# Hybrid Cost-sensitive Decision Tree

Shengli Sheng, Charles X. Ling

Department of Computer Science, The University of Western Ontario
London, Ontario N6A 5B7, Canada
{cling, ssheng}@ csd.uwo.ca

**Abstract.** Cost-sensitive decision tree and cost-sensitive naïve Bayes are both new cost-sensitive learning models proposed recently to minimize the total cost of test and misclassifications. Each of them has its advantages and disadvantages. In this paper, we propose a novel cost-sensitive learning model, a hybrid cost-sensitive decision tree, called DTNB, to reduce the minimum total cost, which integrates the advantages of cost-sensitive decision tree and of the cost-sensitive naïve Bayes together. We empirically evaluate it over various test strategies, and our experiments show that our DTNB outperforms cost-sensitive decision and the cost-sensitive naïve Bayes significantly in minimizing the total cost of tests and misclassification based on the same sequential test strategies, and single batch strategies.

## 1 Introduction

Inductive learning techniques have had great success in building classifiers and classifying test examples into classes with a high accuracy or low error rate. However, in many real-world applications, lowing misclassification error is not the goal as "errors" can cost very differently. This type of learning is called cost-sensitive learning. Turney [14] surveys a whole range of costs in cost-sensitive learning, among which two types of costs are most important: misclassification costs and test costs. For example, in a binary classification task, the cost of false positive (FP) and the cost of false negative (FN) are often very different. In addition, attributes (tests) may have different costs, and acquiring values of attributes also incurs costs. The goal of learning is to minimize the sum of the misclassification costs and the test costs.

Tasks involving both misclassification and test costs are abundant in real-world applications. For example, when building a model for medical diagnosis from the training data, we must consider the cost of tests (such as blood tests, X-ray, etc.) and the cost of misclassifications (errors in the diagnosis). Further, when a doctor sees a new patient (a test example), tests are normally ordered, at a cost to the patient or his/her insurance company. To better diagnose or predict the disease of the patient (i.e., reducing the misclassification cost). Doctors must balance the trade-off between potential misclassification costs and test costs to determinate which tests should be ordered, and at

what order, to reduce the expected total cost. A case study on heart disease is given in the paper.

In this paper, we propose a new cost-sensitive learning model, DTNB, which integrates the advantages of the cost-sensitive decision tree and the cost-sensitive naïve Bayes, both of which minimize the total cost of misclassifications and tests. DTNB uses the cost-sensitive decision tree to collect the required tests for test examples, and uses the cost-sensitive naïve Bayes to classify. For a test example, after the required tests are collected according to the cost-sensitive decision tree, the tests are performed with a cost and their results are available. Then the cost-sensitive naïve Bayes built on all the training data is applied to classify the test example. The naïve Bayes model can make use of the known values which do not appear in the path which the test example follows to go down to a leaf in the cost-sensitive decision tree. Thus, we can expect that the cost-sensitive DTNB can achieve lower total cost than the cost-sensitive decision tree and the cost-sensitive naïve Bayes do alone.

The rest of paper is organized as follows. We first review the related work in Section 2. Then we describe our new cost-sensitive learning model, DTNB, to reduce the minimum total cost of tests and misclassifications in Section 3. In Section 4, we present empirical experiments. The paper concludes with discussion and some directions for the future work.

## 2 Review of Previous Work

Cost-sensitive learning has received extensive attentions in recent years. Turney [14] analyzes a variety of costs in machine learning, such as misclassification costs, test costs, active learning costs, computation cost, human-computer interaction cost, etc. Two types of costs are singled out as the most important in machine learning: misclassification costs and test costs, and test costs are normally considered in conjunction with misclassification costs. Much work has been done in considering non-uniform misclassification costs (alone), such as [4, 5, 7]. Those works can often used to solve problem of learning with very imbalanced datasets [3]. Some previous work, such as [10, 12], consider the test cost alone without incorporating misclassification cost. As pointed out by [14] it is obviously an oversight. As far as we know, the only work considering both misclassification and test costs includes [13, 15, 9, 2]. We discuss these works in detail below.

