MLT - Assignment 1

Ayush Sekhari

January 25, 2015

1 Observations

- The data set has missing values
- Attribute values are numerical - either integers or real numbers
- Training data is labelled - Supervised Learning
- class value 1 is interpreted as ”tested positive for diabetes”
- In my solution, I would be considering Branching factor or Branching Ratio to be 2.
- The attributes have been renamed as follows :
  - Number of times pregnant → Num_Preg
  - Plasma glucose concentration a 2 hours in an oral glucose tolerance test → PGC
  - Diastolic blood pressure (mm Hg) → BP
  - Triceps skin fold thickness (mm) → Tricept_Thickness
  - 2-Hour serum insulin (mu U/ml) → Insulin
  - Diabetes pedigree function → DPF
  - Age (years) → Age
  - Body mass index (weight in kg/(height in m)^2) → BMI
  - Class variable → Label
- The random seed has been set to 2 for deterministic analysis

2 Libraries Used

The Programming is done in `R` Programming Language. The following Libraries are used:

1. `rpart` : To build, test and plot classification trees
2. `party` : To divide our data-set for 5 fold- cross validation

3 Handling Missing Attributes

The missing data in only in the columns 2-8(Number of times pregnant can be zero).
Since the data is numeric, we have to use some substitute for the missing values or handle it in some way(Else, in the case of nominal data, we could have made the missing value as a separate label and classified accordingly). We can handle it in two ways :

3.1 For Training Data

parameter : Missing_Attribute_Train

- **Missing_Attribute_Train = "UseNA" :** The 0(missing) value is replace by NA in the training data, We do not use this instance while calculating the impurity with respect to this attribute for finding the splitting criteria.
- **Missing_Attribute_Train = "UseMean" :** The 0(missing) value is replaced by the class average for that corresponding attribute in the training data. Class average seems to be a better substitute as compared to total average. We can replace by other metrics as well like median or mode but I would be using mean.

3.2 For Test Data

parameter : Missing_Attribute_Test

- **Missing_Attribute_Test = "UseNA" :** The 0(missing) value is replace by NA in the training data, We use surrogate splits to estimate in the cases where the missing attribute is required for splitting. These surrogate splits were calculated while training.
- **Missing_Attribute_Test = "UseMean" :** The 0(missing) value is replaced by the class average for that corresponding attribute in the training data. Training data is used instead of test data because we don't know much about the test data, test data in real life is very small and given tuple wise. Class average seems to be a better substitute as compared to total average. We can replace by other metrics as well like median or mode but I would be using mean.

4 5 Fold Cross-Validation

We have randomly divided our data set into 5 disjoint (approximately same size) data sets. One part is used as the test set and other is used as the training set.

5 Paramters Used

The following Parameters signify the conditions in which the classification tree was constructed on training data and performance was measured on test data.

