Cost-Sensitive Classification

Minhui Li
ml4026@columbia.edu

Tian Xia
tx2178@columbia.edu

Yuqin Xu
yx2478@columbia.edu

Abstract

In this report, we introduce our reproduction result of a paper solving the problem of Cost-Sensitive Classification. We also implement a bunch of classification algorithms and compare their performance in minimizing the cost. Further, we propose more advanced algorithms, Cost-Sensitive Forward Subset Selection inspired by the Forward Subset Selection of Linear Regression, and the extension of Decision Tree and Neural Network.

1 INTRODUCTION

As an automated and intelligent deep analysis technology, machine learning has very important applications in information, medicine, biology, transportation and many other fields. Especially in the healthcare domain, the progress of machine learning is of great significance. Breast cancer is currently one of the most common cancer. According to data released by DeepMind, more than 1.6 million people worldwide are diagnosed with breast cancer each year, and 500,000 people died because of breast cancer. As can also be shown in Figure 1, the 1st common type of cancer types over the world in 2016 is breast cancer. Because it takes a lot of manpower and resources to care for and treat patients, it also imposes a large economic burden on health care systems around the world. According to the introduction of X-MOL[6], in order to diagnose breast cancer, doctors need to search for traces of tumors in the tissue biopsy including at least 10 billion pixels with the help of a microscope. Usually, a doctor needs years of training to gain sufficient professional knowledge and experience, but misdiagnosis and missed diagnosis can still occur. If the machine learning algorithm can be used to train models to analyze the patient's test data and then predict the diagnosis, it can improve the efficiency and accuracy of the diagnosis, which will be of great help to both doctors and patients.

Google[4] has used machine learning, predictive analytic, and pattern recognition to diagnose breast cancer with an accuracy rate of 89%. It is much higher than the average diagnostic accuracy of human pathologists, which is 73%. Massachusetts Institute of Technology[1] also uses random-forest classifier to successfully reduce the number of unnecessary surgeries and to diagnose more cancerous lesions. Specifically, machine learning model can diagnose 97% of cancers, while traditional surgery can only diagnose 79% of cancers. It can be seen that although the use of machine learning algorithms to predict disease is still lacking the breadth of knowledge compared with experienced pathologists, it is still happy to see that well-trained models can complete the established tasks. And we look forward to developing improved models which can diagnosis cancer more quickly and reliably.

The traditional machine learning classification algorithms are aimed at pursuing classification accuracy, and assume that the misclassification costs between different categories are the same. But this is not the case. In many practical applications, the cost of dividing a category error into other categories is sometimes much higher than the cost of dividing other category errors into that category. Especially in the field of medical diagnosis, misclassification of false positive and false negative have different effects on patients. At the same time, the test costs required for classification prediction using different attributes are also different. Each test can improve the accuracy of diagnosis, and meanwhile will bring a certain amount of test cost. Our goal is to minimize the sum of test cost and

misclassification cost. And in many real-life applications, data also contains inherent uncertainty due to many reasons. We also considers the uncertainty of data.

In our project, we firstly reproduced a new machine learning algorithm which considers test cost and uncertainty of data. We also combined this algorithm with many exiting statistical learning models and trained them to predict the class of breast cancer. By adjusting the test cost and misclassification cost and comparing different models, we hope to establish a better model to help doctors diagnose cancer early and precisely.

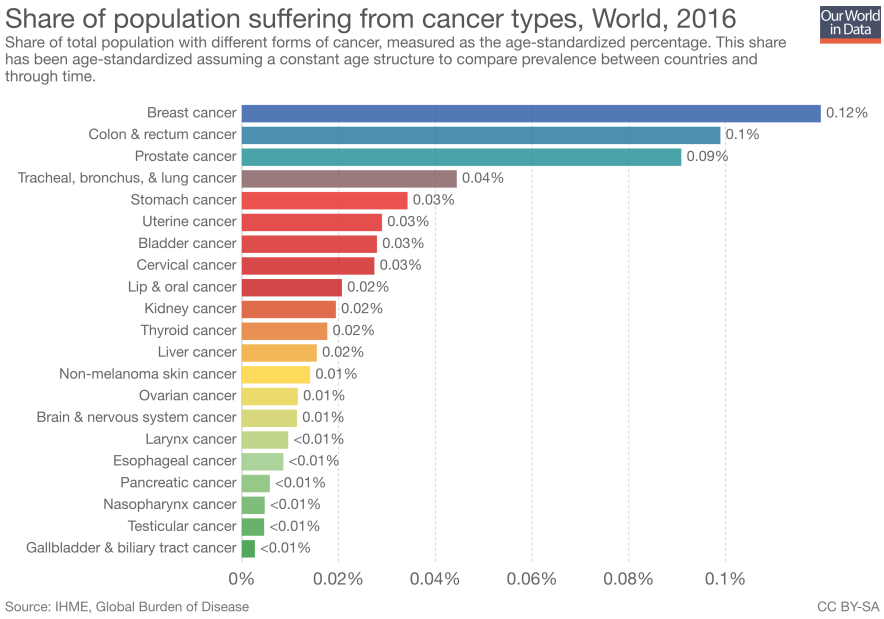


Fig. 1. Population suffering from cancer types, World, 2016

2 DATASETS

In this project, we used Breast Cancer Wisconsin (Original) Dataset from UCI Machine Learning repository[5][8]. This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. As shown in Figure 2, this dataset have 10 attributes: Sample code number, Clump Thickness, Uniformity of Cell, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses. Apart from Sample code number, other attributes characteristics are integers from 1 to 10. There are 699 Instances at all, but we remove 16 instances with missing values from them before using. We used 9 attributes to predict the binary class of breast cancer, either Benign or Malignant.

