CS 4641B Assignment1

Shengrui Lyu GTID: 903392423

Datasets:

Car-evaluation:

The dataset is for evaluating a car from some of its attributes. There are 4 classes:

Unacceptable(unacc), Acceptable(acc), Good(good), Very Good(vgood).

And there several attributes:

Buying (buying price): vhigh (very high), high, med, low.

Maint (price of the maintenance): vhigh (very high), high, med, low.

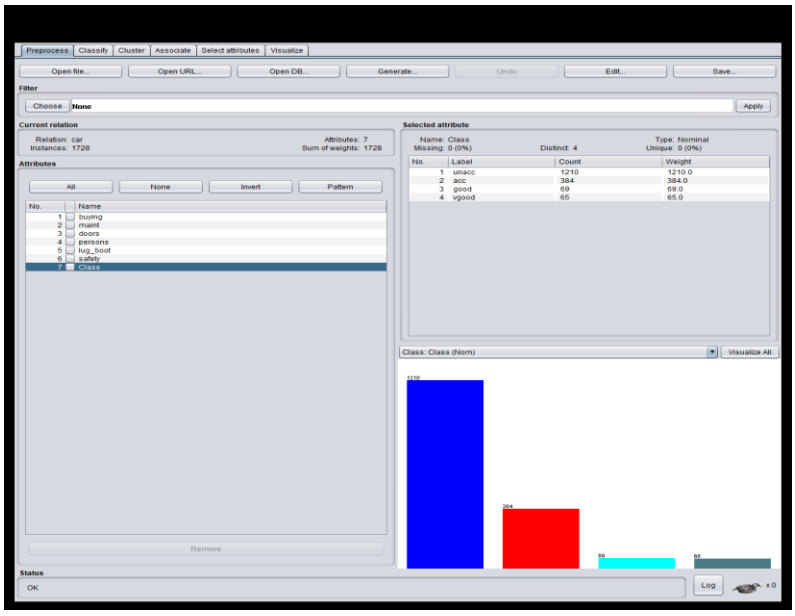
Doors (number of doors): 2, 3, 4, 5more (5 or more).

Persons (capacity in terms of persons to carry): 2, 4, more.

lug_boot (the size of luggage boot): small, med, big.

Safety (estimated safety of the car): low, med, high

And here is a distribution of the class values:



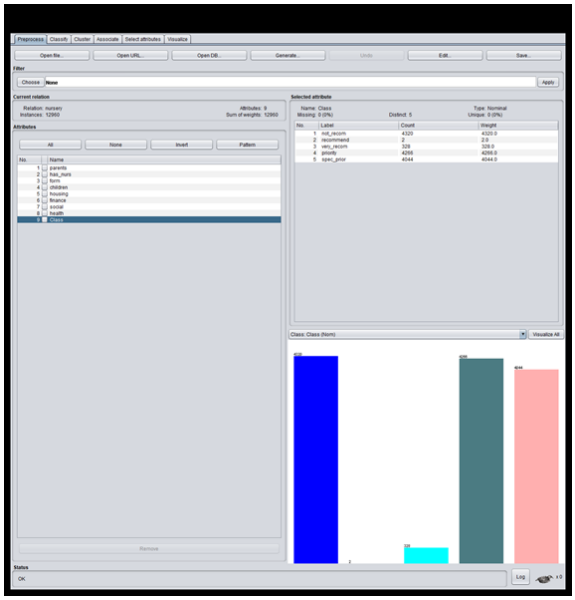
Nursery Data Set:

This data set is derived from a decision model originally developed to rank applications for nursery schools. It was used during several years in 1980's when there was excessive enrollment to these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation. The final decision depended on three subproblems: occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family. And in this data set, we use 8

attributes to describe them:

has_nurs: Child's nursery, parents: Parents' occupation, form Form of the family, children: Number of children, children Number of children, finance: Financial standing of the family, social :Social conditions, and health: Health conditions. (details can be found in nursery.names)