In [15], the cost-sensitive learning problem is cast as a Markov Decision Process (MDP), and an optimal solution is given as a search in a state space for optimal policies. While related to our work, their research adopts an optimal search strategy, which may incur very high computational cost to conduct the search. In contrast, we adopt the local search similar to [11] using a polynomial time algorithm to build a new decision trees, and our test strategies are also polynomial to the tree size. (Greiner et al. 2002) studied the theoretical aspects of active learning with test costs using a PAC learning framework, which models how to use a budget to collect the relevant information for the real-world applications with no actual data at beginning. Our algorithm builds a model from history

data to minimize the total cost of misclassification and tests for a new case with missing values. Turney [13] presented a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the cost of tests and misclassification. Our algorithm essentially adopts the same decision-tree building framework as in [11], and it is expected to be more efficient than Turney's genetic algorithm based approach.

Ling et al. [9] propose a cost-sensitive decision tree learning program that minimizes the total cost of tests and misclassifications. They also propose several test strategies, and compare their results to C4.5. However, for a test example, the cost-sensitive decision tree ignores the information supplied by the known attributes which do not appear in the path which the test example follows to go down to a leaf in the cost-sensitive decision tree. Chai et al. [2] propose a cost-sensitive naïve Bayes based algorithm, called CSNB, which searches for minimal total cost of tests and misclassifications. They also propose a sequential test strategy and a single batch test strategy. However, the cost-sensitive naïve Bayes does not learn the general attribute structure (such as the tree structure) but only probability tables from training data. The test sequence for each test example is less comprehensible.

Our model, DTNB, combines the advantages of cost-sensitive decision tree and naïve Bayes. It utilizes the structure of the cost-sensitive decision tree to collect the beneficiary tests for a test example and makes use of the information in the known attributes which are ignored by the cost-sensitive decision tree to reduce the misclassification cost. We expect that our DTNB outperform cost-sensitive decision tree and cost-sensitive naïve Bayes alone in terms of the total cost of tests and misclassification.

The new cost-sensitive model, DTNB, is composed of decision tree and naïve Bayes, but it is much different from NBTree [8] proposed by Kohavi. First of all, NBTree is not a cost-sensitive learning model. The learning algorithm of NBTree is similar to C4.5 [Qui93]. DTNB is a cost-sensitive learning to minimize the total cost of tests and misclassification. Secondly, in NBTree, a naïve Bayes is constructed for each leaf using the data associated with the leaf. However, DTNB only constructs one naïve Bayes using all the training data. This naïve Bayes acts as a hidden node at each node (including the leaves) of the cost-sensitive decision tree. The details of difference between NBTree and DTNB are explained in Section 3.

## 3 The New Cost-sensitive Learning - DTNB

We assume that we are given a set of training data (with possible missing attribute values), the misclassification costs, and test costs for each attribute. We propose a novel cost-sensitive learning model, DTNB, which combines the advantages of cost-sensitive decision tree and naïve Bayes. The rationale of DTNB is based on our observations. We note that cost-sensitive decision tree has the ability of learning a general structure, and the structure of the tree plays an important role for collecting the most beneficiary unknown values. However, the decision tree ignores the original known values which do not appear

in the tree for classify a test example. In non-cost-sensitive learning, this is one reasonable feature of decision tree. But in cost-sensitive learning, any value is available with a certain cost. We do not want waste any available information. Naturally, making use of all known values can reduce the total cost. The information of the known attributes which do not appear in the path through which the test example goes down to a leaf of the tree is useful for cost-sensitive classification to reduce the misclassification cost. Fortunately, cost-sensitive naïve Bayes indeed utilizes all known attributes for misclassification, but it does not have a structure learning ability to help determine which tests and in what order should be done for unknown attributes.