3 PAPER AND REPRODUCTION

In this section, we first describe the main algorithm in the paper we focused on in Section 3.1. Then we demonstrate our reproduction result in Section 3.2.

Sample Index	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoil	Mitoses	Class
index	1~10	1~10	1~10	1~10	1~10	1~10	1~10	1~10	1~10	2 for benign
										4 for malignant

Fig. 2. Dataset

3.1 Description

In 2014, [9] developed an algorithm named Cost-Sensitive Uncertain Naive Bayes (CSUNB) that works on the cost-sensitive classification problem. The algorithm is based on some assumptions.

- For each attribute in the classification model, in order to know its value, we need to spend a test cost. In the paper, each test cost is a random variable uniformly distributed from 0 to 100.
- After making the prediction, the total misclassification cost is accumulated that each False Positive (FP) prediction and False Negative (FN) prediction is punished by a manually set cost respectively. In one of the experiments in the papers, the misclassification costs are set to be 1000 for both False Positive and False Negative.
- Attributes are all categorical. This assumption makes the uncertainty in the dataset easier to handle. It also reduces the computation complexity of the algorithm.
- All attributes are independent from each other. It is a required assumption for Naive Bayesian classifiers.
- Data available to us in the dataset is uncertain or noisy with a manually set uncertainty, which means values we see in the set can slip to another value. Specifically in the Wisconsin dataset for example, if the uncertainty is set to be 0.2 and the value of an attribute of a data sample is 4 in the dataset, its true value has a probability of 0.8 to be 4, but a probability of 0.2 to be a value from 1 to 10 other than 4.

The CSUNB algorithm consists of the training phase and the test phase. The training phase is similar to that in the Classical Naive Bayesian classifiers. It learns the Probabilistic Cardinality of each class and each attribute-class combination to estimate the distribution.

The test phase applies a test strategy to each test data sample, make a prediction, and calculate the costs. For each test data sample, initially, a set $\tilde{A}$ of known attributes is assigned to be empty, and a set $\bar{A}$ of unknown attributes are assigned to be all the attributes in the dataset. Then the algorithm goes through an iterative procedure. In each iteration, an attribute is selected to be taken from the unknown set to the known set based on a property $Util$ of an attribute. $Util$, as Equation 1, demonstrates whether the decrease in misclassification cost with knowledge of the value of the attribute worths its test cost. In Equation 1, $\bar{A}_i$ is a certain attribute currently in the unknown set; c_{mc} is the misclassification cost; c_{test} is the test cost.

$$Util(\bar{A}_i) = c_{mc}(\tilde{A}) - c_{mc}(\tilde{A} \cup \bar{A}_i) - c_{test}(\bar{A}_i) \tag{1}$$

Nevertheless, before the test is actually implemented, the value of $\bar{A}_i$ is not revealed. Thus, the actual $c_{mc}(\tilde{A} \cup \bar{A}_i)$ is unavailable, and we calculate its expectation instead as Equation 2, where R represents the expectation; j is a class prediction; $|C|$ is the number of classes; A is the set of attributes used to calculate the misclassification cost.

$$R(y_j|A) = \sum_{k=1}^{|C|} cost_{kj} \times P(y_j|A) \tag{2}$$

The posterior $P(y_j|A)$ is calculated by Bayes Rule.

$$P(y_j|A) = \frac{P(A|y_j)P(y_j)}{P(A)} \tag{3}$$

The prior can be acquired from the training phase. The likelihood is a product of the conditional probability of each attribute as suggested by the Bayesian assumption of independence.

$$P(A|y_j) = \prod_{A_i \in A} P(A_i|y_j) \tag{4}$$

As the data is uncertain, the conditional probability needs to take the uncertainty into consideration.

$$P(A_i|y_j) = \sum_{k \in V} P_{ik} P(A_{ik}|y_j) \tag{5}$$

V represents the domain of the attribute A_i. Equation 5 is valid as the assumption that the attributes are categorical.

Finally, the conditional probability of each value can be easily calculated by the probabilistic cardinality (PC) obtained from the training phase.

$$P(A_{ik}|y_j) = \frac{PC(Aik, y_j)}{PC(y_j)} \tag{6}$$

From Equation 1 to 6, the $Util$ property is obtained. If the $Util$ of every attribute in the unknown set is negative, the loop breaks. If the $Util$ of some attributes are positive, the loop goes to the next iteration. After the iterations, based on the acquired known attribute set, we predict a label with minimum expected cost just like predicting a label with maximum expected accuracy in Naive Bayesian Classifiers. The complete test strategy for each data sample is listed as Algorithm 1.

Algorithm 1 CSUNB Test Procedure

$\tilde{A} = \varnothing$
$\bar{A} =$ Attribute set
$Cost_{exp} = 0$
while $\bar{A} \neq \varnothing$ **do**
  for attribute $a \in \bar{A}$ **do**
    Calculate $Util(a)$
  end for
  $\bar{A}^* = argmax_i(Util(\bar{A}_i)$
  $cost_{exp} = cost_{exp} + cost_{test}(\bar{A}^*)$
  if $Util(\bar{A}^*) < 0$ **then**
    Break
  end if
  Reveal the value of $\bar{A}^*$
  $\tilde{A} = \tilde{A} \cup \bar{A}^*$
  $\bar{A} = \bar{A} - \bar{A}^*$
end while
$label = argmin_y(R(y|\tilde{A}) + cost_{exp})$

3.2 Reproduction

Following the procedure discussed in Section 3.1, we managed to reproduce the line plot demonstrating the relation between the average total cost and the uncertainty. The plot of the original paper is shown in Figure 3. Our reproduction result is shown in Figure 4. Generally, the cost with regard to the uncertainty forms a linear trend. The difference in the magnitude of costs is probably caused by the randomness involved in the test cost. Another key difference is, when it comes to an uncertainty of 0.5 the paper shows a significant drop while our result shows a flattened increase. It might be caused by the randomness of value slippery in the dataset as well.