And there are 5 class values: not_recom (not recommended), recommend, very_recom(very recommended), priority, spec_prior(special priority). Here is a distribution of class values:



Why these two datasets are interesting:

For practical implications: Having a car is very common for every households in United States, and the model help people to determine which car to buy based on some facts about the cars can be helpful to a lot of people (maybe including me in the future)! And for the nursery dataset, I found it interesting and useful because I used to have the dream of being admitted by a medical school, however, the tuition fee and the long learning period let me put aside my dream, because my family cannot afford that. However, this nursery data set can be trained into a model for students to evaluate their application before they apply to the medical schools. I hope this will help them realize their medical dream!

In terms of machine learning: I choose these two problems because one of them have relatively small size: 1728 samples, and the other have 12960 samples. I want to see the result for different algorithm with respect to the training samples. Also these two dataset are imbalanced data set, and I want to see the ability for the different models to predict the minority classes (class with little samples in the training set).

Decision Tree:

Algorithm:

For Car, I use the J48 algorithm which is "top-down induction of decision trees". It bases on information theory, and always select the node that break the data as purely as possible.

Pruning: The way I prune my decision tree is to set the "unpruned" in Weka to be false, which means let Weka to prune my tree automatically. Besides, I use the following ways to further prune it:

Change the confidence factor and minNumObj(minimum of object when you reach a leaf node). After several tests, I found that the confidence factor should be 0.25 which is the default value in order to get maximum accuracy. And the minNumObj is better to be 2 for these two datasets.

Result:

For Nurse problem, the accuracy on testing set is 96.37%. And the average accuracy on my **10-folds cross-validation** is 97.05%.

For Car problem, the accuracy on testing set is 90.9%. And the 10-folds cross-validation accuracy is 92.36%.

Analysis:

Notes for Learning curves analysis:

In Weka, I used "Filtered Classifier", and use "Resample" filter with no replacement to perform the following test.

For Nurse:

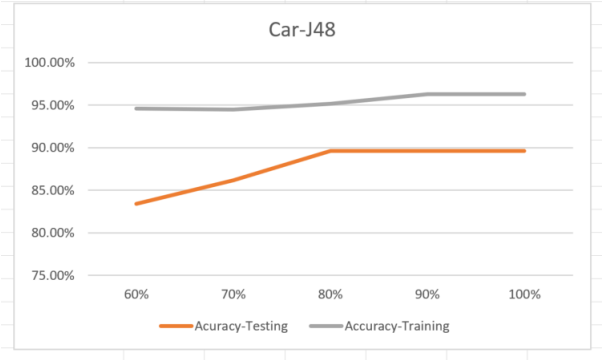
Learning curves analysis: When I try to plot the learning curve, I change the number of the training samples from 4200 to 9072, and the accuracy doesn't change much and remain high (94%-96%). Therefore, to save pages, I didn't include the learning curve. I think maybe because there is only 3 major class (Note: Although there are 5 classes in this problem, 2 of them has only 2 and 328 samples). Because there are only major 3 classes in both training and testing set, it is easy for decision tree to work.

Cross-validation analysis: The accuracy for 10-folds cross-validation is very close to the accuracy of my test set. That means there are little bias when I split the training and test set. The interesting thing is that the cross-validation accuracy is slight higher than the test accuracy. I think it may due to the minor classes (in 12960 samples, only 2 samples are in "recommended "class and only 328 samples in "priority" classes) in this problem : Their contribution to training set is so little that the model didn't learn them. Therefore, in test set, the model can't identify them.

For Car:

Learning curves analysis: From the plot, we can see that the error rate doesn't change

much when we use 80% of the training set, when we use more data than that, the algorithm can't further learn this problem, therefore the accuracy on the test set didn't change anymore. And the Gap of between Training and Testing isn't very large, means that our algorithm didn't suffer from high variance, there is no overfitting problem.



Note for above graph: The total dataset has 1728 samples, and the percentage in the graph is the percentage of the 1728 samples.

