Occupation Classifier

Sebastián Jiménez-Bonnet - 05427277

December 15, 2011

1 Introduction

In this project I explored how accurately could the occupation of a person be guesses based on other demographic variables. To this end, several algorithms were attempted, such as multinomial logistic regression, SVM and Classification and Regression Trees (CART). I present here the results from the most successful algorithm, CART, and discuss some of the reasons that made it perform better.

2 The Data

The data set used is an extract form a bigger marketing database[1], originally intended to classify people by income. The original data set consists of 9409 questionnaires filled up by shopping mall customers in the San Francisco Bay area. The questionnaires contained 502 questions, and the extract used consists on 14 demographic attributes. This dataset is a mixed set in which we can find many categorical and numerical variables with a lot of missing values. After those entries for which the response variable was missing were removed, a total of 8857 observations served as our dataset to train and test the predictor. All the attributes were discretized and are listed below with their corresponding number of classes [2]:

1. Occupation (9 levels)
2. Annual income of household (9 levels)
3. Sex (2 levels)
4. Marital status (5 levels)
5. Age (7 levels)
6. Education (6 levels)
7. How long have you lived in the San Francisco/ Oakland/San Jose area? (5 levels)
8. Dual incomes, if married (3 levels)
9. Persons in your household (9 levels)
10. Persons in household under 18 (9 levels)
11. Householder status (3 levels)
12. Type of home (5 levels)
13. Ethnic classification (8 leveles)
14. What language is spoken most often in your home? (3 levels)

Many of the variables have many levels, including the response variable *Occupation* which has 9 levels. This, summed to the fact that almost 20% (1743 observations) of the data set have at least one missing value, makes this data set hard to work with. The treatment of these missing values played a central role the project.

The dataset was randomly split into a training set with 7085 entries (80%) and a learning set with 1772 entries (20%). The former was used to fit our model and the latter was kept separate until the very end where it was used to estimate the prediction error of our final model.

[1] Source: Impact Resources, Inc., Columbus, OH (1987).

[2] For a complete listing of the levels visit http://www-stat.stanford.edu/ tibs/ElemStatLearn/

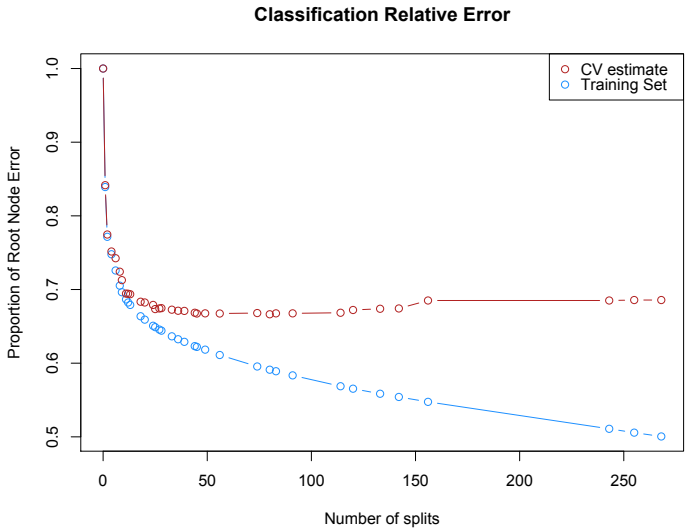

Figure 1: A plot of the CV error and the training set error for each of the fitted trees, as a fraction of the error achieved by the single node tree. On the x-axis we have the number of nodes of the corresponding trees

3 Classification and Regression Trees

The CART algorithm, roughly speaking, generates a sequence of binary splits on one of the input variables at a time, so that the resulting *measure of impurity* is minimized each time. The tree is grown until a stopping criteria is satisfied and finally it is pruned to avoid overfitting. In order to make a prediction, it assigns the class with the most votes to each of the final nodes.

There are several reasons why trees are specially appropriate for our particular problem. First, trees work well with discrete input and response variables that have many levels. Second, they are fairly easy to interpreted which may lead to a deeper understanding of the relation of the input with the response variable. Lastly, trees offer several ways to deal with missing observations, which is really important for our dataset.

3.1 Missing values

Binary trees are specially well suited to deal with missing values. There are several different approaches to this matter, the simplest one is to forget about them and build a model based only on observations that have all their attributes observed. This approach wastes lots of information, (we would dismiss almost 20% of our dataset!) and is not able to offer a prediction for entries that have missing values themselves.

Another approach is to add a dummy level to each of our attributes, corresponding to a missing value. This approach is useful in that it can help uncover a relation between the missing values and the response. But if the missing values

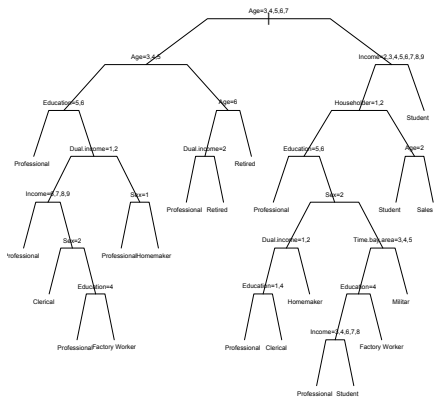

Figure 2: Our final binary tree.

appear randomly or are the product of data processing rather that data collection, this approach is also inefficient.