4 ALGORITHMS

In this section, we describe the basic principles of in total 11 classification algorithms. The algorithms we have implemented are described in Section 4.1, followed by the experiments in Section 5. We also propose 2 advanced algorithms that are, to some extent, hard to implement in Section 4.2.

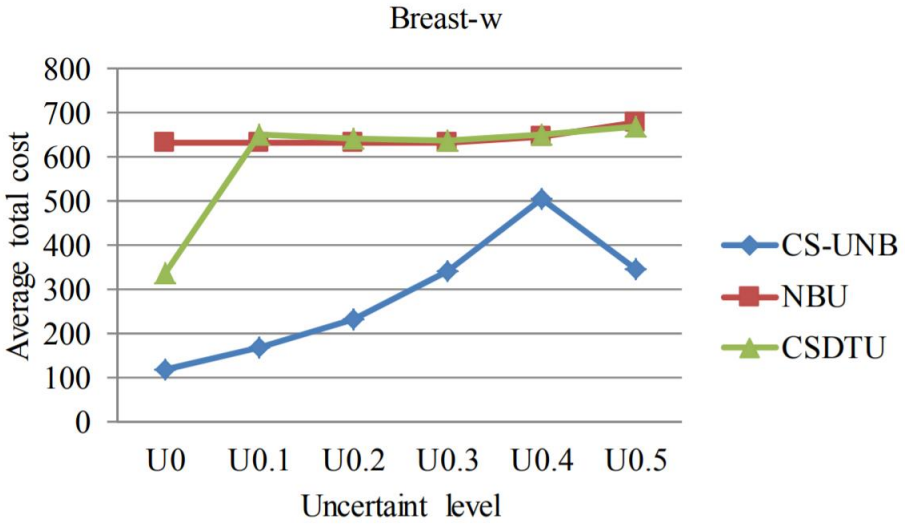


Fig. 3. The second subplot of Figure 1 in the original paper, showing the relationship between the Average Total Cost and the Uncertainty

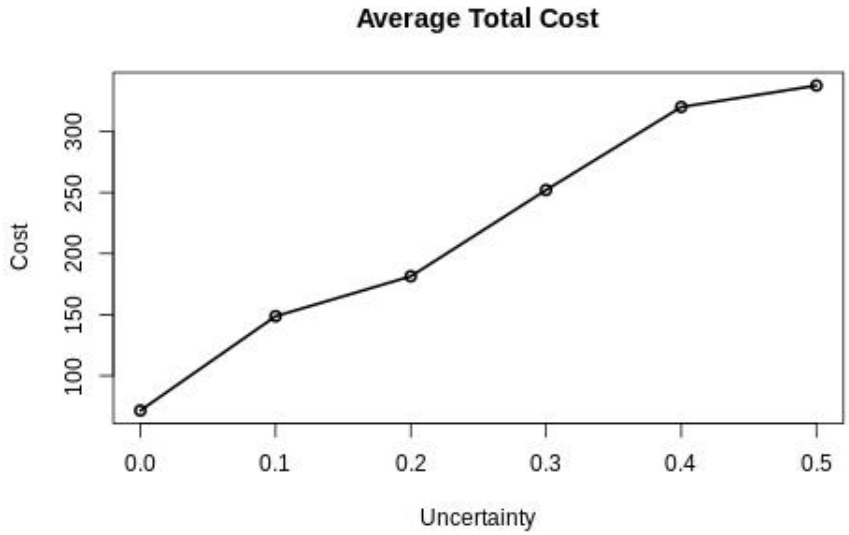


Fig. 4. Our reproduction of the CSUNB line in Figure 3

4.1 Implemented Algorithms

4.1.1 Decision Tree

Decision tree is a method that creates a tree-like model that predicts the value of a target variable based on several input features. Each interior node corresponds to one of the input variables. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A tree can be built by splitting the source set of training data samples into subsets based on attribute value tests. This procedure is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees.

Different algorithms for constructing the tree use different metrics. These generally measure the homogeneity of the target variable within the subsets. The most popular metrics are Gini Index, Information Gain, and Variance Reduction.

4.1.2 Bagging and Boosting

Followed by Decision Tree, we carried out Bagging and Boosting on the dataset. Bagging is a bootstrap model which randomly generates L sets of cardinality N from the original set Z with replacement and corrects the optimistic bias of the R-Method. Referred to as Bootstrap Aggregation, Bagging creates Bootstrap samples of a training set using sampling with replacement. Each bootstrap sample trains a different component of the base classifier and the Classification is done by plurality voting. Regression is carried out by averaging. This technique works for unstable classifiers such as Neural Networks and Decision Trees. Also, Random Forests Algorithm is a Bagging-Variant and is a general class of ensemble building methods using a decision tree as a base classifier. The main reason for error in learning is due to noise, bias and variance. Noise is error by the target function whereas bias occurs when the algorithm cannot learn the target. Variance comes from the sampling, and how it affects the learning algorithm. These errors are reduced by Bagging and Averaging over bootstrap samples could reduce errors from variance especially considering unstable classifiers.

Boosting is a technique which combines multiple base classifiers whose combined performance is significantly better than that of any of the base classifiers. It follows a sequential training of weak learners and each base classifier is trained on weighted data based on the previous classifiers performance.