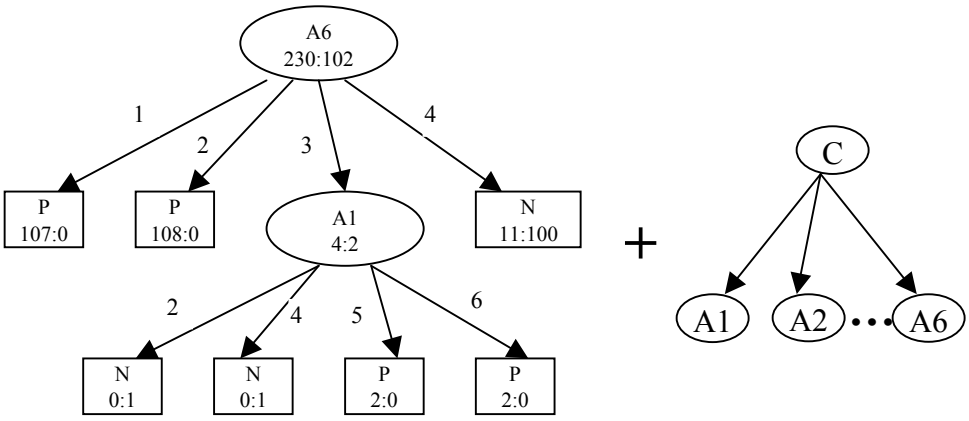


**Fig. 1.** An example of cost-sensitive DTNB

In order to overcome these drawbacks and combine those advantages in the two cost-sensitive models, we propose a novel cost-sensitive learning model, which integrates cost-sensitive decision tree with cost-sensitive naïve Bayes, called DTNB. Figure 1 shows the structure of an example of the novel cost-sensitive learning model DTNB. We can see DTNB is an integration model with two parts. The left part is a cost-sensitive decision tree which is used for finding the required tests for each testing example. Besides the cost-sensitive tree, DTNB also contains a naïve Bayes (right part), which is for classification.

First of all, DTNB builds a cost-sensitive decision tree, given a set of training data, the misclassification costs, and test costs for each attribute. The building procedure is similar to C4.5. Instead of using entropy based splitting criteria, we use the *expected* total misclassification cost to select an attribute for splitting. This gives a more accurate choice for attribute selection. That is, an attribute may be selected as a root node of a decision tree if the sum of the test cost and the expected misclassification costs of all branches is the minimum among other attributes, and is less than that of the root. For a subset of examples with *tp* positive examples and *tn* negative examples, if $C_P = tp \times TP + tn \times FP$ is

the total misclassification cost of being a positive leaf, and $C_N = tn \times TN + tp \times FN$ is the total misclassification cost of being a negative leaf, then the probability of being positive is estimated by the relative cost of $C_P$ and $C_N$; the smaller the cost, the larger the probability (as minimum cost is sought). Thus, the probability of being positive is: $1 - \frac{C_P}{C_P + C_N} = \frac{C_N}{C_P + C_N}$. The expected misclassification cost of being positive is: $E_P = \frac{C_N}{C_P + C_N} \times C_P$. Similarly, the probability of being a negative leaf is $\frac{C_P}{C_P + C_N}$; and the expected misclassification cost of being negative is: $E_N = \frac{C_P}{C_P + C_N} \times C_N$. Therefore, without splitting, the expected total misclassification cost of a given set of examples is: $E = E_P + E_N = \frac{2 \times C_P \times C_N}{C_P + C_N}$. If an attribute $A$ has $l$ branches, then the expected total misclassification cost after splitting on $A$ is: $E_A = 2 \times \sum_{i=1}^{l} \frac{C_{P_i} \times C_{N_i}}{C_{P_i} + C_{N_i}}$. Thus, $(E - E_A - T_C)$ is the expected cost reduction splitting on $A$, where $T_C$ is the total test cost for all examples on $A$. It is easy to find out which attribute has the smallest expected total cost (the sum of the test cost and the expected misclassification cost), and if it is smaller than the one without split (if so, it is worth to split). With the expected total misclassification cost described above as the splitting criterion, the lazy-tree learning algorithm is shown in Figure 2.

