15.095 Homework 3

Kim-Anh-Nhi Nguyen. MIT ID: 9137855521

2018-10-29

1 Optimal Classification Trees (OCT)

We splitted the data into 50%/25%/25% training, validation, and testing sets.

After tuning the parameters, we found that the best parameters for the OCT are:

- $cp = 0.23$ (tuned automatically)
- Max Depth = 2
- Minbucket = 25

The accuracy values are:

- Training accuracy: 97.33%
- Validation accuracy: 94.87%
- Testing accuracy: 91.67%

The model found is the following:

1. (depth 0) Split: PetalLength < 2.45
2. (depth 1, Left) Predict setosa
3. (depth 1, Right) Split: PetalLength < 4.85
4. (depth 2, Left) Predict: versicolor
5. (depth 2, Right) Predict: virginica

2 Variable Importance

(a)

A way to determine variable importance in an OCT model can be to look at node impurity.

The algorithm would be the following:
To calculate the variable importance score, we can look at the improvement in impurity measure corresponding to each variable when they appear in a node. Each time we encounter a variable in a node, we add the improvement to the previous improvement from that same variable. The values of ALL these

improvements are summed over each node and totaled, and are then scaled relative to the best performing variable: the variable with the highest sum of improvements is scored 100, and all other variables will have lower scores ranging downwards toward zero.

A variable can obtain an importance score of zero in the tree only if it never appears in a split. Indeed, because such a variable plays no role anywhere in the tree, eliminating it from the data set should make no difference to the results.

(b)

Using the OCT model trained in question 1, we got the following variables' importance:

Feature	Importance (in %)
PetalLength	0.7361
PetalWidth	0.26389
SepalLength	0.0
SepalWidth	0.0

3 OCT and ANN

It is possible to construct an artificial neural network (ANN) that gives the exact same predictions as the OCT model found in question 1.

The OCT found, which is drawn in the appendix (please, see the last page), has $N_1 = 2$ split nodes and $N_2 = 3$ leaf nodes.

According to theorem 3 of lecture 12 about Neural Networks and Trees, **a neural network with perceptron activation functions, two hidden layers, N_1 nodes in the first hidden layer, and N_2 nodes in the second can classify training data at least as well as a given classification decision tree with N_1 split nodes and N_2 leaf nodes.**

So, our ANN will have 2 layers: one hidden layer and one output layer. The hidden layer will have 2 nodes, and the the output layer will have 3 nodes.

The ANN has been drawn in the appendix (please, see the last page). We can clearly check that it classifies exactly the same way as the OCT does.

4 Missing Data Imputation

The MAE and parameters that we found are the following:

Imputation method	MAE	Best parameters
Mean	0.83084	
`opt.knn`	0.44514	k = 30, Algorithm = CD, Norm = l_1

The MAE for the `opt.knn` imputation is lower than for the mean imputation, so the `opt.knn` imputation is the most accurate.

After re-training the OCT model from 1, we get the following parameters and accuracy scores for each of the method:

Imputation method data	`cp`	`max depth`	`min bucket`	Training accuracy	Validation accuracy	Testing accuracy
Mean	0.5	3	5	33.33%	33.33%	33.33%
`opt.knn`	0.005	5	5	89.33%	94.87%	83.33%

Remark: the OCT trained for the imputed data from the mean imputation method is unchanged.

The `opt.knn` method gives much better results than the mean method in terms of accuracy.
Let us look at the correlation matrices of the real data and the imputed data for both methods:

- The correlation matrix for the imputed data with the mean method is:

1.0	−0.149	0.498	0.432
−0.149	1.0	−0.227	−0.1
0.498	−0.227	1.0	0.453
0.432	−0.1	0.453	1.0

- The correlation matrix for the imputed data with the `opt.knn` method is:

1.0	−0.317	0.912	0.902
−0.317	1.0	−0.495	−0.434
0.912	−0.495	1.0	0.964
0.902	−0.434	0.964	1.0

- The correlation matrix for the original data is:

1.0	−0.118	0.872	0.818
−0.118	1.0	−0.428	−0.366
0.872	−0.428	1.0	0.963
0.818	−0.366	0.963	1.0

We can observe, by comparing the 3 matrices, element by element, that the correlations between variables in the imputed data with `opt.knn` method are much more similar to the correlations in the original data, than in the correlations with the imputed data from the mean method.