4.1.3 AdaBoost

For the boosting algorithm, there are two problems: How to adjust the training set so that the weak classifier trained on the training set can be performed; How to combine the weak classifiers obtained by training to form a strong classifier. For the above two problems, the AdaBoost algorithm has been proposed. AdaBoost uses the weighted training data instead of the randomly selected training samples, so that the training focus is concentrated on the training data samples who are more difficult to classify. It also combines the weak classifiers and use a weighted voting. A weak classifier with good classification effect will have a larger weight, while a classifier with poor classification effect has a smaller weight.

4.1.4 Random Forest

Random Forest (RF) is a kind of ensemble learning. Its basic unit is decision tree. For the classification problem, each decision tree is a classifier. Then for an input sample, N trees will have N classification results. The random forest integrates all the classification voting results, and specifies the category with the most votes as the final output. So random forest is actually using bagging.

Firstly, RF uses the CART decision tree as a weak classifier. At the same time, it only randomly selects a few features to generate each tree. Generally, the square root of the total number of features

is selected by default, while the general CART tree will select all the features to train. Because of the randomness, it is very effective to reduce the variance of the model. Therefore, random forests generally do not need additional pruning, and can achieve lower variance. In addition, the forest makes the results of the overall model more accurate.

4.1.5 Logistic Regression

Logistic Regression is a statistical method that analyses a dataset having one or more independent variables to determine an outcome. The outcome is measured with a dichotomous variable, which means there could be two possible outcomes (True or False). Logistic Regression is used when finding the best fit model to describe the relationship between the dependent variable and set of independent variables.

The Logistic Regression's curve value is between 0 and 1. The Linear model equation is:

$$y = \beta_0 + \beta_1 x \tag{7}$$

where β_0 is the constant and β_1 is the slope which defines the steepness of the curve. The Logistic model equation is in terms of the odds ratio. Hence, as seen:

$$p = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} \tag{8}$$

4.1.6 LDA and QDA

[2]

The Bayesian classifier divides an observation into the largest class of $p_k(x)$, which has the lowest error rate among all classifiers. To calculate the posterior probability $p_k(x)$, we need a way to estimate density function of X for an observation that comes from the k-th class. Linear Discriminant Analysis based on the assumption that the observation in k-th class is derived from the multivariate Gaussian distribution with a class-specific mean vector and a common covariance matrix.

The Gaussian density in LDA is defined as

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{9}$$

Bayesian classifier will divide observation $X = x$ into the class for which

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}{\mu_k}^T\Sigma^{-1}\mu_k + log\pi_k \tag{10}$$

is largest. We can find the discriminant function is a linear function of x.

Same as LDA, QDA (Quadratic Discriminat Analysis) also assumes that the observations for each class are derived from the Gaussian distribution. But in QDA, each class has its own covariance matrix. Discriminant function for QDA is a quadratic function of x.

$$\delta_k(x) = -\frac{1}{2}x^T{\Sigma_k}^{-1}x + x^T{\Sigma_k}^{-1}\mu_k - \frac{1}{2}{\mu_k}^T{\Sigma_k}^{-1}\mu_k - \frac{1}{2}log|\Sigma_k| + log\pi_k \tag{11}$$

4.1.7 Support Vector Machines

A Support Vector Machine (SVM) is a supervised discriminative classifier formally defined by a separating hyperplane that separates most data samples with the largest margin. A kernel can be applied to make the separating hyperplane non-linear. The most common non-linear kernel is a radial kernel.

4.1.8 Neural Networks

Neural Networks, or sometimes referred to as Artificial Neural Networks are computing systems inspired by the biological neural systems that involve animal brains. Such systems learn to perform tasks without being programmed with any task-specific rules or any prior knowledge of the task. They automatically generate identifying characteristics from the learning material that they process.

An ANN architecture is usually based on a collection of densely connected units or nodes called artificial neurons, which loosely mimic the behavior of the neurons in a biological brain. Each connection, similar to the synapses in a brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives the signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear activation function of the sum of its inputs. Artificial neurons typically have a weight that adjusts itself as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs.

4.1.9 Cost-Sensitive Subset Selection

Algorithm 2 Cost-Sensitive Forward Subset Selection

```
Tested attribute set A_t = ∅
Untested attribute set A_u = Attribute set
for step ← 1 to Total number of attributes do
    for attribute a in A_u do
        Fit a model with A_t + a
        Calculate cost of the model
    end for
    Select A with the lowest cost
    Include A in A_t
    Exclude A in A_u
end for
Plot the lowest cost of each step
```

The most important and novel part of our work is to combine the Cost-Sensitive with the basic classification methods mentioned above. Inspired by the greedy Forward Subset Selection of Linear Regression, we propose a similar procedure and name it as Cost-Sensitive Forward Subset Selection. The pseudo code of it is shown as Algorithm 2. This algorithm can be explained as follows: We keep a tested attribute set and an untested one; first let all attributes in the untested attribute set, and the tested attribute set be empty. In each step, add one attribute to tested set to fit a model and calculate the cost of the model, save the one with the lowest total cost, marked the selected attribute as A. Then, exclude A in untested attribute set, and include A in tested attribute set. After all of the attributes go into the tested set with iterations, end for loop and plot the lowest cost for each step.

4.2 Proposed Algorithms

4.2.1 Extended Decision Tree

Classical Decision Tree algorithms are introduced in Section 4.1.1. Generally, the node splitting criteria is designed to maximize the accuracy. But the splitting criteria may not work well in the cost-sensitive problems. For example, in the tree-growing step, if a leaf node has 9 positive samples and 1 negative sample with a positive prediction, the Classical Decision Tree may stop growing. But if in the problem setting, the cost of False Positive is much higher than the test costs, the node should be split to obtain an optimal result. Therefore, a more effective criterion of node splitting should be the decrease of total cost, shown in Figure 5.