Both of these approaches can be implemented by many algorithms, but trees offer an additional one, *surrogate splits*. It consists in choosing several substitute splits for each of the original splits elected for the model. In choosing these surrogate splits, only variables different from the original one are considered. The split is then chosen so that the resulting partition of the data has the biggest overlap with the partition the original split achieved (where only data that has both variables observed is considered). The second surrogate excludes the original variables and the variable for the first surrogate, and so on.

3.2 Fitting the predictor

In this implementation of the CART algorithm, the GINI index is used as the measure of impurity to grow the tree. Three criteria were considered to stop branching.

1. The nodes considered for splitting were required to have at least 5 observations.
2. The end nodes were required to have at least two observations.
3. The split to be made had to decrease our measure of impurity by at least $C_p = 0.001$.

These stopping rules aim to avoid huge trees that are computationally expensive and which would be trivially pruned at a later stage.

To fit the model I used **R** as the software platform, specifically I used the package *rpart*. The algorithm calculates a whole path of nested trees, from the trivial root node, to the full tree reached with our stopping criteria. It indexes each tree by the value of the parameter C_p that would have taken to stop branching exactly at that tree.

To choose the appropriate tree and avoid overfitting I calculated a 10-fold cross-validation (CV) error for each of the trees. In Figure 1 we

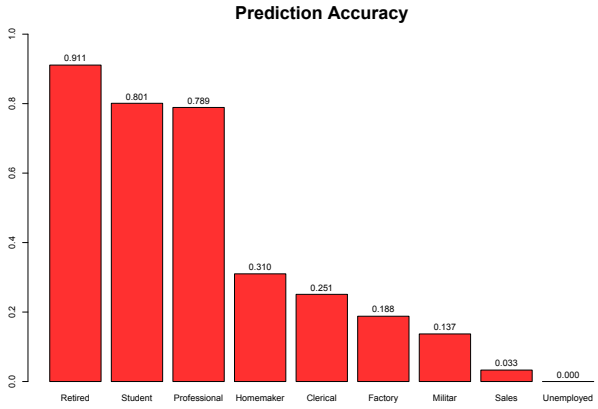

Figure 3: Accuracy for class of the response variable *Occupation*.

can see a plot of the training error and the CV error for each of these trees as a percentage of the trivial root-node classification. The root node trivial error is 0.679, and is achieved by the classifier which assigns the most frequent class (in our case Professional/Managerial) to any input. I used the two standard deviation parsimonious approach in which the selected model is the simplest model (least number of final nodes) which achieves a CV error within to standard deviations of the minimum CV error.

3.3 Final model

Our final model corresponds to a C_p value of 0.00208, which translates into a binary tree with 20 nodes. In figure 2 we can see the structure of the model. Note that the model only uses 7 of the 13 possible variables. The variables that seem to be the most important in our tree are *Age*, *Annual income* and *Education*.

3.4 Prediction

Testing our final model on the test set, it has a resulting test error of 46%. At first sight it may seem a bit high, but considering we have 9 classes for the response variable and a sparse training set, it is actually a quite remarkable performance. For a frame of reference, it can be compared to the root node error, which is 68%. In figure 3 we can see the classification accuracy for each class. We can see that, excluding the *Retired* (which is easily identifiable by age), there is a clear bias in favor of the more frequent classes which are *Professional* and *Student*. In fact the least frequent class, *Unemployed* is never assigned.

4 Conclusions

Our tree predictor had a good performance in classifying our test set. One key feature of our model was its treatment of the missing values. For comparison, I fitted another CART classifier omitting the missing values and got a test error

of 53%, which suggests that there is a considerable loss of information doing this.

There are some questions that still need to be answered in order to asses our results. How general is our model? Can we extend it to the general population of California or the United States? One obstacle I can see, which figure 3 supports, is that our predictions are heavily affected by the frequency of each class. Probably, in order to have more generally applicable results, we could weight our observations in order to balance the frequency of each class with the frequency of the actual population we want to predict.

Regarding the importance of the variables, it is important to keep in mind that binary trees have a really high variance. Little variation on the data can mean many different combinations on the selected variables for each split. In order to better asses the importance of the variables, at least when there is a lack of context to aid us, Random Forests has a more robust *importance* measure for the predictor variables, and may be a good continuation of this work.

5 References

(i) Hastie T., R. Tibshirani and J. Friedman J *The Elements of Statistical Learning* Stanford, California. 2008.

(ii) Bishop, Christofer M. *Pattern Recognition and Machine Learning* Cambridge. 2006.

(iii) Michel, Tom M. *Machine Learning* 1997.