# Mini Project Final Report

Dawei Geng

## Background

In machine learning, classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category is known. An algorithm that implements classification, especially in a concrete implementation, is known as a classifier

In this project, I did research on decision tree, mainly ID3 algorithm. Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

The reason why I am interested in decision tree is that it is very easy to understand compared to other classification algorithm like support vector machine so that it is widely used in the real world. Also, it has pretty good precision when used in classification, especially after improvement methods such as pruning are implemented. Like Regression, it is known as supervised learning algorithm so after training it, the model will be super quick to give you the prediction when you consult the model.

## Research on ID3 Algorithm

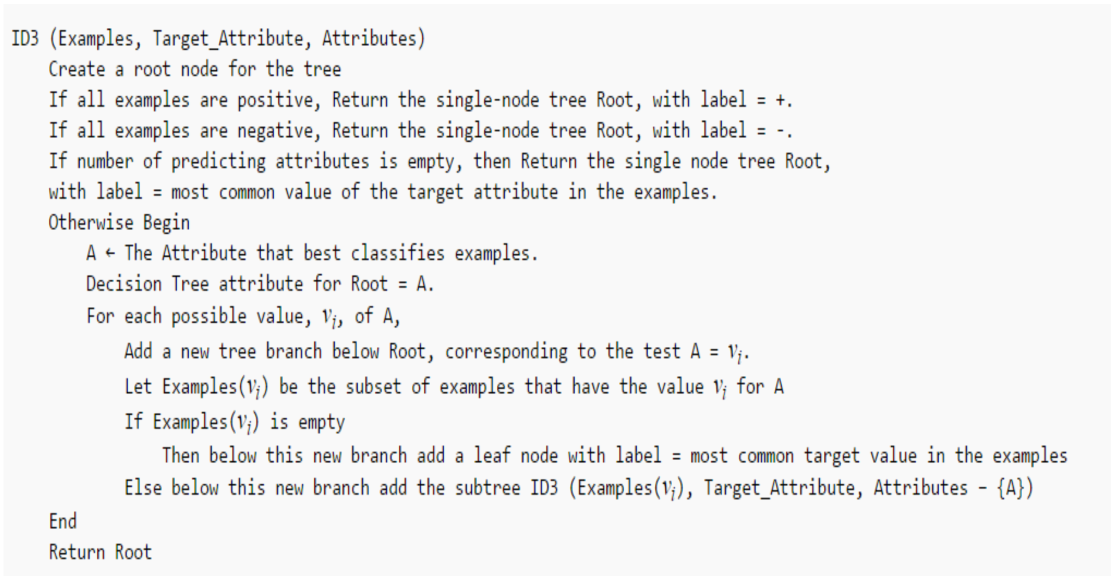
ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. The basic idea of ID3 algorithm is to create a decision tree of given set, by using top-down greedy search to check each attribute at every tree node.

For building ID3 algorithm decision tree consists of nodes and arcs or sweeps which connect nodes. To make a decision, one starts at the root node, and asks questions to determine which arc to follow, until one reaches a leaf node and the decision is made.

Steps to build an ID3 decision tree are:

1. Calculate the entropy of every attribute in the data set
2. Split the dataset into subsets using the attribute for which information gain is maximized.
3. Make a decision tree node containing that attribute.
4. Do recursions on subsets using remaining attributes.

These steps are illustrated in my c code. Here is the Pseudocode:

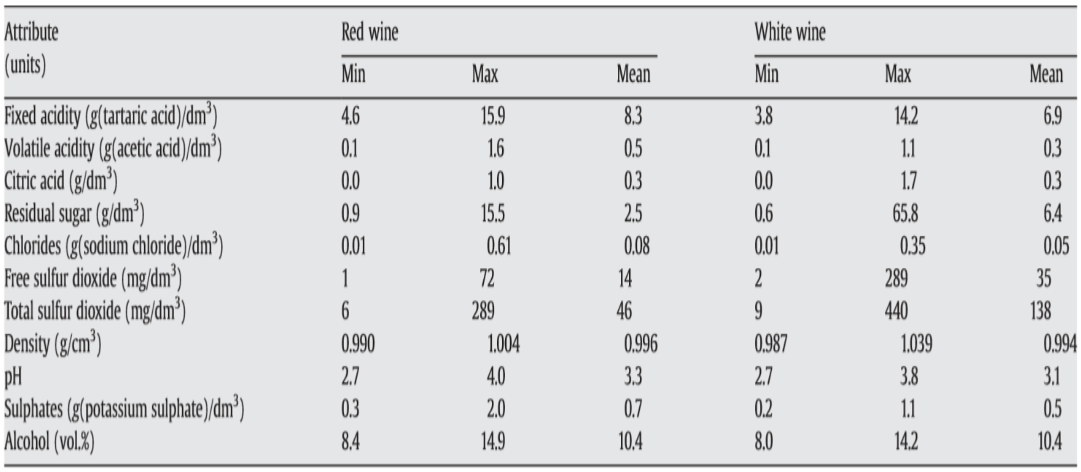


```
ID3 (Examples, Target_Attribute, Attributes)
    Create a root node for the tree
    If all examples are positive, Return the single-node tree Root, with label = +.
    If all examples are negative, Return the single-node tree Root, with label = -.
    If number of predicting attributes is empty, then Return the single node tree Root,
    with label = most common value of the target attribute in the examples.
    Otherwise Begin
        A ← The Attribute that best classifies examples.
        Decision Tree attribute for Root = A.
        For each possible value, $v_i$, of A,
            Add a new tree branch below Root, corresponding to the test A = $v_i$.
            Let Examples($v_i$) be the subset of examples that have the value $v_i$ for A
            If Examples($v_i$) is empty
                Then below this new branch add a leaf node with label = most common target value in the examples
            Else below this new branch add the subtree ID3 (Examples($v_i$), Target_Attribute, Attributes – {A})
    End
    Return Root
```

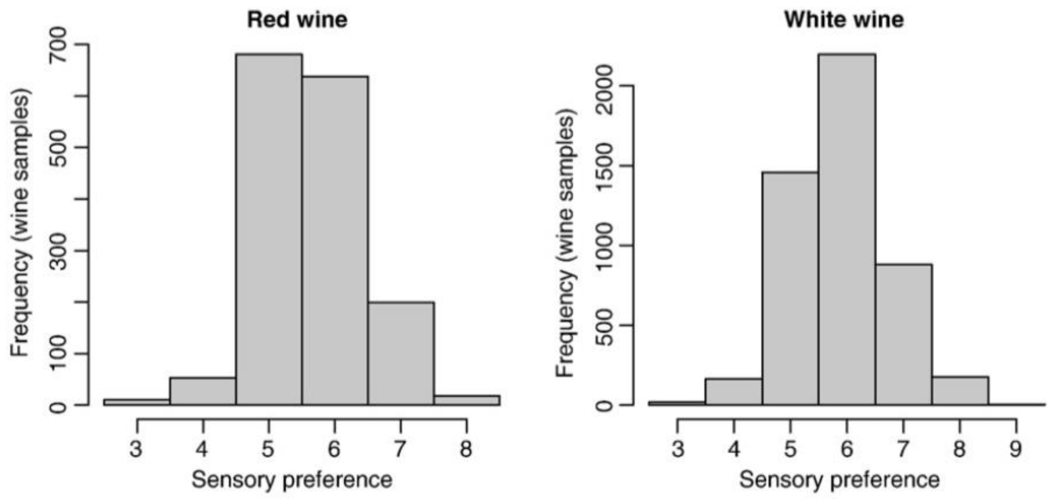
This is a greedy algorithm which use recursive manner of top-down, divide and conquer to construct a decision tree. The termination condition of recursion is: all samples within a node are of the same category. If no attribute can be used to divide current sample set, then voting principle is used to make it a Compulsory leaf node, and mark it with the category of having the most number of sample types.