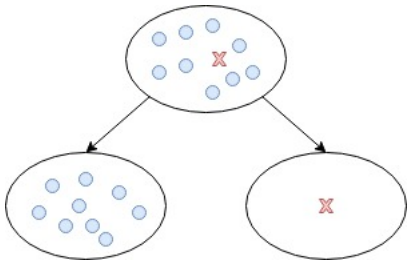


Fig. 5. Node splitting

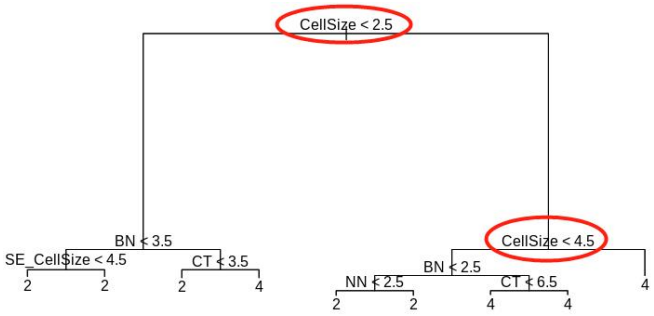


Fig. 6. For such a Decision Tree, if we get a data sample with CellSize greater than 4, we do not need to test other attributes, though a bunch of attributes are used to fit the tree.

Another improvement lies in the test phase. When testing each data sample, we can follow the path and only accumulate cost of used attributes instead of all the attributes that are used to fit the model. Take the Decision Tree in Figure 6 as an example, the attributes of CellSize, BN, SE_CellSize, CT, NN are utilized to fit the tree, but if we have a data sample with CellSize greater than 4, it goes along the path of nodes circled in the figure, and reaches a leaf of node of prediction 4 without testing attributes other than CellSize, in fact. Thus, for this specific data sample, only the test cost of the attribute CellSize is necessary. Such calculation will significantly decrease the actual test cost than the program we implemented.

4.2.2 Extended Neural Networks

One of the drawbacks of Neural Networks described in Section 4.1.8 is that the input shape of the network is not flexible. Intuitively, if we want to figure out the subset of features that minimizes the total cost, we need to first iterate through different numbers of features, and in each iteration, try every combination of features. The computation complexity is obviously exponential. Considering the long training time of the networks, the algorithm is inpractical. In order to make the subset selection trainable, we can add an extra binary embedding layer just after the inputs, as shown in Figure 7.

One source of intuition of this kind of embedding layer is Dropout [7], a common regularization technique at present to prevent overfitting. Dropout "drops" a set of neurons by setting their values to 0 with a certain probability so that during training after Dropout, different sets of intermediate neurons (features) are selected to fit the data so as to prevent overfitting.

Another source of intuition is Word Embedding [3]. Traditionally, Natural Language Processing researchers treat words as one-hot vectors. The one-hot encoding breaks the relative relationship between words and does not have any suggestions of the properties of words. So an embedding layer is trained to map the one-hot vector of the word to a certain length of vector. The word embedding encoding pre-trained from a specific problem may still work on another problem. The encoding also tends to express the property of the word and its relationship with other words as well.

Our proposed embedding layer after all, takes all of the attributes as input, and outputs a vector of binary neurons of the same lengths. Inside the layer, the input forward propagates through an affine

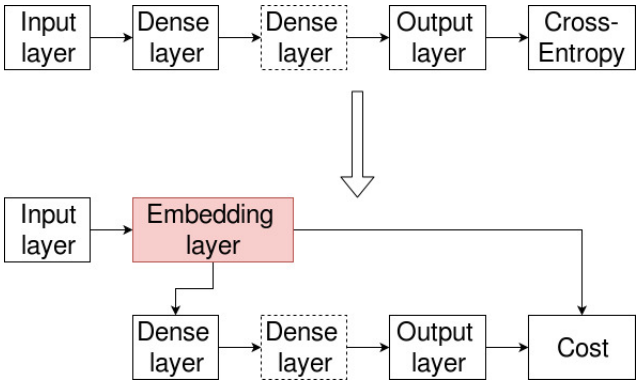


Fig. 7. Structure of the implemented Neural Network proposed Neural Network with Embedding Layer

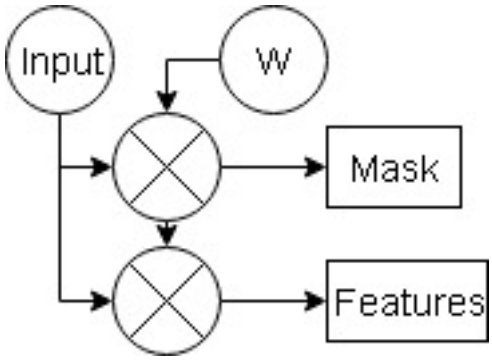


Fig. 8. Block Diagram of the Embedding Layer

layer with an activation that maps the values to 1 and 0 by comparing them with a threshold. After that, the test cost can be calculated from the dot product of the output mask and the test cost vector, and the attribute vector can be calculated from the dot product of the output mask and the original input. The attribute vector is used to continue the classification training. In the end, the loss function combines the sum of the test cost vector and the misclassification cost from the output layer of the network, and back-propagates to train the network. The proposed building block diagram is shown in Figure 8.

5 EXPERIMENTS

Three experiments are done to get results of fitting ability, generalization ability, minimal total cost of each method.

5.1 Fitting Ability

False positive and false negative are important terms to evaluate the result of binary classification problems. Considering the ten algorithms on Breast Cancer Wisconsin Dataset, if the predicted result of a sample is benign, while the actual result in dataset of it is malignant, it should be included in false negative result set. On the other hand, if the predicted result of a sample is malignant, while the actual result in dataset of it is benign, it should be included in false negative result set.

