**Project Report**

1. **Description on the question and the data**

We choose data set 2 to build a classification tree. This project aims at building a classification tree from train data set, and use the tree to predict whether the quality score of each wine in test data set is above 6 or not. After prediction, we would report the accuracy of prediction.

In train data set and test data set, each type of wine has 11 features. And all feature values are continuous type, which means a data preprocessing method based on multi-continuous attribute discretization would be adopted later. All feature values are given, which means no information is missing, with value “NA”. Thus, we don’t need to find a method to revised the data. At the end of the tree, each type of wine would be labeled as “<=6” or “>6” based on whether its quality value is greater than 6 or not.

1. **Review of some of the approaches that we tried/thought about trying.** 
   1. **Review and comparison of approaches we thought about trying in the tree-growing process**

We first thought about choosing one approach from ID3, C4.5 and CART algorithms. We made a comparison on these three approaches.

|  |  |  |
| --- | --- | --- |
| Algorithm | Basic ideas | deficiency |
| ID3 | Select the attribute which has the smallest entropy (or largest information gain) value by calculating H(S):    The data set is then split by the selected attribute to produce subsets of the data. | 1. The entropy model used in selecting features and attribute values make it less efficiency. 2. Features with more values are more likely to be chosen when splitting since it complies with the smallest entropy gain. |
| C4.5 | Select the attribute which has the largest information gain ratio by calculating H(D):    Then create a threshold and split the data set into those whose attribute value is above the threshold and those that are less than or equal to it, so that it can deal with continuous type. | 1. The entropy model used in selecting features and threshold make it less efficiency. 2. C4.5 create a multi-branches tree, which would be more complicated than binary trees. |
| CART | Select the attribute which has the smallest  Gini impurity calculated by  Then the data set is split by the selected attribute to produce subsets of the data. | 1. Data in chronological order should be preprocessed. 2. Since CART use the idea of dichotomy, too much features will reduce accuracy, which means a pruning process is required. |

Table 1

* 1. **Some approaches we tried or thought about trying in pruning tree**

The end condition of theoriginal tree is there is only one piece of data in that branch or all pieces of data in that branch are “<=6” or “>6”. This tree is very deep because all data of each labels has been considered. The tree will probably overfit since some noisy data which is uncorrelated or even mislead to the prediction is considered and divided. Thus, in order to lower the probability of overfitting, we need to prune the tree to increase its generalization ability, to make the tree fits for any data set. We thought about two main ways to prune the tree:

1. **Pre-pruning**

Set a maximum depth of the tree or set a minimum number of the data sizes of each node. Once the layer of the tree reaches the maximum depth or there isn’t enough data in one node, even if we haven’t precisely divide each data into a certain leaf node and can’t use only one label to summarize all these data accurately, the tree should stop growing.

1. **Post-pruning**

We tried three ways to do the post-pruning. They are Reduced-Error pruning(REP), Pessimistic-Error pruning(PEP), Cost-Complexity pruning(CCP).

**①REP (Reduced-Error Pruning)**

1. For every sub-tree,

2. For each node which is not a leaf node, calculate the error rate of the current tree (or sub tree). Assume to convert the current sub tree to a leaf node, and calculate the error rate after pruning.

3. If the error rate after pruning is smaller than the error rate before pruning, cut the sub- tree. That is, convert it to a node.

**②PEP (Pessimistic-Error Pruning)**

1. error rate:

E: number of incorrect predictions under the sub-tree

0.5: half-unit correction for continuity

L: the number of leaf nodes under the sub-tree

N: the number of examples of data in the sub-tree

2. For every node that is not a leaf node, calculate the error rate before cutting down the sub-tree and use **p** to represent it. Consider the error distribution as a binomial distribution, then use **p** to calculate the expectation of the incorrect predictions **E₁** in the sub-tree and the standard deviation **std** of the error distribution.

3. Assume cutting down the sub-tree and calculate the error rate of the node, denoted by **e**. Consider the error distribution as a binomial distribution, then use **p** to calculate the expectation of the incorrect predictions **E₂** in the sub-tree.