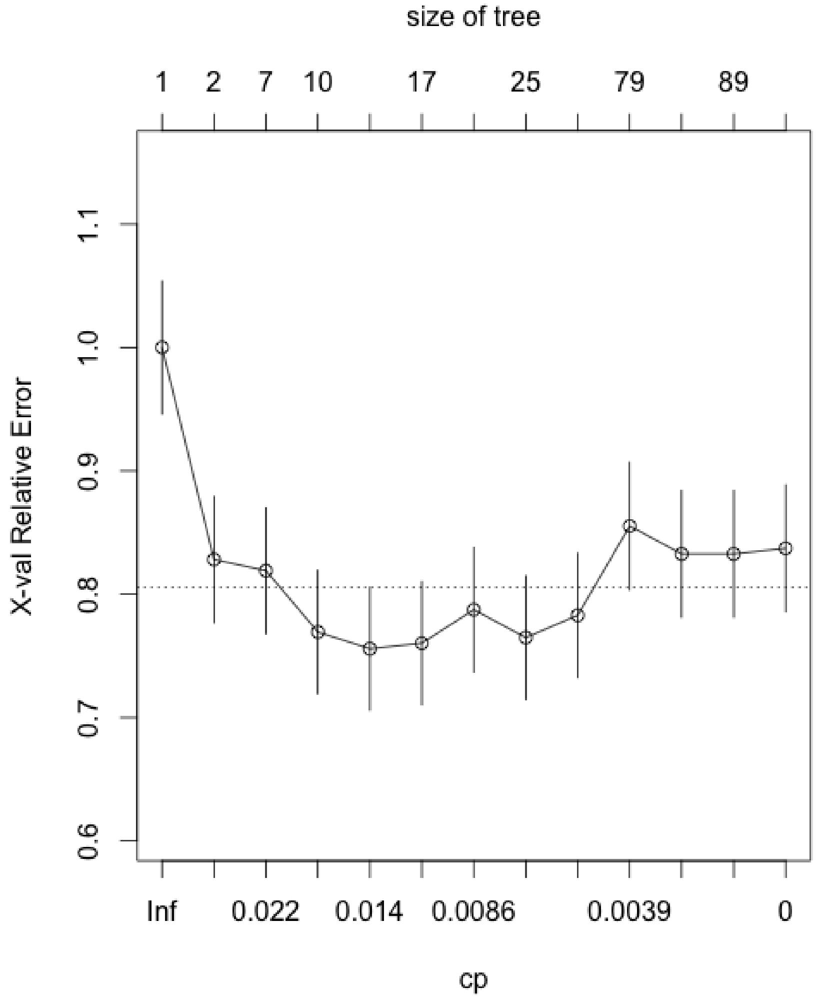
- **Impurity_Function (I)** : may be "gini" for gini index or "information" for entropy as the impurity function
- **Missing_Attributes_Train (MATr)** : may be "useNA" or "useMean"
- **Missing_Attributes_Test(MATe)** : may be "useNA" or "useMean"

6 Grow the tree as much as possible and then Prune

Script Used : `Pruning.R`

Here for each cross validation set iteration, we first build a completely overfitting tree for the training data by setting the parameters cp = 0 and minsplit = 1 .
The we pruned the tree using the value of cp for which `X-Val Relative` error is the minimum. This is done using the cptable data of the decision tree. Help was taken from the online source `http://www.statmethods.net/advstats/cart.html`
The Plot of X-val relative vs CP for the training set 5 is as shown in figure 1. The minima in this graph is used for the value of cp at which pruning is done.

Figure 1: X-val Relative Error vs CP for training set 5



Various Performance metrics under different cases are as shown in Tables below.

Table 1: Various Performance metrics under different parameters while pruning

Parameters	**accuracy**	**precision**	**recall**	**TP Rate**	**FP Rate**	**F1 Score**
I = gini MATr = UseNA MATe = UseNA	**0.753**	**0.686**	**0.562**	**0.562**	**0.14**	**0.61**
I = information MATr = UseNA MATe = UseNA	**0.741**	**0.658**	**0.577**	**0.577**	**0.169**	**0.606**
I = gini MATr = UseMean MATE = UseNA	**0.725**	**0.623**	**0.548**	**0.548**	**0.179**	**0.581**
I = information MATr = UseMean MATE = UseNA	**0.732**	**0.628**	**0.557**	**0.557**	**0.175**	**0.589**
I = gini , MATr = UseNA , MATe = UseMean	**0.74**	**0.677**	**0.512**	**0.512**	**0.134**	**0.575**
I = information , MATr = UseNA , MATe = UseMean	**0.737**	**0.655**	**0.56**	**0.56**	**0.167**	**0.596**
I = gini , MATr = UseMean , MATe = UseMean	**0.697**	**0.672**	**0.26**	**0.26**	**0.07**	**0.374**
I = information , MATr = UseMean , MATe = UseMean	**0.704**	**0.644**	**0.33**	**0.33**	**0.096**	**0.426**

7 Threshold on the number of data vectors at a node while splitting

Script Used : `MinSplit.R`
We alter the `minsplit` parameter in `rpart.control` to set the minimum number of observations that must exist in a node in order for a split to be attempted while building the classification tree. Here, we divide the data into 4:1 parts, training is done on the larger part and we chose the value of the parameter for which accuracy on the test data is the best. We have used the average obtained in this way. (It was seen after dividing the data into three parts i.e. CV + Test + Train, that our model performs good). The parameter `cp` is set to be 0 for independent effects of minsplit. Final accuracies are though reported using 5 Fold Cross-Validation. Various performance metrics under different parameter settings can be seen in the tables below.