Simultaneously, we build a cost sensitive naïve Bayes. Note that this model is built on all the training data, and for all nodes in the tree. However, NBTree [Koh96] treats the segmentation of decision tree as an advantage. It builds a naïve Bayes at each leaf of the decision tree. And the naïve Bayes constructed for a leaf uses only the data associated with the leaf. However, as the tree grows, the training data are split into the lower level nodes. Finally, there are very little data in the leaves. The classification based on these leaves is far less accurate, so that the misclassification cost goes higher. This is reason that NBTree is proposed for larger dataset. However, without larger dataset assumption DTNB overcomes the shortcoming of segmentation of decision tree by constructing only one naïve Bayes using all the training data. This naïve Bayes acts as a hidden model at each node (including the leaves) of the cost-sensitive decision tree. The hidden model is only for classification. Thus, DTNB does not utilize the data which go down into a leaf of the tree to classify a testing example which drops into this leaf. It classifies the test example by the only hidden cost-sensitive naïve Bayes.

DTNB only builds one general naïve Bayes from all the training data. Whereas, the posterior probabilities of a test example *e* are computed from the known attributes and the tested unknown attributes. The unknown attributes which are not selected to perform

testing are not concerned. With the posterior probabilities, if $FN \times P(+|e) > FP \times P(-|e)$, this test example is classified as negative, otherwise, as positive. A misclassification cost may be incurred if the prediction of the test example is wrong. Thus, for each test example, not only the attributes appearing on the tree, but also the known attributes can be fully used to make correct classification, so that the total misclassification cost can be reduced, as any known value is worthy of a certain cost. But for the cost-sensitive decision tree, it is possible some known attributes are not used to split the training data, so that they become useless for the classification. DTNB makes use of all known attributes, as well as the available values of the collected unknown attributes at certain test costs.

*CSDT*(*Examples, Attributes, TestCosts*)

1. Create a *root* node for the tree
2. If all examples are positive, return the single-node tree, with *label* = +
3. If all examples are negative, return the single-node tree, with *label* = -
4. If attributes is empty, return the single-node tree, with label assigned according to *min* ($E_P$, $E_N$)
5. Otherwise Begin
   a. If *maximum cost reduction* < *0* return the single-node tree, with label assigned according to *min* ($E_P$, $E_N$)
   b. *A* is an attribute which produces maximum cost reduction among all the remaining attributes
   c. Assign the attribute *A* as the tree *root*
   d. For each possible value vi of the attribute *A*
      i. Add a new branch below root, corresponding to the test $A=v_i$
      ii. Segment the training examples into each branch *Example*_$v_i$
      iii. If no examples in a branch, add a leaf node in this branch, with label assigned according to *min* ($E_P$, $E_N$)
      iv. Else add a subtree below this branch, CSDT(*examples*_$v_i$, *Attributes-A, TestCosts*)
6. End
7. Return *root*

**Fig. 2**. Algorithm of cost-sensitive decision tree

In the naïve Bayes model of DTNB, the Laplace Correction is applied. That is, $p(a \mid +) = \frac{N_a + 1}{N + m}$, where *Na* is the number of instances whose attribute $A_1=a$, *N* is the number of instances whose class is +, and m is the number of classes.

After DTNB is built, for each testing example, there are two steps to find the minimum total cost of tests and misclassifications. The first step is to utilize the tree structure of the cost-sensitive decision tree to collect a set of tests which need be performed according to a

certain strategy (there are several strategies explained in Section 4). The total test cost is accumulated in the step. After the set of tests are done, the values of the unknown attributes in the test example are available. It automatically goes to the second step, where the cost-sensitive naïve Bayes model is used to classify the test example into a certain class. The naïve Bayes uses not only the unknown attributes tested but also all known attributes. If it is classified incorrectly, there is misclassification cost. We empirically evaluate it over various test strategies in next section.

# 4 Experiments

We evaluate the performance of DTNB on two categories of test strategies: Sequential Test, and Single Batch Test. For a given test example with unknown attributes, the Sequential Test can request only one test at a time, and wait for the test result to decide which attribute to be tested next, or if a final prediction is made. The Single Batch Test, on the other hand, can request one set (batch) of one or many tests to be done simultaneously before a final prediction is made.