## Wine Quality Dataset

This dataset comes from University of California at Irvine machine learning repository. The inputs of the data include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent). Their description statistics are described as below: (here we focus on white wine)



| Attribute (units) | Red wine | | | White wine | | |
|---|---|---|---|---|---|---|
| | Min | Max | Mean | Min | Max | Mean |
| Fixed acidity (g(tartaric acid)/dm$^3$) | 4.6 | 15.9 | 8.3 | 3.8 | 14.2 | 6.9 |
| Volatile acidity (g(acetic acid)/dm$^3$) | 0.1 | 1.6 | 0.5 | 0.1 | 1.1 | 0.3 |
| Citric acid (g/dm$^3$) | 0.0 | 1.0 | 0.3 | 0.0 | 1.7 | 0.3 |
| Residual sugar (g/dm$^3$) | 0.9 | 15.5 | 2.5 | 0.6 | 65.8 | 6.4 |
| Chlorides (g(sodium chloride)/dm$^3$) | 0.01 | 0.61 | 0.08 | 0.01 | 0.35 | 0.05 |
| Free sulfur dioxide (mg/dm$^3$) | 1 | 72 | 14 | 2 | 289 | 35 |
| Total sulfur dioxide (mg/dm$^3$) | 6 | 289 | 46 | 9 | 440 | 138 |
| Density (g/cm$^3$) | 0.990 | 1.004 | 0.996 | 0.987 | 1.039 | 0.994 |
| pH | 2.7 | 4.0 | 3.3 | 2.7 | 3.8 | 3.1 |
| Sulphates (g(potassium sulphate)/dm$^3$) | 0.3 | 2.0 | 0.7 | 0.2 | 1.1 | 0.5 |
| Alcohol (vol.%) | 8.4 | 14.9 | 10.4 | 8.0 | 14.2 | 10.4 |



In order to twist the dataset to the one we could use in decision tree, I classify the target variable -- wine quality into 0 and 1, 0 representing wine's quality below 7, 1 representing wine's quality greater or equal to 7. This could be justified by the histogram above, only a few wines fall into the range greater or equal 7.

This is how I end up with 12 variables, in which binary variable - quality I just created will be used as target value, other 11 variables will be used for decision tree to split on.

## C Code

To process the data, I use matrix structure to store them. Because these functions are basically what we did in the first assignment except for the ReadData function. I will skip the detailed explanation here. Here are those function prototype:

In matrix.c:

- **typedef double** matrix;**
- matrix **make_matrix(int n1, int n2);**
- **int free_matrix(int n1, int n2, matrix a);**
- **void matrix_print(int n1, int n2, matrix a);**
- matrix **ReadData(FILE* dataset,unsigned line, unsigned column);**

Note: ReadData function will read the dataset line by line, store them into matrix, and then return.

In DecisionTreeFunc.c, following functions constitute ID3 algorithm.

- **typedef struct node1 node1;**
- **struct node1* make_tree();**
- **void create_tree(matrix m,struct node1* array1,int row, int column,int pos, int num);**
- **int classify(struct node1* array2, double* m,int level);**
- **void free_tree(struct node1* array2);**

Explanation:

Node1 is the structure I used to store the entire decision tree. Make_tree function create an empty tree. Create_tree function grows the tree to the point all the elements in the node belong to the same category. Classify function are used when we want to consult decision tree with new data. Free_tree function is used to free the memory of the entire tree. The process of growing the tree is essentially a pre-order tree traversal, one kind of depth first search algorithm.

## Analysis

My analysis consists of two parts. One part is to compare the prediction accuracy with algorithms defined in Weka using 10 fold cross validation. Find the most appropriate algorithm to classify this dataset. The other part is to analyze OpenMP results to see how parallel computing could speed up the algorithm. I ran my code on Jinx, where I got my results.