8 Threshold on the decrease in impurity while splitting

Script Used : `Decrease.R`
We alter the `cp` parameter in `rpart.control`. Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted. Here, we divide the data into 4:1 parts, training is done on the larger part and we chose the value of the parameter for which accuracy on the test data is the best. We have used the average obtained in this way. (It was seen after dividing the data into three parts i.e. CV + Test + Train, that our model performs good). The parameter `minsplit` is set to be 1 for independent effects of cp. Final accuracies are though reported using 5 Fold Cross-Validation. Various performance metrics under different parameter settings can be seen in the tables below.

Table 2: Various Performance metrics under different parameters while using threshold on the number of data vectors at a node while splitting

Parameters	**accuracy**	**precision**	**recall**	**TP Rate**	**FP Rate**	**F1 Score**
I = gini , MATr = UseNA , MATe = UseNA	**0.78**	**0.705**	**0.628**	**0.628**	**0.14**	**0.661**
I = information , MATr = UseNA , MATe = UseNA	**0.783**	**0.69**	**0.687**	**0.687**	**0.167**	**0.688**
I = gini , MATr = UseMean , MATe = UseNA	**0.736**	**0.64**	**0.606**	**0.606**	**0.192**	**0.613**
I = information , MATr = UseMean , MATe = UseNA	**0.738**	**0.641**	**0.598**	**0.598**	**0.185**	**0.612**
I = gini , MATr = UseNA , MATe = UseMean	**0.773**	**0.705**	**0.601**	**0.601**	**0.136**	**0.644**
I = information , MATr = UseNA , MATe = UseMean	**0.78**	**0.689**	**0.679**	**0.679**	**0.167**	**0.684**
I = gini , MATr = UseMean , MATe = UseMean	**0.715**	**0.685**	**0.364**	**0.364**	**0.102**	**0.464**
I = information , MATr = UseMean , MATe = UseMean	**0.713**	**0.678**	**0.355**	**0.355**	**0.094**	**0.453**

Table 3: Various Performance metrics under different parameters while using threshold on the decrease in impurity while splitting

Parameters	**accuracy**	**precision**	**recall**	**TP Rate**	**FP Rate**	**F1 Score**
I = gini , MATr = UseNA , MATe = UseNA	**0.771**	**0.689**	**0.628**	**0.628**	**0.152**	**0.655**
I = information , MATr = UseNA , MATe = UseNA	**0.767**	**0.72**	**0.569**	**0.569**	**0.124**	**0.62**
I = gini , MATr = UseMean , MATe = UseNA	**0.742**	**0.634**	**0.647**	**0.647**	**0.205**	**0.635**
I = information , MATr = UseMean , MATe = UseNA	**0.743**	**0.644**	**0.595**	**0.595**	**0.177**	**0.616**
I = gini , MATr = UseNA , MATe = UseMean	**0.758**	**0.682**	**0.576**	**0.576**	**0.146**	**0.619**
I = information , MATr = UseNA , MATe = UseMean	**0.766**	**0.732**	**0.531**	**0.531**	**0.106**	**0.604**
I = gini , MATr = UseMean , MATe = UseMean	**0.714**	**0.679**	**0.363**	**0.363**	**0.104**	**0.463**
I = information , MATr = UseMean , MATe = UseMean	**0.711**	**0.651**	**0.377**	**0.377**	**0.109**	**0.468**

9 Conclusions

In General the average accuracy observed is between 72-77%. This is quite good with such simple models.