In the first experiment, we use all data samples in the dataset to train the model, and all to test the model. In this process, the number of false positive and false negative results are the factor we mainly used to evaluate the fitting ability of one algorithm. The number of false positive and false negative results are shown in Figure 9.

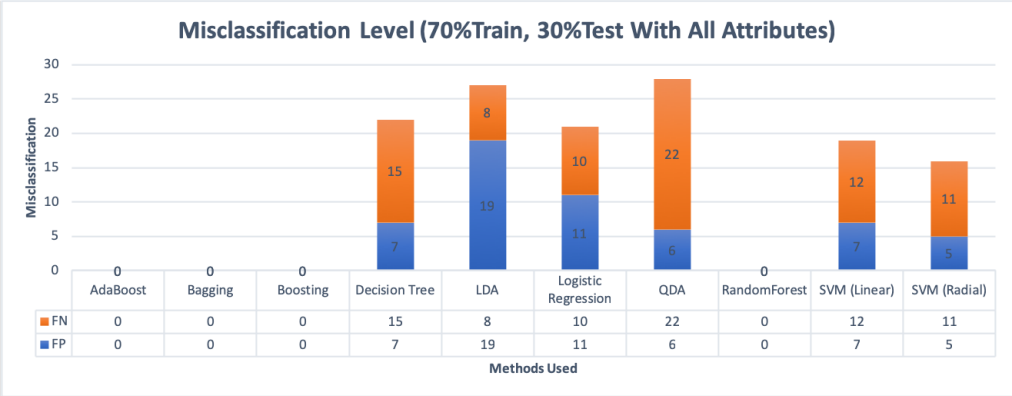


Fig. 9. Number of FP and FN Results of Ten Algorithms

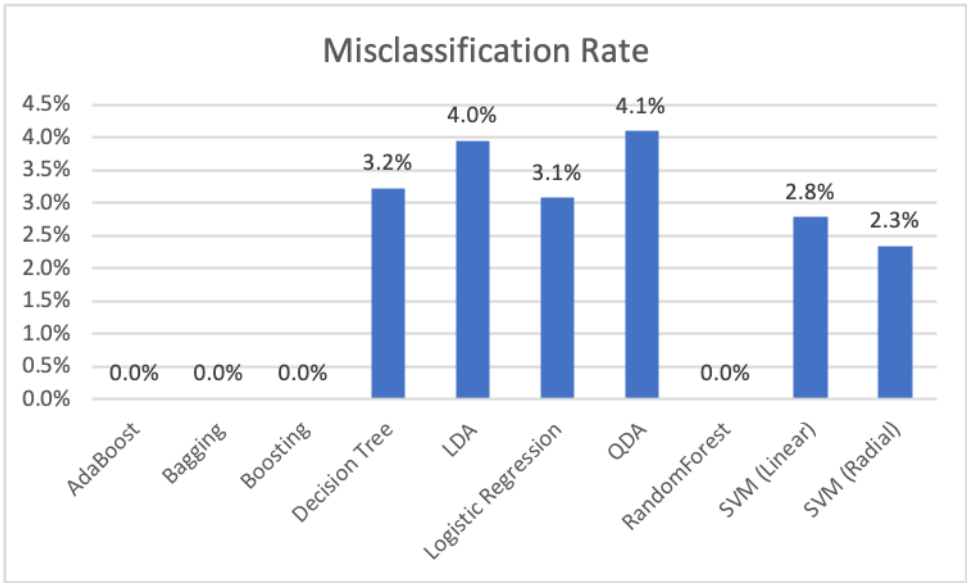


Fig. 10. Misclassification Rate of Ten Algorithms

Additionally, as shown in 12, When calculate misclassification rate of a model, we should first get the sum of false positive results and false negative results, and the number of test samples. Considering the Breast Cancer Wisconsin Dataset, the number of test samples is 683. Figure 10 shows the misclassification rate of algorithms. In comparison of ten algorithms, AdaBoost, Bagging, Boosting, Random Forest appear to have zero misclassification cases, while other algorithms show their misclassification problem more or less. Among all of them, QDA suffers from this problem the most in that it has misclassification rate of 4.1%. However, it can still be considered acceptable.

$$misclassificationrate = (FP + FN)/(P + N) \tag{12}$$

In conclusion, although ten algorithms we used suffer from different levels of misclassification, we can assume that they all have acceptable fitting ability. That is to say, algorithms themselves will be capable to conduct the experiments afterwards.

5.2 Generalization Ability

Then in the second experiment, in order to get the generalization ability of ten methods, we exclude test cost and pay attention only on the misclassification cost. So we use 70% samples in the dataset

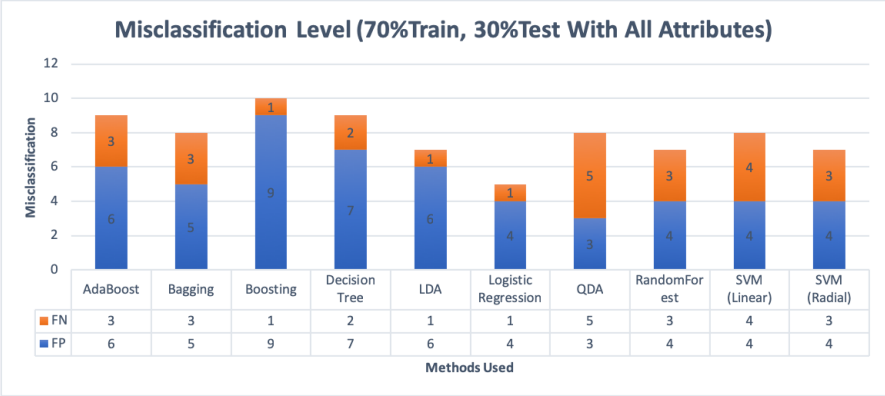


Fig. 11. Number of FP and FN Results of Ten Algorithms (use 70%-30% split)