## 4.1 DTNB's Optimal Sequential Test

Recall that Sequential Test allows one test to be performed (at a cost) each time before the next test is determined, until a final prediction is made. Ling, et al. [9] described a simple strategy called *Optimal Sequential Test* (or OST in short) that directly utilizes the decision tree built to guide the sequence of tests to be performed in the following way: when the test example is classified by the tree, and is stopped by an attribute whose value is unknown, a test of that attribute is made at a cost. This process continues until the test case reaches a leaf of the tree. According to the leaf reached, a prediction is made, which may incur a misclassification cost if the prediction is wrong. Clearly the time complexity of OST is only linear to the depth of the tree.

One weakness with this approach is that it ignores some known attributes which do not appear in the path through which a test example goes down to a leaf. However, these attributes can be useful for reducing the misclassification cost. Like the OST, We also propose an Optimal Sequential Test strategy for DTNB (section 3), called DNOST in short. It has the similar process as OST. The only difference is that the class prediction which is not made by the leaf it reached, but the naïve Bayesian classification model in DTNB. This strategy utilizes the tree structure to collect the most useful tests for a test example. And it also utilizes the entire original known attributes in the test example with the unknown attributes tested to predict the class of the test example. We can expect DNOST outperforms OST.

**Comparing Sequential Test Strategies.** To compare various sequential test strategies, we choose 10 real-world datasets which are listed in Table 1, from the UCI Machine

Learning Repository [1]. The datasets are first discretized using the minimal entropy method [6]. These datasets are chosen because they are binary class, have at least some discrete attributes, and have a good number of examples. Each dataset is split into two parts: the training set (60%) and the test set (40%). Unlike the case study of heart disease, the detailed test costs and group information [13] of these datasets are unknown. To make the comparison possible, we simply choose randomly the test costs of all attributes to be some values between 0 and 100. This is reasonable because we compare the relative performance of all test strategies under the same chosen costs. To make the comparisons straightforward, we set up the same misclassification costs 200/600 (200 for false positive and 600 for false negative). For test examples, a certain ratio of attributes (0.2, 0.4, 0.6, 0.8, and 1) are randomly selected and marked as unknown to simulate test cases with various degrees of missing values.

**Table 1.** Datasets used in the experiments

| | No. of Attributes | No. of Examples | Class dist. (N/P) |
|---|---|---|---|
| Ecoli | 6 | 332 | 230/102 |
| Breast | 9 | 683 | 444/239 |
| Heart | 8 | 161 | 98/163 |
| Thyroid | 24 | 2000 | 1762/238 |
| Australia | 15 | 653 | 296/357 |
| Tic-tac-toe | 9 | 958 | 332/626 |
| Mushroom | 21 | 8124 | 4208/3916 |
| Kr-vs-kp | 36 | 3196 | 1527/1669 |
| Voting | 16 | 232 | 108/124 |
| Cars | 6 | 446 | 328/118 |

In this section, we compare our DNOST with the other two sequential test strategies available, OST, and CSNB [2] on 10 real-world datasets to see which one is better (having a smaller total cost). Note that DNOST and OST use the same decision tree to collect beneficiary tests. However, DNOST uses DTNB's naïve Bayes for classification, while OST uses the leaves of tree to classify test examples. CSNB follows the same test strategy: determine next test based on the previous test result. However, it is based on the naïve Bayes only. In all, all of them are based on the same test strategy, but they are applied different cost-sensitive learning models. That is, their performances directly stand for the performances of different learning models. We repeat this process 25 times, and the average total costs for the 10 datasets are plotted in Figure 3.