4. If E₁ > E₂ + std, that is, the expectation of the number of errors before pruning is larger than the expectation of the number of errors after pruning, convert the current subtree into a node. Otherwise, continue to analyze the sub-trees of the node.

**③CCP (Cost-Complexity Pruning)**

1. α = (C(t) - C(Tt)) / (|Tt| - 1)

C(t): The error cost of node t

C(t) = misclassification rate of node t \* the proportion of sample sizes in node t

C(Tt): The error cost of every leaf node under the node t

C(Tt) = sigma R(i), i is every leaf node under node t

Tt: The number of leaves under the node t

1. Calculate α of every node that isn’t a leaf node. Cut down the sub tree whose root node has the minimum α. Cut down means use a leaf node to substitute the original sub tree. If there are multiple sub trees that have the same minimum α, then cut down the sub tree who has the most leaf nodes.
2. Loop pruning the sub tree who has the minimum α, until there is only one node left. Store each newly pruned tree.
3. Plug the second half of the train data set into each tree that has been pruned in the previous steps. Choose the pruned-tree that predicts with the highest accuracy as the

final tree to predict the test data set.

1. **Summary of the final approach we used and why we chose this approach.**
   1. **Summary of the final approach we used in growing the decision tree and why we chose this approach.**

After comparing the three algorithms in table 1, we finally decide to grow the classification tree using CART algorithm. The reasons include:

* 1. There are only two labels assigned to each type of red wine on its quality value, “<=6” and “>6”. Therefore, multi-branches tree algorithm like C4.5 is not necessary. Instead, binary tree would be simpler and more appropriate.
  2. Since there would be a data preprocessing process based on multi-continuous attribute discretization, the time complexity would be relatively large. If we introduce the entropy model, which is adopted by ID3 and C4.5, then there would be too much calculation in each iteration, which would be time consuming. By contrast, CART uses Gini index in each iteration, which reduces the total time since it only contains basic operations.
  3. **The reason that we don’t use the pre-pruning method and choose the post-pruning method.**

① Deficiency of pre-pruning method: It’s hard to decide the threshold value of the maximum layer of the tree or the minimum data sizes of a node. There will be a high probability to underfit the tree.

② Efficiency of the post-pruning method: The tree that is post-pruned usually has more brunches than the tree is pre-pruned since we cut down the sub trees after the data set has been completely and accurately divided into each leaf, and choose the best tree with highest forecast accuracy. In this way, there will be a relatively small probability of overfitting and underfitting since we do the pruning and make the tree has more brunches than the pre-pruned tree. Although compared to pre-pruning, post-pruning is more time-consuming, we can bear the relatively longer time since the data set of this problem is not that large.

③ Why we finally choose the CCP to be the final approach to prune the tree? It simply because in this way, the forecast accuracy of the test data set is the highest.

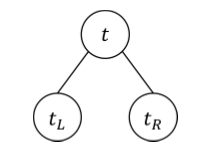
1. **Explain the working principle and logic of the approach used.**
   1. **Explain the working principle and logic of the approach used in growing the classification tree.**

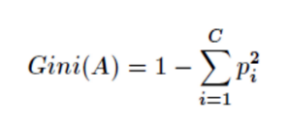
The tree is constructed in a nested dictionary using recursion. The left and right datasets(nodes) after split of the current dataset(node) are stored as values under the keys ‘left’ or ‘right’ of the original dictionary. And all nodes that need further splitting are dictionaries. When the tree comes to its termination, the value of ‘left’ or ‘right’ will not be a dictionary but a string ‘<=6’ or ‘>6’ based on the assigned label.

The basic logic in constructing a classification tree includes the following two components. The working principle of each components are shown under each component.

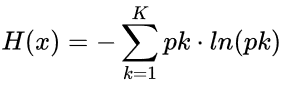
1. Determine “good splitting criteria” and splitting rules.

Given a node t, we define the best split as the one that decreases the impurity the most. The following node t is split into t(L) and t(R).



