true, the classifier may output additional information to the console.

searchAlgorithm: select the method used to search the network structure.

doNotCheckCapabilities: if set, classifier functions are not checked before building the classifier (use with care to reduce runtime).

BIFFile: set the file name in BIF XML format. The Bayesian network learned from the data can be compared with the Bayesian network represented by the BIF file. The statistics computed are o.a. the number of missing arcs and extra arcs.

useADTree: When using ADTree (a data structure used to increase the counting speed, not to be confused with the classifier of the same name), the learning time usually decreases. However, memory problems may occur because ADTrees take up a lot of memory. Turning this option off will make the structure learning algorithm slower and run with less memory. By default, ADTrees will be used.

One of the adjustable parameters of the estimator is alpha, which is used to estimate the probability table and can be interpreted as the initial count of each value. The estimators included in it are

BayesNetEstimator: base class for estimating the conditional probability table of the Bayesian network after learning the structure.

BMAEstimator: uses Bayesian Model Averaging (BMA) to estimate the conditional probability table of a Bayesian network.

MultiNomialBMAEstimator: multivariate BMA estimator

SimpleEstimator: Used to estimate the conditional probability table of a Bayesian network after learning the structure of that network. Estimates probabilities directly from the data.

The searchAlgorithm corresponds to different search algorithms corresponding to different adjustable parameters, and the algorithms are GeneticSearch, HillClimber, K2, LAGDHillClimber, RepeatedHillClimber, SimulatedAnnealing TabuSearch, and TAN.

And it is easy to know that the above parameters only improve the training efficiency of the algorithm and do not contribute to any of the model accuracy.