Final

Ezana Sahle

1. Introduction

The data used in this paper is from the 2nd session of the 84th Congress (1984) which contains 435 observations. These observations are broken down into training observations, $n = 290$, and testing observations, $m = 145$. There are 33 variables in total, including the output variable and the dummy variables. For each of the 16 input variables, two dummy variables were created, yielding $\rho = 32$. The output variable in this dataset is the "party" variable which indicates the respective party of each congress person. The goal of this paper is to solve the classification problem found in this dataset by training the input variables to accurately classify congress members as either a democrat or a republican.

In order to solve this problem, the train-test split is applied to each model to evaluate how well each one can classify the new test observations. The methods used to address this problem include a linear probability model, a logistic regression model, a classification tree, bagging, and random forests. The respective error rates of each model will be used to evaluate performance.

2. Data

The output variable "party" contains either democrat or republican. To prepare the data to be analyzed, the values of "party" were converted into binary values of 1 for a democrat, and 0 for a republican. Similarly, each of the input variables were converted into binary values as well. However, each input variable can take one of three values: yay; nay; or question mark. Therefore, two dummy variables were created for each original input variable. To account for the three options most effectively, a dummy variable for yay and nay were created. A value of 1 means the

respective dummy (yay or nay) occurred and a value of 0 represents either of the other two options. The table below describes each variable within the congressional voting record (1984) dataset.

Table 1: Variable Definitions

Variable Name	Type	Description
party	Output	Congress Representative's Party affiliation: democrat or republican
aidcontra	Input	Vote on providing aid to the Nicaraguan Contras
antisatellite	Input	Vote on anti-satellite test ban
budget	Input	Vote on the adoption of the budget resolution
crime	Input	Vote on crime
dutyfree	Input	Vote on duty free exports
education	Input	Vote on education spending
elsalvador	Input	Vote on providing aid to El Salvador
exportafrica	Input	Vote on the South African *Export Administration Act*
handicapped	Input	Vote on handicapped infants
immigration	Input	Vote on immigration
missile	Input	Vote on the mx missile
physician	Input	Vote on freezing physician fees
religious	Input	Vote on religious groups in schools
sue	Input	Vote on superfund right to sue
synfuels	Input	Vote on corporate cutback for synfuels
water	Input	Vote on water project cost sharing

3. Methods

The five methods used in this paper are the linear probability model, the logistic regression model, a basic classification tree, bagging, and random forests. For the linear probability model, similar to an OLS model, the output variable was regressed over the input variables to identify an optimal linear function. This optimal linear function is calculated using the 290 observation in the training data. Then, the values in this linear model are classified into binary values based on whether their output is greater than or less than 0.5. If greater than 0.5, then the observation is classified as a 1 (democrat), and if less than 0.5, it is classified as a 0 (republican). Then these "fitted values" are used to compared against those of the test set to determine the performance of the model by

counting the number of classification errors. The difference between the linear probability model and a regular linear regression model is that the values in this probability model are categorical, and not continuous. The problem that arises when using a linear probability model, is that the output values of the optimal regression line go beyond one and zero. Since a number outside one and zero is not a probability, the linear regression does not accurately reflect the nature of the data (Hastie et al. 2009, p.130). For example, a voter cannot be 110% democrat, or -10% republican. Nonetheless, the results of the model will not have much impact on the model's ability to classify new data, as demonstrated later in the paper.

For the logistic model, the same process is followed. The logistic regression is carried out on the training data. Then the model is evaluated against the outputs in the test data, and the performance of the model is evaluated. However, this time the logistic regression model will determine the probability that the output will be democrat and all the probabilities will be between zero and one, unlike the linear probability model that contained values that were above one. The logistic regression model is able to ensure that its probabilities are between zero and one through a monotone transformation of the linear probability model called the logit transformation. Each estimator is then determined through the maximum likelihood (ML) of the observed input variables (Hastie et al. 2009, p.132). These ML estimates are then used to classify the test data.

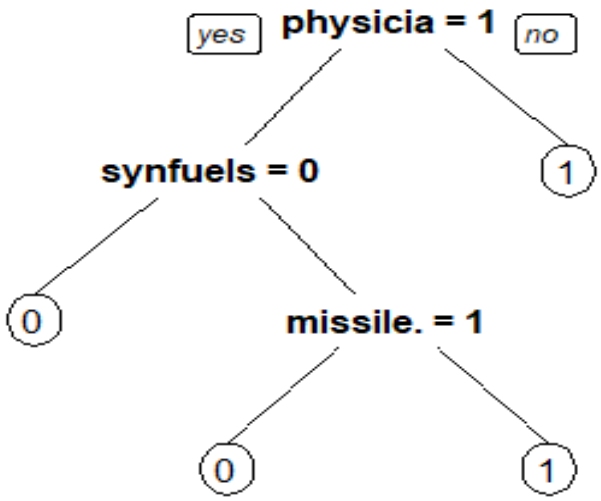
For the classification tree, this method utilizes binary recursive partitioning (also called the top-down, greedy approach) to grow the tree seen in Figure 1. The split in the dataset is determined by the point which maximizes the decrease in impurity and continues to maximize until the terminal nodes cannot be split any further. In simpler terms, a tree attempts to find linear (vertical, or horizontal) partitions in the data's input variables. These partitions can then be used to classify an observation as one of the output variables' categories. In this case, a binary classification