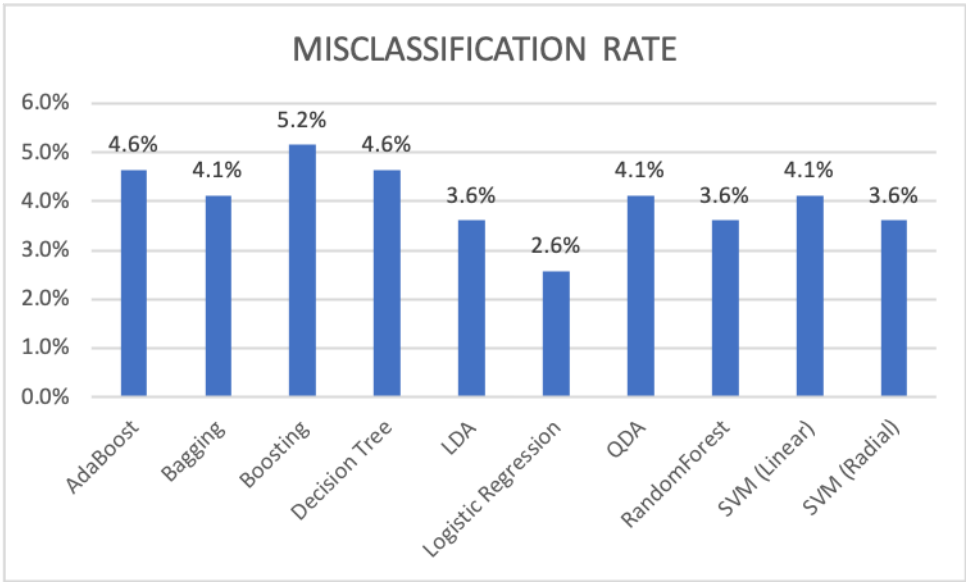


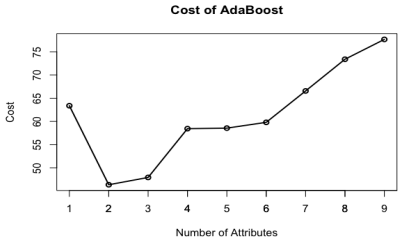
Fig. 12. Misclassification Rate of Algorithms(use 70%-30% split)

to train the model, and the other 30% to test. As elaborated in experiment A, it is easy to conduct experiment and get the sum of false positive results and false negative results. Results are shown in Figure 11. From this figure, Logistic Regression performs best in that it only has 4 false positive results and 1 false negative result. Meanwhile, Boosting with 9 false positive results and 1 false negative result, performs worst.

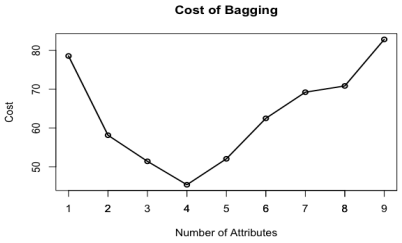
Then, in the same way, the misclassification rates of ten algorithms using 70%-30% split are calculated. In this case, the number of samples in test set should be 194. Figure 12 shows the result.

5.3 Cost-Sensitive Subset Selection

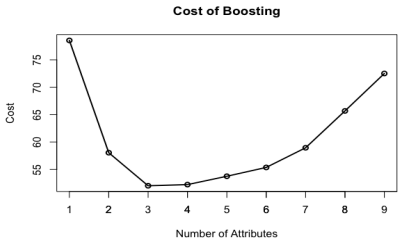
The ten plots are shown in Figure 13. The trends of each plot are approximately going down at first and then going up. It is mainly because of the trade-off between the test cost and the misclassification cost of a model. With the number of attributes increases, the test cost increases and the misclassification cost decreases. Therefore, the sum of them may reach a lowest total sum of cost in a number of attribute. For each algorithm, though the plot may have local minimum (for example, the Radial SVM algorithm result shown in Figure (x)), or local maximum, the main trend of it are the same. That is one way to vividly explain the trade-off. Also, to be clear, in this experiment, we