Cross-validation analysis: The 10-folds cross validation accuracy is 2% higher than test accuracy, maybe it is because the process of splitting the test set. Same as Nurse problem, the data in this problem is also unbalanced, therefore the model can't learn all classes well.

But why the test accuracy of Car problem is lower than Nurse problem? I think it is because in Nurse problem, there is 3 major class, and 2 minor class, and the number of samples in minor class in small. But in Car problem, there is only 1 major class, and 3 minor class, and although the percentage of minor class in Car problem is higher than the percentage of minor class in Nurse problem. Therefore, there is more chance for decision tree to misclassify items in Car problem.

Neural Networks:

Parameter Choosing:

For Car: With number of epochs set to 500, I use sets of parameters to train the neural network model. And I compared their accuracy, it turns out that when leaning rate = 0.2 or 0.3, momentum = 0.2, the model achieves 99.6% accuracy on test set. To be more convincing, I use 5-folds cross validation on these sets of parameters, and the accuracy is 99.19%. It proves that these set of parameters do work.

Set of Parameters	L=0.1, M=0.2	L=0.2, M=0.2	L=0.3, M=0.2	L=0.1, M=0.0	L=0.1, M=0.1	L=0.1, M=0.3
Test Accuracy	98.45%	99.61%	99.61%	98.84%	98.84%	98.84%

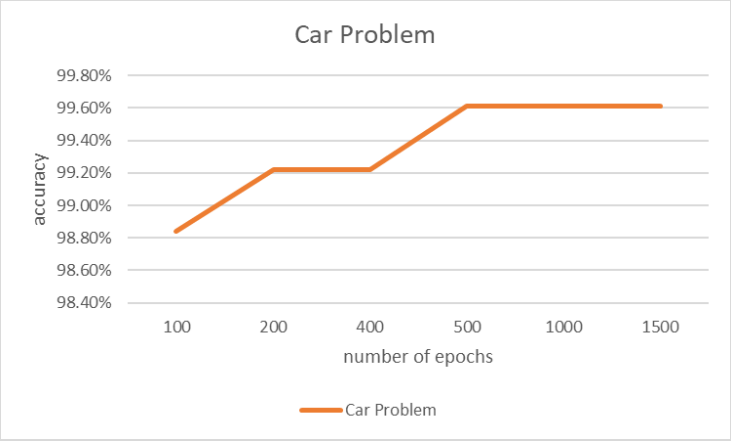
For Nurse: I use the same sets of parameters and fix the training time to 500. The set of

L =0.2, M=0.2 can achieve the highest test accuracy, which is 99.97%, and 5-folds cross-validation accuracy is 99.71%.

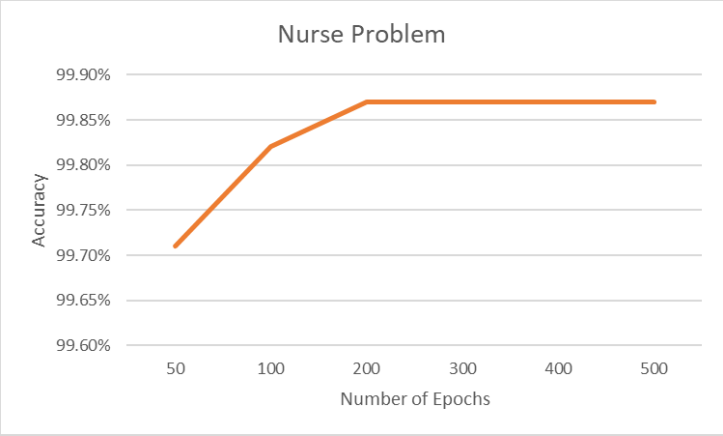
Set of Parameters	L=0.1, M=0.2	L=0.2, M=0.2	L=0.3, M=0.2	L=0.1, M=0.0	L=0.1, M=0.1	L=0.1, M=0.3
Test Accuracy	97.32%	99.97%	99.87%	97.32%	97.32%	97.32%

Result:

For Car problem, the accuracy of the neural networks with learning rate = 0.2 and momentum = 0.2:



For Nurse Problem, the result for L=0.2, M=0.2:



Analysis:

Findings during parameter choosing: In car problem, I found that the accuracy doesn't change much with the change of the momentum value. I think maybe it is because momentum is for the algorithm to cross the stationary point when we use gradient decent to train our model. So, it doesn't affect the model that much once the model is trained.