The Gini-index of t(L) and t(R) can be calculated as , in which pi refers to under the node t(L)/ t(R), the possibility of the event that a type of red wine belongs to class A or B. Then the Gini-index of node t is the weighted sum of Gini-indexes of t(L) and t(R), calculated by: *Gini*(*D*,*A*)=|*D*1|/|*D*|*Gini*(*D*1)+|*D*2|/|*D*|*Gini*(*D*2). After iterating over all features and every split point in each feature, we can determine the best feature and best split point by finding out when the smallest Gini-index occurred.

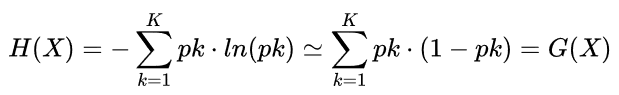
To illustrate the principle, now let’s prove the corresponding feature and the split point of the smallest Gini-index are the best feature and the best split since it leads to the largest information gain ratio. The formula in calculating the information gain is:



The first order Taylor Series of f(x) = ln x at x = 1 is



Thus, we get the expression of calculating the related Gini-index:



Therefore, after iterating through all features and all split points, by choosing the feature and split point where the smallest Gini-index occur, we can determine the best feature and best split point to choose at a node.

1. Determine class assignment rule.

The following are three termination conditions. When one of them occurs, the classification tree stops growing and a class label will be assigned to every type of red wine in the current node.

1. When quality values of every example in the current node are all <= 6 or > 6, the tree comes to the termination. In this case, there’s no need to split the data into two different set anymore. The corresponding class label can be directly assigned to every example in the node.
2. Although not all values of every example in the current node are the same, all features have been iterated and thus no feature can be used in splitting any more, the tree then comes to the termination. In this case, we’ll follow the majority principle. Number of examples that have quality values <= 6 and number of example that have quality values >6 will be counted, then the greater one of these two labels will be assigned to every example in the current node.
   1. **Pruning tree**
3. **REP**

The REP method uses a test data set to correct the overfitting problem of the decision tree.

For every node in the tree that is not a leaf node, try to convert the subtree to a node whose label is the class with the largest number of training samples covered by the subtree. Compare the performance of the original subtree and the simplified node in the test data set. The performance of the subtree is measured by the error rate:

If the error rate after converting the sub-tree is smaller than the error rate before, probably the sub-tree under this node is overfitting. Then prune the sub-tree.

The algorithm traverses all sub-trees in a bottom-up manner until no sub-trees can be replaced to improve the performance of the test data set.

1. **PEP**

Consider the error distribution of the prediction of the tree as a binomial distribution.

We use L to represent the number of leaf nodes under the sub-tree; E to represent number of incorrect predictions under the sub-tree; N to represent the number of examples of data in the sub-tree.

Let the error rate of the tree before pruning be p. When n is large and p is fairly small, binomial distribution can be approximated by the Poisson distribution. By central limit theorem, converges in distribution in N(0,1) as n→∞. To make up for approximation deviations, we use a half-unit correction for continuity to convert the binomial distribution. Error rate **p** should be . Then the expectation of the number of incorrect prediction before pruning is E₁ = Np, and the standard deviation of incorrect prediction is .

Assume to convert the sub-tree to a leaf node. Let the error rate of the sub-tree after pruning be **e**. As the sub-tree has only one node after pruning, . Then the expectation of the number of incorrect prediction after pruning is E₂ = Ne.

To prevent excessive pruning, errors within the standard deviation are allowed. Therefore, if E₁ + std > E₂, that is, the expectation of error before pruning is larger than that after pruning by a standard deviation, the sub-tree may seem to be overfitting.

For every node from top to bottom, we consider its error distribution as binomial distribution, and compare its error rate before pruning and error rate if after pruning to decide whether to prune the sub-tree.

1. **CCP**

The formula of loss function is:

Cα(T) = C(T) + α|T|

C(T) is the forecast error, we call it the cost of the sub tree. T is the number of leaf nodes under each node that isn’t a leaf node, we call it the complexity of the sub tree. Since we want to cut some sub trees to lower the probability of overfitting (that is making C(T) a little bit larger than the sub tree without pruning) but at the same time we are worried about cutting down too much leaves and causing underfit (if T is too small than the tree maybe underfit), thus we make α a parameter to balance the two sides.

