School of Computer Science
The University of Adelaide

Artificial Intelligence
Assignment 2

Semester 1 2021
Due 11:59pm Tuesday 4 May

---

1 Wine Quality Prediction with Decision Tree

Wine experts evaluate the quality of wine based on sensory data. We could also collect the features of wine from objective tests, thus the objective features could be used to predict the expert's judgement, which is the quality rating of the wine. This could be formed as a supervised learning problem with the objective features as the data features and wine quality rating as the data labels.

In this assignment, we provide objective features obtained from physicochemical statistics for each white wine sample and its corresponding rating provided by wine experts. You are expect to implement **decision tree learning (DTL)**, and use the training set to train your decision tree, then provide wine quality prediction on the test set.

Wine quality rating is measured in the range of 0-9. In our dataset, we only keep the samples for quality ratings 5, 6 and 7. The 11 objective features are listed as follows [1]:

- f_acid: fixed acidity
- v_acid: volatile acidity
- c_acid: citric acid
- res_sugar: residual sugar
- chlorides: chlorides
- fs_dioxide: free sulfur dioxide
- ts_dioxide: total sulfur dioxide
- density: density
- pH: pH
- sulphates: sulphates
- alcohol: alcohol

Explanation of the Data.
`train`: The first 11 columns represent the 11 features and the 12th column is the wine quality. A sample is depicted as follows:

```
 f_acid         v_acid       c_acid       res_sugar  chlorides         fs_dioxide         ts_dioxide  density    pH  sulphates  alcohol  quality
   8.10          0.270         0.41            1.45      0.033               11.0               63.0  0.99080  2.99       0.56     12.0        5
   8.60          0.230         0.40            4.20      0.035               17.0              109.0  0.99470  3.14       0.53      9.7        5
   7.90          0.180         0.37            1.20      0.040               16.0               75.0  0.99200  3.18       0.63     10.8        5
   8.30          0.420         0.62           19.25      0.040               41.0              172.0  1.00020  2.98       0.67      9.7        5
   6.50          0.310         0.14            7.50      0.044               34.0              133.0  0.99550  3.22       0.50      9.5        5
```

`test`: Similar to `train`, but without the 12th colum as they are the values your model will predict. A sample is depicted as follows:

```
f_acid          v_acid         c_acid        res_sugar  chlorides           fs_dioxide           ts_dioxide  density     pH   sulphates  alcohol
   7.0           0.360           0.14            11.60      0.043                 35.0                228.0  0.99770   3.13        0.51   8.900000
   6.3           0.270           0.18             7.70      0.048                 45.0                186.0  0.99620   3.23        0.47   9.000000
   7.2           0.290           0.20             7.70      0.046                 51.0                174.0  0.99582   3.16        0.52   9.500000
   7.1           0.140           0.35             1.40      0.039                 24.0                128.0  0.99212   2.97        0.68  10.400000
   7.6           0.480           0.28            10.40      0.049                 57.0                205.0  0.99748   3.24        0.45   9.300000
```

1.1 Decision Tree Learning

From the given training data, our goal is to learn a function that can predict the wine quality rating of a wine sample, based on the objective features. In this assignment, the predictor function will be constructed as a decision tree. Since the attributes (objective features) are continuous valued, you shall apply the DTL algorithm for continuous data, as outlined in Algorithms 1 and 2. Once the tree is constructed, Algorithm 3 to predict the wine quality to a new wine sample.

Algorithm 1 DTL(*data*, *minleaf*)

```
Require: data in the form of $N$ input-output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, $minleaf \geq 1$
 1: if $(N \leq minleaf)$ or $(y_i = y_j$ for all $i, j)$ or $(\mathbf{x}_i = \mathbf{x}_j$ for all $i, j)$ then
 2:     Create new leaf node $n$.
 3:     if there is a unique mode (most frequent value) in $\{y_i\}_{i=1}^{N}$ then
 4:         $n.label \leftarrow$ mode in $\{y_i\}_{i=1}^{N}$
 5:     else
 6:         $n.label \leftarrow$ unknown
 7:     end if
 8:     return $n$
 9: end if
10: [attr, splitval] $\leftarrow$ ChooseSplit(data) $\Longrightarrow$ Algorithm 2
11: Create new node $n$.
12: $n.attr \leftarrow attr$
13: $n.splitval \leftarrow splitval$
14: $n.left \leftarrow$ DTL($data$ with $\mathbf{x}_i[attr] \leq splitval, minleaf$)
15: $n.right \leftarrow$ DTL($data$ with $\mathbf{x}_i[attr] > splitval, minleaf$)
16: return $n$
```