## First part—ID3 algorithm

Here I focus on algorithm itself and analyze the accuracy and time. 10 fold cross validation accuracies I calculated in C are tabled below:

| 10 fold | Accuracy | Time to build model |
|---|---|---|
| 1 fold | 0.644898 | 2.76 |
| 2 fold | 0.706122 | 2.75 |
| 3 fold | 0.708163 | 2.76 |
| 4 fold | 0.687755 | 2.76 |
| 5 fold | 0.679592 | 2.76 |
| 6 fold | 0.663265 | 2.76 |
| 7 fold | 0.636735 | 2.77 |
| 8 fold | 0.563265 | 2.77 |
| 9 fold | 0.717791 | 2.77 |
| 10 fold | 0.689162 | 2.76 |
| Average | 0.669675 | 2.76 |

Compare with modern classifiers developed in Weka

| Modern Classifiers | Accuracy | Time to build model |
|---|---|---|
| Support Vector Machine | 82.9318 % | 343.87 |
| J48 Decision Tree | 78.3585 % | 0.16 |
| Logistics Regression | 74.8673 % | 95.44 |
| RBFNetwork | 81.9927 % | 0.28 |
| AdaBoostM1 | 78.3585 % | 0.11 |
| Bagging | 80.1143 % | 1.18 |
| ID3 | 60.1062 % | 0.57 |
| Random Forest | 86.668 % | 1.96 |

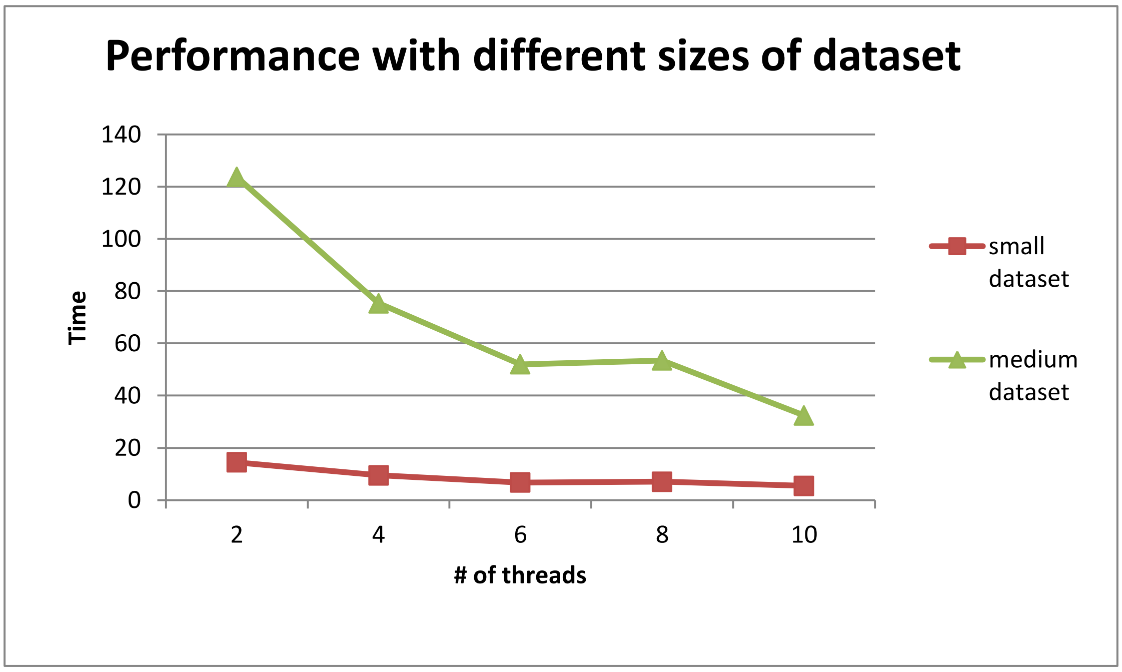
The ID3 algorithm I developed in C outperforms that in Weka in terms of the accuracy.