For each node that isn’t a leaf node, originally the loss function of this node is:

Cα(Tt) = C(Tt) + α|Tt|

Tt is the original sub tree whose root node is this particular node. If we collapse this sub tree into a single root node, then after pruning the loss function of this node is:

Cα(t) = C(t) + α

t is the sub tree after pruning, since the tree after pruning has only one node, |t| equals to one.

If the loss function before pruning has the same value with the loss function after pruning, that is:

Cα(Tt) = Cα(t)

This means before and after pruning, the sub tree and the root node formed after collapsing the sub tree have the same analog effect. According to Occam’s Razor theorem, we should collapse the sub tree to make the tree simpler. Thus, the tree after pruning will be more generalization and has the same forecast ability compared to the original one.

Expand the previous formula, we can get:

C(Tt) + α|Tt| = C(t) + α

α = (C(t) - C(Tt)) / (|Tt| - 1)

For each node that isn’t a leaf node, we can calculate the α of this node. Then we will get n α of the original tree, named α1, α2 …… αn, sequenced from smallest to largest. Each αi correspond to a pruned tree Ti which turns the node whose α is αi into a root node.

For each α in the range [αi, α i +1), we have the inequality:

C(Tt) + α|Tt| > C(t) + α , t is the pruned tree whose α of the original node is αi.

Thus, we can get: Cα(Tt) > Cα(t). This inequality means for every α in the range [αi, αi+1), pruning the sub tree whose α is αi into a root node is better than preserve the original sub tree. Thus the pruned tree Ti is the optimal tree when α in the range [αi, α i +1).

Then, we can get the conclusion: every Ti is the optimal tree if α in the range [αi, α i +1). Now we have got n optimal trees since i can get the value from {1,2,3,……,n}.

The next step is to decide from which node to start pruning? From the node who has the largest α, or from the node who has the smallest α, or from the node whose C(t) + α is the smallest?

We choose the node who has the smallest α, because when α goes to infinity, the optional tree is the root node, when α goes to zero, the optional tree is the original tree, when α is relatively smaller, the sub tree been pruned is smaller and has less leaves. Thus, the probability of underfit will be less if we choose the node who has the smallest α to prune.

**5. Summary of the result**

In the final version of the classification tree, we choose the best feature and best split point by comparing Gini-indexes, and use CCP to prune the tree to avoid over fitting. The final result shows that, before pruning, the accuracy of prediction is 86.25%. After pruning, the accuracy of prediction by using CCP-pruning is 90.21%. That means pruning using CCP can improve the accuracy by 4.6%. While using REP to prune the tree can only increase the accuracy by 0.63%, from 86.25% to 86.88%. The accuracy will be improved from 87.71% to 88.54% using PEP method, whose final accuracy and improvement of accuracy were still lower than CCP.

**6.Conclusions**

In this machine learning program, we choose to build a classification tree using CART algorithm after comparing with two other algorithms. Then we use three methods: REP, PEP, CCP to prune the tree in order to increase the tree’s generalization ability. After plug in the test data set, we find the best pruning method is cost-complexity pruning.

We not only understand how to write the code to grow and prune the tree, but also understand the logic of these algorithms. For example, we know the differences in implementation methods among CART, C4.5, ID3 and the advantages and disadvantages of these three grow-tree methods. We also understand why we need to prune the tree and the benefits and deficiencies of pre-pruning and post-pruning. For specific ways of post-pruning, we know why these methods make sense, why by doing so we can increase the generalization ability of the tree.

1. **Contribution of each teammate**

盘依岚：Code: PEP-pruning + REP-pruning (Revision of the tree)

Report: everything about PEP and REP

李子琪: Code: CCP-pruning (Revision of the tree)

Report: everything about CCP and pre-pruning + conclusion

孙乐杨：Code: grow-tree （Tree setup）

Report: everything about grow-tree + description of the data + summary of the

result