Moreover, we can notice that the correlation matrix of the original data has high correlations among some variables (e.g. between x_3 and x_4, or, x_1 and x_3). Therefore, by using the `opt.knn` method, we are able to cluster the data points that have similar value in variables, which indicates that the data should be similar where the high correlations are.

So, we can conclude that the `opt.knn` imputation method better takes the correlations between variables into account, so it gives better results than the mean imputation method.

5 Imputation including the Y-variable

We now include the outcome column variable as a column in the data frame to be imputed. However, we assume that we only now the Y-variable for the **training set and the validation set**. So in the following paragraphs, we are doing:

- Imputation of the missing training and validation data including the outcome variable
- Imputation of the missing testing data, without the outcome variable

The new MAE and parameters that we found are the following (here imputations were done on the whole dataset including the Y-variable because there was no way to separate the datasets with missing values):

Method	MAE	Best parameters
Mean (unchanged)	0.83084	
`opt.knn`	0.24311	k = 20, Algorithm = BCD, Norm = l_1

Then, we kept the imputed values for the training and validation sets only.

Then, we performed another data imputation for the testing set, leaving out the outcome variable. Here are the results:

Method	MAE	Best parameters
Mean	0.84569	
`opt.knn`	0.34430	k = 10, Algorithm = BCD, Norm = l_1

The MAE for both methods are worse (greater) than before, because we obviously have less data in the testing set than in the whole dataset.

After re-training the OCT model from 1, we get the following parameters and accuracy scores for each of the method:

Imputation method	`cp`	`max depth`	`min bucket`	Training accuracy	Validation accuracy	Testing accuracy
Mean	0.01	6	5	82.67%	94.87%	88.89%
`opt.knn`	0.5	3	5	100.0%	100.0%	86.11%

In both methods, the training and validation accuracies have improved compared to the previous question.

Specifically, the mean method performs very well on the validation. However the testing accuracy is lower than the testing accuracy in the real dataset.
The `opt.knn` method gives much better results than in the previous question. However, there is **over-fitting**: the predictions are perfect for training and validation data, then, they are poorer in the testing set, than both mean method imputed data AND the real dataset.

Let us look at the correlation matrices of the real data and the imputed data for both methods:

- The correlation matrix for the imputed data with the mean method is:

1.0	−0.089	0.541	0.451
−0.089	1.0	−0.195	−0.050
0.541	−0.195	1.0	0.369
0.451	−0.050	0.369	1.0

- The correlation matrix for the imputed data with the `opt.knn` method is:

1.0	−0.240	0.908	0.899
−0.240	1.0	−0.461	−0.413
0.908	−0.461	1.0	0.977
0.899	−0.413	0.977	1.0

- The correlation matrix for the original data is unchanged.

We can observe, by comparing the 3 matrices, element by element, that the correlations between variables in the imputed data with `opt.knn` method are even more similar to the correlations in the original data, than in the previous question.
On the contrary, there is no improvement in similarity between the imputed data with mean method and

the real data.

Conclusions:
Including the Y-variable is benefinital in order to get as close as possible to the real data: including the Y-variable in our imputations with the `opt.knn` method takes into account the similarities between the X-variables and the Y-variable, in addition to the similarities among the X-variables. With this, we saw that it gave better predictions than when using the original data. Omitting the Y variable when imputing data prevents us from having more information on missing values: outcomes like *'Has the patient survived the disease X'* carries information about the missing values such as *'Blood pressure'*, so this information should be used in those cases, where it is important to be as close from reality as possible.

However, the downside of including the Y-variable is **overfitting**.