problem, either democrat (1) or republican (0). After a series of partitions, the tree should be able to accurately classify the data with a high degree of effectiveness and provide a relatively intuitive model like the one found in Figure 1. The top-down greedy approach to growing trees refers to starting at the top of a tree with all the training observations and inputs, and then identifying the best partition to make. The greedy aspect of this approach refers to the "best" split, which is usually the best split that accurately classifies observations right away. However, the best split does not always lead to the most effective tree in the long run, as sacrificing the most optimal split in the first step could lead to a better split later on in the tree (Hastie et al. 2009, p.306). The tree building process also requires pruning in order to ensure that the model is not over fitting. A model that is over fit means that it is highly specific to the training data and does produce good results when classifying new data. The pruning process entails growing a very large tree and then reducing the number of partitions within the tree in order to provide better results. While pruning does reduce variance it also compromises on bias. In order to identify the most optimal tree size, cost complexity pruning is used to compare the number of terminal nodes in a large number of trees and determines the optimal trade-off between complexity/variance and bias. This process uses cross-validation and results in the optimal tree size (Hastie et al. 2009, p.311).

Figure 1 depicts the splits in the data as well as the values presented at the terminal nodes. If we follow along with the logic presented in the figure, it starts at the top with the all the observations. Then the first split is at the yay physician variable, so if a congress person did not vote yay for the physician vote, they are classified as a democrat, since the value of the terminal node is one. If the congress person did vote yay, the tree moves onto the synfuels vote to classify at another split. The next split is at the yay to synfuels variable. If a congress person did not vote yay to the synfuels vote, they would be classified as a republican, and if they did vote yay, the tree

move onto the missile vote to classify at another split. Finally, the last node is the yay to missile variable. If a congress person did not vote yay to the missile vote, they are classified as a republican, and if they did vote yay, they are classified as a democrat.

Figure 1: Classification Tree



For bagging (bootstrap aggregation), the same process is followed as the regular classification tree, except bagging utilizes all the input variables within the training dataset ($\rho = 32$) and aggregates the bootstrapped classification outputs or "fitted values" to determine its predictions. Since this is a binary classification problem, the aggregation uses a majority vote to make the final classification each test observation. That is, which ever classification/category has the most trees predicting that output will be the overall model's prediction for that test observation. By taking the aggregate of these different bootstrapped classification trees, the variance of is reduced (divided) by the number of observations. Since the variance is being reduced through aggregation, there is no need to prune the classification trees when bagging (Hastie et al. 2009, p.317).

For random forests, a slight modification is made to the bagging model. Instead of using all the variables when building a tree, random forests use a subset of inputs when determining where

to make its partitions. The subset of predictors usually takes the square root of the total number of inputs, which would round up to six in this case ($\rho = \sqrt{32}$). By modifying the model this way, the variance is reduced further as the trees that are aggregated are less correlated. Even though some inputs are not being used all the time, the use of different inputs over a large number of trees will add a random element (Hastie et al. 2009, p.319).

4. Results

The results from each of the models provide strong performance when predicting whether the class of the test observations are either democrat or republican. Of the 145 observations in the test data, the linear probability model, and the classification tree yielded error rates of 4.13%. The two more advanced methods, bagging, and random forests did the best as they performed slightly better than the other two models mentioned with an error rate of 3.45%. The logistic regression model's performance was surprisingly the worst, since it was not able to classify 10 observations correctly, which yielded an error rate of 6.897%. The table below summarizes the results of each model when attempting to address the classification problem.

Table 2: Classification Errors of Each Model

Model	Inputs	# of Classification Errors
Linear Probability Model	All inputs	6/145
Logistic Model	All Inputs	10/145
Classification Tree	All Inputs	6/145
Bagging	All Inputs	5/145
Random Forest	6 Inputs ($\sqrt{32} \approx 6$)	5/145

In conclusion, all the models performed very well. For a point of reference, if we were to compare these results to a "naïve" classification (i.e., classifying everything as democrat) the resulting error rate would be 38.6%, which is a lot worse than each of the models demonstrated in this paper. Therefore, bagging, and random forests performed exceptionally well and would be the

best choices, the linear probability model and classification tree are simpler but also perform very well when applied to this classification problem.

References

Congressional Quarterly Almanac. (1984). "98th Congress, 2nd Session." *Volume XL: Congressional Quarterly Inc.* Washington, D.C., 1985

Hastie, T., Tibshirani, R., Witten, D. & Gareth James. (2009). *An Introduction to Statistical Learning: with Applications in R*, 2nd edition. Springer.