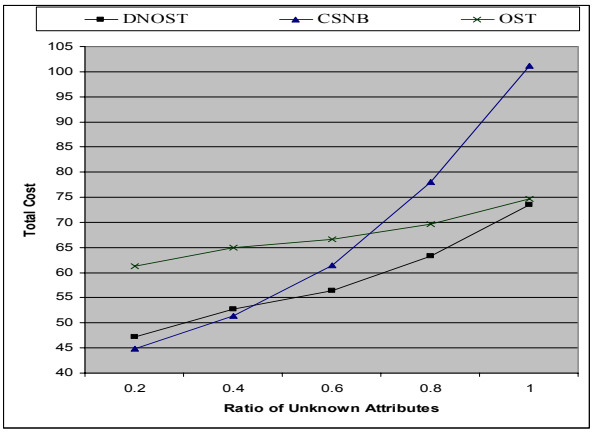


**Fig. 3.** The total cost of our new Sequential Test Strategy DNOST compared to previous strategies (OST and CSNB)

We can make several interesting conclusions. First, DNOST performs the best among the three sequential test strategies. When the unknown attribute ratio is higher, the difference between DNOST and CSNB becomes bigger. However, DNOST is gradually close to OST when the unknown ratio is increased. When the unknown ratio is lower, the difference between DNOST and OST is bigger, as more known attributes are utilized in DTNB, but they are ignored in cost-sensitive decision tree. Second, the results proof our expectation which DTNB integrates the advantage of the decision tree and the naïve Bayes and overcomes their defects. When the unknown ratio is lower, there are more known attributes ignored by OST, so that OST performs worse, whereas DNOST and CSNB perform better and are closer, as they make use of the known values. When the unknown ratio is higher, there are less known attributes ignored by OST and both DNOST and OST utilize the tree structure to collect the most beneficiary tests, so that they perform better and are close to each other.

### 4.2 Single Batch Test Strategies

The Sequential Test Strategies have to wait for the result of each test to determine which test will be the next one. Waiting not only costs much time, but also increases the pressure and affects the life quality of patients in medical diagnosis. In manufacturing diagnoses, it delays the progress of engineering. Even in some particular situations, for example, emergence, we have to make decisions as soon as possible. In medical emergence, doctors normally order one set of tests (at a cost) to be done at once. This is the case of the Single Batch Test.

In [9] a very simple heuristic is described. The basic idea is that when a test example is classified by a minimum-cost tree and is stopped by the first attribute whose value is unknown in the test case, all unknown attributes under and including this first attribute would be tested, as a single batch. Clearly, this strategy would have exactly the same misclassification cost as the Optimal Sequential Test, but the total test cost is higher as extra tests are performed. This strategy is called Naïve Single Batch (NSB).

The weakness of NSB is that it ignores some known attributes which do not appear in the path through which a test example goes down to a leaf after the tests are performed. However, these attributes can be useful for reducing the misclassification cost. Like the NSB, we apply the similar process on DTNB. The only difference is the class prediction which is not made by the leaf a test example reached after the tests are performed, but by the naïve Bayes classification model. We call this process DTNB's Naïve Single Batch Test (or DN-NSB in short).

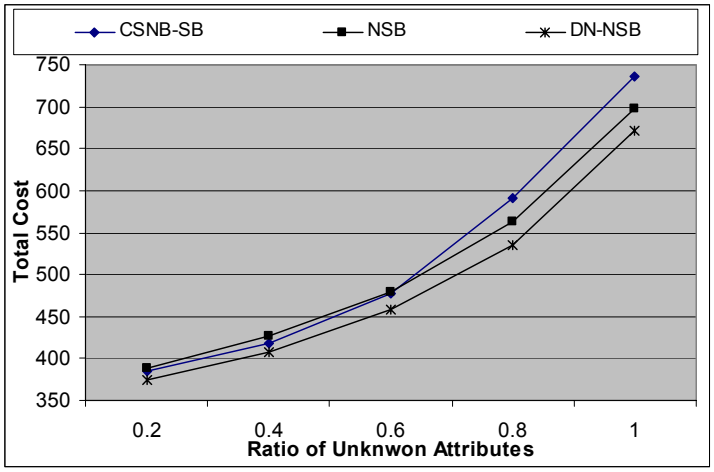


**Fig. 4.** The total cost of our new Single Batch Test Strategies DN-NSB compared to their previous strategies (NSB and CSNB-SB).