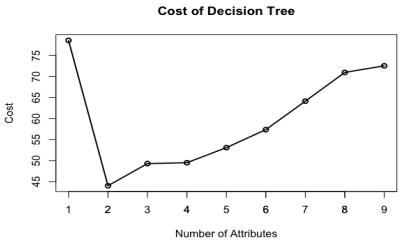
(i) Cost of AdaBoost



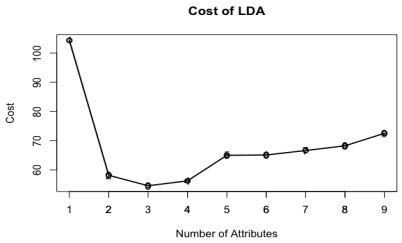
(ii) Cost of Bagging



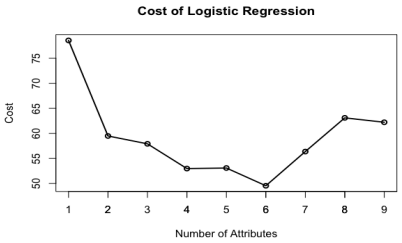
(iii) Cost of Boosting



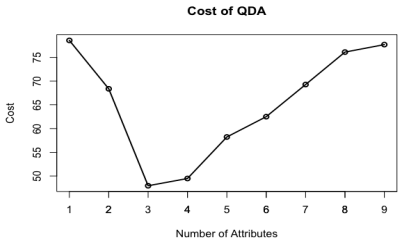
(iv) Cost of Decision Tree



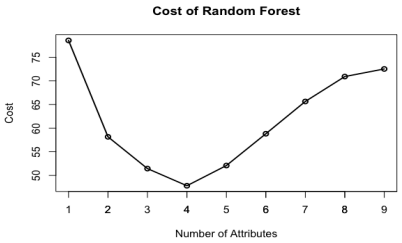
(v) Cost of LDA



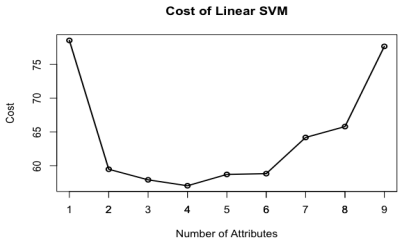
(vi) Cost of Logistic Regression



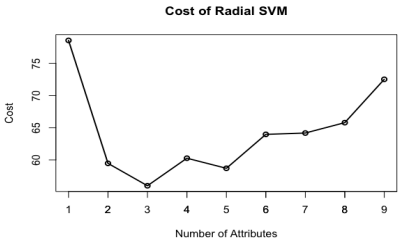
(vii) Cost of QDA



(viii) Cost of Random Forest



(ix) Cost of Linear SVM



(x) Cost of Radial SVM

Fig. 13. Cost of Ten Algorithms combined with Cost-Sensitive Algorithm

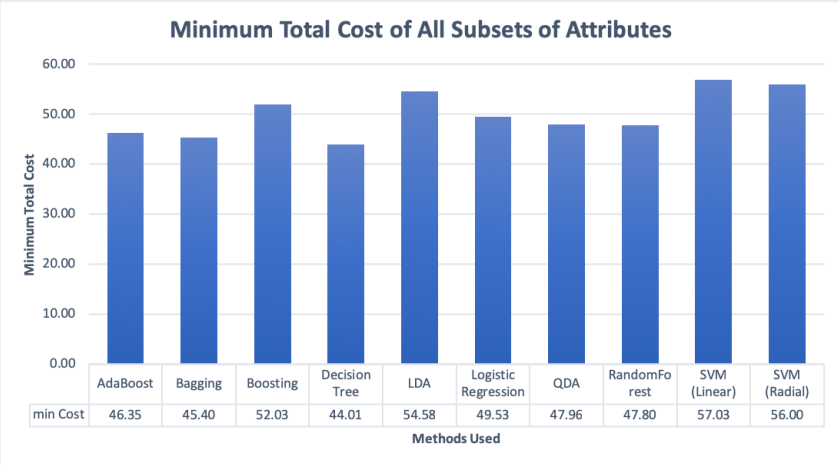


Fig. 14. Minimum Total Cost of All Subsets of Attributes

set the test cost to be [0, 10] Uniformly distributed, and for the misclassification cost, 1000 for FP and 1000 for FN. If we change the cost value set up in the very beginning, the results may differ. In reality, these values are set based on problems we are solving.

In order to compare all methods in the same scale, Figure 14 shows the minimum total cost of all subsets of attributes of each algorithm. Among all of them, Decision Tree performs the best in that it only has 44 value of minimum total cost.

6 FUTURE WORK

6.1 Proposed Algorithms

Algorithms in Section 4.2 are what we consider effective to solve the cost-sensitive classification problem. Yet the successful implementation of the algorithms lies in both careful design of programs and a large-scale and high-quality dataset. The implementation and experiments of the algorithms are left for one part of the future works.

6.2 True Test and Misclassification Costs

Throughout the reproduction and our experiments, we treat test cost of each attribute as random variables following uniform distribution. This assumption makes the results of our experiments meaningless in practical. The subset of attributes we chose from Algorithm 3.1 and 2 tends to lean on attributes with a low test cost, because the misclassification costs are distributed among all the data samples but the test cost is added to each sample. However the test cost of those attributes may not be small in reality. So in order to make the algorithms more practical, we need investigations of the true test costs from domain experts.

On the other hand, different people have different preference towards the trade-off between misclassification cost and test cost. Generally, people with more wealth tend to require more precise prescriptions from the medical institutions. Moreover, doctors, hospitals, insurance companies, and patients may have different bias towards cost of False Positive and that of False Negative. A precise knowledge of the true cost values allows the algorithms to work better.

7 CONCLUSIONS

In this project, we focus on Cost-Sensitive Classification problem that handles the trade-off between test cost and misclassification cost. We have reproduced results of a Cost-Sensitive classifier based on Naive Bayes and have developed a successful subset selection algorithm that is valid for almost all classification algorithms. We have also proposed algorithms for the future work.

ACKNOWLEDGEMENT

We would like to express our gratitude here to Professor Predrag Jelenkovic in the Electrical Engineering Department of Columbia University, who provided essential suggestions and guidance for us.

References

[1] Adam Conner-Simons. Using artificial intelligence to improve early breast cancer detection. *MIT CSAIL*, 2017.

[2] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2017.

[3] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[4] Yun Liu, Krishna Gadepalli, and Mohammad Norouzi. Detecting cancer metastases on gigapixel pathology images. *Google Brain*, 2017.

[5] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[6] Zheng Ping, Yuchao Xia, and Tiansheng Shen. A microscopic landscape of the invasive breast cancer genome. *Nature*, 2016.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[8] William H. Wolberg and O.L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87:9193–9196, 1990.

[9] Xing Zhang, Mei Li, Yang Zhang, and Jifeng Ning. Cost-sensitive naïve bayes classification of uncertain data. *JCP*, 9(8):1897–1903, 2014.