Findings about training time: Although the training time is set to be 500 in Weka for both car and nurse problem, there are huge difference between the training time of the two

problems. The average training time is around 4 second for car problem. However, the training time is around 40 second for nurse problem. I think it is because the difference in number of samples between the two data sets. The car problem only has 1728 samples; however, the nurse problem has 12960 samples. Therefore, I concluded that the training time mainly depends on the number of training samples.

Difference between Neural Networks and Decision Tree: The accuracy of Neural Networks is higher than Decision Tree, however, the neural network takes longer time to train. To account for this, I think neural networks use backpropagation to train every its neurons, therefore it can learn deeper (sometimes cannot be understood by human) features, and it can capture some features of the minority, therefore, it can achieve higher accuracy. And because it uses backpropagation for each of its training sample, it takes longer time especially when training set is large.

Boosting:

Pruning: I use different sets of values of Confidence factor and Unpruned to achieve different pruning effect. You can see the accuracy for each set in the result.

Result: I use different pruning effect and boost iterations to see the effect.

For Car:

Unpruned	Confidence Factor	Number of Iterations	Test Accuracy
Ture	0.125	10	93.62%
False	0.125	10	95.17%
False	0.25	10	94.21%
False	0.5	10	94.21%
True	0.125	20	93.82%
False	0.125	20	94.87%
False	0.25	20	94.21%
False	0.5	20	94.98%
True	0.125	40	94.21%
False	0.125	40	94.98%
False	0.25	40	95.56%
False	0.5	40	95.17%

For Nurse:

Unpruned	Confidence Factor	Number of Iterations	Test Accuracy
Ture	0.125	10	98.81%
False	0.125	10	99.04%
False	0.25	10	98.94%
False	0.5	10	98.81%
True	0.125	20	99.17%
False	0.125	20	99.12%
False	0.25	20	99.30%
False	0.5	20	99.07%
True	0.125	40	99.27%
False	0.125	40	99.30%
False	0.25	40	99.28%
False	0.5	40	99.33%

Analysis:

Comparison with decision tree without boosting: From the table, I found that the result for boosted version decision tree can achieve better result. I think it is because boosted version can use multiple decision tree as weak classifier, and in each iteration it will focus on the sample that is misclassified, so that it can capture the small portion of my data set (the minority class) which usually be ignored by the decision tree and misclassified, so it can achieve higher accuracy.

Number of Iterations Analysis: Firstly, I found that the training time is correlated with the number of iterations. For 10 iterations in Nurse problem, it took around 0.03s. And 20 iterations took around 0.05s, and 40 iterations took 0.1s. Intuitively, the more iterations you train, the more time you need. Secondly, for both problem, I found that the more iterations you train, the more accurate your model can achieve. I think it is because both data sets contain some minor class (with relatively small amount of the samples), more iterations can help the model to capture their features.

Pruning analysis: I found that after using boosting, the aggressive pruning methods which cannot achieve high accuracy with single decision tree can achieve satisfying result. And I found that increasing the number of iterations can be more useful in choosing aggressively pruned decision tree as weak classifier. (Can be seen in Unpruned = False, Confidence factor =0.5). I think it is mainly because boosting can improve the accuracy by using multiple weak classifier. So, the weaker the classifier is, the more boosting can improve the original classifier.

Support Vector Machines:

Result: I use two different kernel functions with different parameters to see the effect of kernel and parameter. For Poly Kernel, the parameter is exponent, for RBF Kernel, the parameter is gamma.