**Comparing Single Batch Test Strategies.** We use the same experiment procedure on the same 10 datasets used in Section 4.1 (see Table 1) to compare various Single Batch Test strategies including CSNB-SB [2]. The only change is the misclassification costs, which are set to 2000/6000 (2000 for false positive and 6000 for false negative). The misclassification costs are set to be larger so the trees will be larger and the batch effect is more evident. Note that DN-NSB and NSB use the same decision tree to collect beneficiary tests. However, DN-NSB uses DTNB's naïve Bayes for classification, while NSB uses the leaves of tree to classify test examples. CSNB follows the same test strategy: request one set (batch) of one or many tests to be done simultaneously before a final prediction is made. However, it is based on the naïve Bayes only. In all, all of them are

based on the same test strategy, but they are applied to different cost-sensitive learning models. That is, their performances directly stand for the performances of different learning models. The total costs for the 10 datasets are compared and plotted in Figure 4.

We can make several interesting conclusions. First, the single batch test strategy (DN-NSB) based on DTNB outperforms others on any unknown ratio. CSNB-SB outperforms NSB when the unknown ratio is higher, but it is worse than NSB when the unknown ratio goes down. Second, the results again proof our expectation which DTNB integrates the advantage of the decision tree and the naïve Bayes and overcomes their defects. When the unknown ratio is lower, there are more known attributes ignored by NSB, so that NSB performs worse. DN-NSB and CSNB-SB perform better, as they make use of the known values. When the unknown ratio is higher, there are less known attributes ignored by NSB and both DN-NSB and NSB utilize the tree structure to collect the most beneficiary tests, so that they perform better.

## 5 Conclusion and Future Work

In this paper, we present a hybrid decision tree learning algorithm, which integrate with naïve Bayes, to minimize the total cost of misclassifications and tests. We evaluate the performance (in terms of the total cost) empirically, compared to previous methods using decision tree and naïve Bayes alone. The results show that our novel learning algorithm, DTNB, performs significantly better than the decision tree learning and the naïve Bayes learning alone.

In our future work we plan to design smart single batch test strategies. We also plan to incorporate other types of costs in our hybrid decision tree learning DTNB and test strategies.

## References:


1. Blake, C.L., and Merz, C.J., *UCI Repository of machine learning databases (website)*. Irvine, CA: University of California, Department of Information and Computer Science (1998).
2. Chai, X., Deng, L., Yang, Q., and Ling,C.X., Test-Cost Sensitive Naïve Bayesian Classification. *In Proceedings of the Fourth IEEE International Conference on Data Mining*. Brighton, UK : IEEE Computer Society Press (2004).
3. Chawla,N.V., Japkowicz, N., and Kolcz, A. eds., *Special Issue on Learning from Imbalanced Datasets*. SIGKDD, 6(1): ACM Press (2004).
4. Domingos, P., MetaCost: A General Method for Making Classifiers Cost-Sensitive. *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 155-164. San Diego, CA: ACM Press (1999).
5. Elkan, C., The Foundations of Cost-Sensitive Learning. *In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence*, 973-978. Seattle, Washington: Morgan Kaufmann (2001).

6. Fayyad, U.M., and Irani, K.B., Multi-interval discretization of continuous-valued attributes for classification learning. *In Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022-1027. France: Morgan Kaufmann (1993).
7. Ting, K.M., Inducing Cost-Sensitive Trees via Instance Weighting. *In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, 23-26. Springer-Verlag (1998).
8. Kohavi, R., Scaling up the accuracy of Naïve-Bayes Classifier: a Decision-Tree Hybrid. *In Proceeding of the Second International Conference on Knowledge Discovery and Data Mining* (KDD96). AAAI Press (1996) 202-207.
9. Ling, C.X., Yang, Q., Wang, J., and Zhang, S., Decision Trees with Minimal Costs. *In Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta: Morgan Kaufmann (2004).
10. Nunez, M., The use of background knowledge in decision tree induction. *Machine learning*, 6:231-250 (1991).
11. Quinlan, J.R. eds., *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).
12. Tan, M., Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning Journal*, 13:7-33 (1993).
13. Turney, P.D., Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2:369-409 (1995).
14. Turney, P.D., Types of cost in inductive concept learning. *In Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, California (2000).
15. Zubek, V.B., and Dietterich, T., Pruning improves heuristic search for cost-sensitive learning. *In Proceedings of the Nineteenth International Conference of Machine Learning*, 27-35, Sydney, Australia: Morgan Kaufmann (2002).