Algorithm 2 ChooseSplit(*data*)

```
Require: data in the form of N input-output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$.
 1: $bestgain \leftarrow 0$
 2: for each $attr$ in $data$ do
 3:     Sort the array $\mathbf{x}_1[attr], \mathbf{x}_2[attr], ..., \mathbf{x}_N[attr]$.
 4:     for $i = 1, 2, ...N - 1$ do
 5:         $splitval \leftarrow 0.5(\mathbf{x}_i[attr] + \mathbf{x}_{i+1}[attr])$
 6:         $gain \leftarrow$ Information gain of $(attr, splitval)$ // See lecture slides.
 7:         if $gain > begingain$ then
 8:             $bestattr \leftarrow attr$ and $bestsplitval \leftarrow splitval$
 9:         end if
10:     end for
11: end for
12: return $(bestattr, bestsplitval)$
```

Algorithm 3 Predict_DTL($n, data$)

```
Require: Decision tree root node $n$, $data$ in the form of attribute values $\mathbf{x}$.
1: while $n$ is not a leaf node do
2:     if $\mathbf{x}[n.attr] \leq n.splitval$ then
3:         $n \leftarrow n.left$
4:     else
5:         $n \leftarrow n.right$
6:     end if
7: end while
8: return $n.label$
```

1.2 Deliverable

Implement random forest for wine quality prediction in either C/C++, Java or Python.

C++. In the case of C/C++, you must supply a makefile (`Makefile`) with a rule called `winequality` to compile your program into a Linux executable named `winequality.bin`. Your program must be able to be compiled and run as follows:

```
$ make winequality
$ ./winequality.bin [train] [test] [minleaf]
```

JAVA. In the case of JAVA, you must write your program in the file `winequality.java`. Your program must be able to be compiled and run as follows:

```
$ javac winequality.java
$ java winequality [train] [test] [minleaf]
```

Python. In the case of Python, write you program in the file `winequality.py`. Your program must be able to be run as follows:

```
$ python winequality.py [train] [test] [minleaf]
```

At the moment, only an older version of Python (version 2.7.5) is supported on the school servers. If you are not familiar enough with this version of Python, PLEASE DO NOT USE PYTHON for the assignment.

The marking program will decide which of the above to invoke using the following structure:

```
If Makefile exists then
    Compile and run C/C++ submission.
else if winequality.java exists then
    Compile and run Java submission.
else
    Run Python submission.
end if
```

Explanation of the input parameters

- `[train]` specifies the path to a set of training data file.
- `[test]` specifies the path to a set of testing data file.
- `[minleaf]` is an integer greater than zero which specifies the second input parameter to the DTL algorithm (Algorithm 1).

Given the inputs, your program must learn a decision tree (following the prescribed algorithms) using the training data, then predict the quality rating of each of the wine sample in the testing data. Your program must then print to standard output (i.e., the command prompt) the list of predicted wine quality ratings, vertically based on the order in which the testing cases appear in `[test]`.

1.3 Web Submission Instructions

You must submit your program on the Computer Science Web Submission System. This means you must create the assignment under your own SVN repository to store the submission files. The SVN key for this submission is
`2021/s1/ai/Assignment2`
The link to the Web Submission System used for this assignment is:
`https://cs.adelaide.edu.au/services/websubmission/`

After submit your codes through SVN, navigate to 2021, Semester 1, Artificial Intelligence, `Assignment 2 (for COMP SCI 3007 students)`. Then, click Tab "Make Submission" for this assignment and indicate that you agree to the declaration. The automark script will then evaluate your codes.

1.4 Due date and late submission policy

This assignment is due by **11:59pm Tuesday 4 May**. If your submission is late the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.

1.5 Grading

I will compile and run your code on different tests. If it passes all tests you will get **15%** (undergrads) or **12%** (postgrads) of the overall course mark. If you are undergraduate students and you could also finish the task in Section 2 correctly, you will get bonus mark 3%.

There will be no further manual inspection/grading of your program. on the basis of coding style, commenting or "amount" of code written.

Continues next page for postgraduate section.

2 Wine Quality Prediction with Random Forest

For postgraduate students, completing this section will give you the remaining **3%** of the assignment marks.

Random forest is an ensemble method which combines predictions of multiple decision trees. In this task, you will extend your knowledge learnt from decision tree learning to random forest learning (RFL). The process for a simplified RFL given N input-output pairs is:

(1) Randomly select a set of N samples via bootstrap sampling (see explianation later). This dataset is used for training a decision tree (i.e., the root node of the decision tree).

(2) Build a decision tree on the dataset from (1) and apply Algorithm 1.

(3) Repeat (1) and (2) until reaching the maximum number of trees.

This process is also shown in Algorithm 4. In random forest learning, a sample set is used to train a decision tree. That is to say, different trees in the forest could have different root data. For prediction, the random forest will choose the most voted label as its prediction.

For the wine quality prediction task, you shall apply Algorithm 4 for random forest learning and apply Algorithm 5 to predict the wine quality for a new wine sample.