For Car:

Kernel	Parameter	Accuracy	Training Time
Poly	1.0	93.62%	0.26s
Poly	2.0	99.03%	0.58s
Poly	3.0	100%	1.65s
RBF	0.01	85.1%	0.86s
RBF	0.5	97.1%	1.79s
RBF	1.0	90.34%	3.38s

For Nurse:

Kernel	Parameter	Accuracy	Training Time
Poly	1.0	92.48%	10.07s
Poly	2.0	100%	30.22s
Poly	3.0	99.97%	88.6s
RBF	0.01	92.05%	72.35s
RBF	0.5	99.89%	394s
RBF	1.0	97.58%	611s

Analysis:

Training time with the parameter: The first major difference between SVM and other algorithms is the training time, the training time for SVM is even longer than neural network, which surprised me. And the training time largely depends on the parameter I choose, which is the gamma for RBF and the exponent for Poly. This is reasonable because the parameter determines the complexity of the model. For example, lager exponent will result in a more complex polynomial, which takes longer time to train.

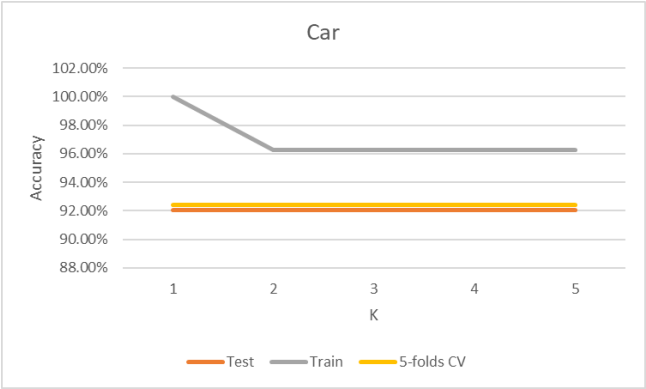
Training time with the data set: Also, I found that the number of samples also determines the training time for SVM. From the tables, we can see that the training time for nurse problem is much longer than the car problem.

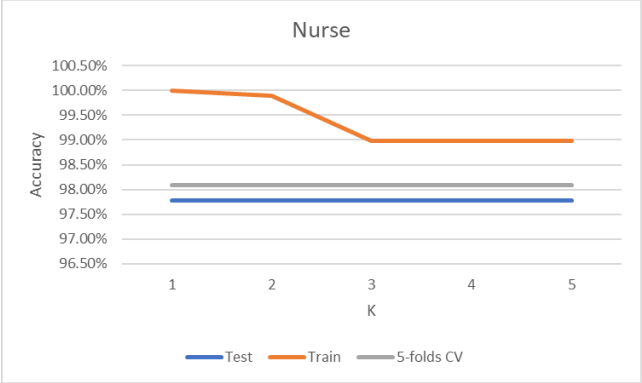
Accuracy with the parameter: The first exciting thing is we got several 100%s in SVM! And I found that the accuracy for polynomial kernel will increase with the increment of exponent. It is because the lager exponent gives us a more flexible model which can fit our problem better. And although I cannot find what exactly gamma is in a RBF kernel online, I found that larger gamma will result in a longer training time but not necessarily more accurate model (see the results for gamma =0.5 versus gamma =1.0), therefore I think it is related to the complexity of RBF kernel, and due to the properties of RBF, the

more complex model will not always be the better model.

k-Nearest Neighbors:

Results and Analysis:





After I plotted the test accuracy curve, I found that it doesn't change with the change of K, so I thought it might be the bias of my test set, so I use 5-folds cross validation. However, the 5-folds cross-validation has the same behavior! I think it may due to the distribution of my data sets. The most part of the data in the set is cluster together, so that the result for k from 1 to 5 didn't change much.

And I think the minority class in the car testing data set are misclassified to the huge cluster near them, so the accuracy is not high for Car problem.

KNN does not require "training", therefore it is called "lazy model" in Weka. However, it takes longer time during testing. I found that the testing time will increase with the value of K. That's reasonable because it requires more computation for larger K value.