But my version takes longer to build the model, which might be an issue when I have large amount of data. Because ID3 is very basic decision tree, so called precursor, so it is reasonable that other tree algorithms perform somewhat better than ID3. As we could see from the table, Random Forest and Support Vector Machine are best algorithm to classify this dataset in terms of accuracy. Random Forest performs best considering the little time it takes to build the model.

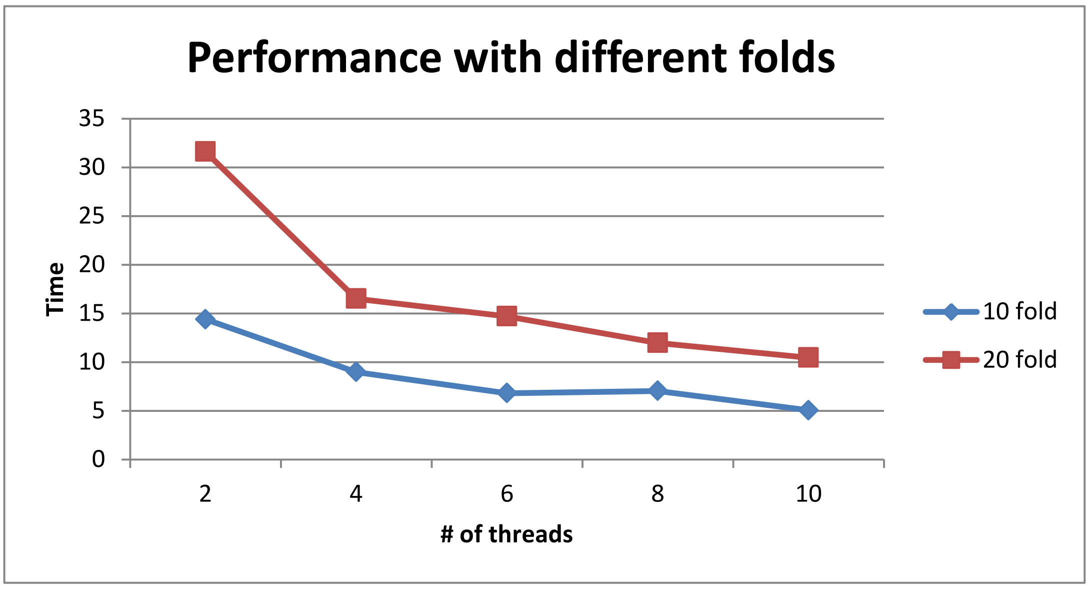
## Second Part—Parallel Computing

In this part, I focus on how parallel computing improves the speed of ID3 algorithm and cross validation.

I created another dataset with 3 times number of data entry than the number of wine data entry(4898*12), which I considered as medium dataset(14694*12 data). I plot time elapsed(Y) for building and querying tree versus number of threads(X). As you can see, with more threads, speed could be increased significantly and time significantly reduced regardless of the size of data as long as enough threads are used.



I also plot time vs number of fold in the cross-validation. I notice that when we doing cross validation, it’s better to set number of threads to be rather big and also a factor of number of fold so that number of loops could be reduced and overhead for the thread is lower.



## Appendix

- Mitchell, Tom M. Machine Learning. McGraw-Hill, 1997. pp. 55–58.
- Grzymala-Busse, Jerzy W. "Selected Algorithms of Machine Learning from Examples." Fundamenta Informaticae 18, (1993): 193–207.
- Quinlan, J. R. (1987). "Simplifying decision trees". International Journal of Man-Machine Studies 27 (3): 221.
- Rokach, Lior; Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc. ISBN 978-9812771711.
- Hastie, T., Tibshirani, R., Friedman, J. H. (2001). The elements of statistical learning : Data mining, inference, and prediction. New York: Springer Verlag.