Bootstrap sampling. It is a random sampling with replacement (i.e., replace the sampled data back to the original dataset). For example, if $N = 5$, so we have $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_5\}$ (omit y_i for simplicity). To perform bootstrap sampling, we randomly select one data point from the five, let's say $\mathbf{x}_3$, to be the first sample. Then we put $\mathbf{x}_3$ back to the dataset and draw another sample from the five data points. This time, the sample could also be $\mathbf{x}_3$ or another one. We repeat this process until we get N samples and form a sample set. In this way, the sample set could contain repeated data points. So the following cases are possible:

$\{\mathbf{x}_3, \mathbf{x}_3, \mathbf{x}_3, \mathbf{x}_3, \mathbf{x}_3\}$, when all the data samples are the same.
$\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_2, \mathbf{x}_5\}$, when all the data samples are different.
$\{\mathbf{x}_3, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1\}$, when some samples are repeated.
...

Algorithm 4 RFL(*data, minleaf, n_trees, rand_seed)*

```
Require: data in the form of N input-output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$, $n\_trees > 1, minleaf \geq 1$.
1: $forest \leftarrow []$
2: sample_indexes ← $N * n\_trees$ random integers with value in $[0, N)$ generated by rand_seed
3: count ← 0
4: for count < n_trees do
5:     sampled_data ← N data pairs selected by N indexes from sample_indexes sequentially
6:     n=DTL(sampled_data, minleaf) $\Longrightarrow$ Algorithm 1
7:     forest.append(n)
8: end for
9: return forest
```

Algorithm 5 Predict_RFL($forest, data$)

```
Require: forest is a list of tree roots, data in the form of attribute values $\mathbf{x}$.
1: labels ← []
2: for Each tree n in the forest do
3:     label ← Predict_DTL (n, data) $\Longrightarrow$ Algorithm 3
4:     labels.append(n)
5: end for
6: return the most voted label in labels
```

2.1 Deliverables

Implement random forest for wine quality prediction in either C/C++, Java or Python.

C++. In the case of C/C++, you must supply a makefile (`Makefile`) with a rule called `winequalityRFL` to compile your program into a Linux executable named `winequalityRFL.bin`. Your program must be able to be compiled and run as follows:

```
$ make winequalityRFL
$ ./winequalityRFL.bin [train] [test] [minleaf] [n_trees] [random_seed]
```

JAVA. In the case of JAVA, you must write your program in the file `winequalityRFL.java`. Your program must be able to be compiled and run as follows:

```
$ javac winequalityRFL.java
$ java winequalityRFL [train] [test] [minleaf] [n_trees] [random_seed]
```

Python. In the case of Python, write you program in the file `winequalityRFL.py`. Your program must be able to be run as follows:

```
$ python winequalityRFL.py [train] [test] [minleaf] [n_trees] [random_seed]
```

At the moment, only an older version of Python (version 2.7.5) is supported on the school servers. If you are not familiar enough with this version of Python, PLEASE DO NOT USE PYTHON for the assignment.

Explanation of the input parameters

- `[train]` specifies the path to a set of training data file.
- `[test]` specifies the path to a set of testing data file.
- `[minleaf]` is an integer greater than zero which specifies the second input parameter to the RFL algorithm (Algorithm 4).
- `[n_trees]` is an integer greater than one which specifies the third input parameter to the RFL algorithm (Algorithm 4).
- `[random_seed]` is the seed value generate random values. Please use:
  - C++: srand(*rand_seed*) to set seed value and rand() % (b - a + 1) + a to create a random integer in [a, b], repeat rand to get enough random values.
  - JAVA: import java.util.Random, use random.setSeed(*rand_seed*) to set seed value and Random.nextInt(M)to create a random value in [0, M), repeat Random.nextInt to get enough random values.
  - Python: import random package, use random.seed(*rand_seed*) to set seed value and random.randint(0, M) to create a random value in [0, M], repeat random.randint to get enough random values.

Given the inputs, your program must learn a random forest (following the prescribed algorithms) using the training data, then predict the quality rating of each wine sample in the testing data. Your program must then print to standard output (i.e., the command prompt) the list of predicted wine quality ratings, vertically based on the order in which the testing cases appear in `[test]`.

2.2 Web Submission Instructions

You must submit your program on the Computer Science Web Submission System. This means you must create the assignment under your own SVN repository to store the submission files. The SVN key for this submission is
`2021/s1/ai/Assignment2`
The link to the Web Submission System used for this assignment is:
`https://cs.adelaide.edu.au/services/websubmission/`

After submit your code through SVN, navigate to 2021, Semester 1, Artificial Intelligence, `Assignment 2 (for COMP SCI 7059 students)`. Then, click Tab "Make Submission" for this assignment and indicate that you agree to the declaration. The automark script will then check whether your code compiles. You can make as many resubmissions as you like. If your final solution does not compile you won't get any marks for this solution.

2.3 Due date and late submission policy

This assignment is due by **11:59pm Tuesday 4 May**. If your submission is late the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.

2.4 Grading

I will compile and run your code on different tests. If it passes all tests you will get **3%** of the overall course mark. Undergraduate students who could finish the task correctly will get bonus mark **3%**.

There will be no further manual inspection/grading of your program. on the basis of coding style, commenting or "amount" of code written.

References

[1] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems 47*, 4 (2009), 547–553.