CYO - Adult - Report Submission

Nikhil

1/5/2021

Overview

In this project, I used information on adults from the 1994 Census database to predict whether individuals had an income higher than $50,000 or not based on certain known features about the individuals.

The dataset I used for this project comes from the UCI Machine Learning Repository. It consists of 14 features and 1 outcome. The features are listed here:

```
##  [1] "age"            "workclass"      "fnlwgt"         "education"
##  [5] "education-num"  "marital-status" "occupation"     "relationship"
##  [9] "race"           "sex"            "capital-gain"   "capital-loss"
## [13] "hours-per-week" "native-country"
```

The outcome is a boolean value (True or False) indicating whether the income is greater than 50K or not.

To predict the outcome using the features, I created two different non-linear and non-logistic models to capture the relationship between the features and the outcome. One model was a Decision Tree and the other was a Random Forest. These models tend to perform better than linear or logistic models, so that is why I used them.

Methods

Before creating and using the models, I had to perform a few data preparation steps.

First, I downloaded the dataset and split it into two different datasets. One would be used for training the model and the other would be for testing the final model. I split it so that 90% of the original data would be the training set, and the rest would be the testing set. I did it this way so that the models can be better trained with more data to power them.

To create the models, I called the built-in *train* function from the *caret* package in two different instances. The first call to *train* used the parameter `method = "rpart"` - This was for the decision tree model. The second call statement used the parameter `method = "rf"`, and was for the random forest model. An example of my call statements is below:

```
train(train_x, train_y, method = "rpart") #This statement creates the Decision Tree model
```

The random forest model comes with a warning, because the time it takes to create that model is *very* slow.

After creating the models and training them on the training dataset, I ran them on the test dataset to come up with predictions for the test set's outcome. Here is an example of that kind of statement:

```
predict(train_rf, test_x) #Where train_rf is the trained random forest model
```

Results

Predicting using the models produced two kinds of results. One was the best tuning parameter used in each model. The second was the accuracy of the model's predictions when compared to the desired test outcome.

The tuning parameter in each model case (Decision Tree and Random Forest) was the complexity parameter, or "cp". These are the actual best complexity parameters for each model:

```
## Decision Tree:  0.03755669
```

```
## Random Forest:  2
```

The accuracy of each model was computed by taking the overall accuracy, which was the proportion of all the times the model correctly predicted an individual's income level being over 50K or not. Here is an example of what that kind of statement would look like:

```
mean(rf_preds == test_y) #rf preds is the set of predictions, and test_y is the set of actual outcomes
```

And these are the final accuracies:

```
## Decision Tree's accuracy:  0.8265275
```

```
## Random Forest's accuracy:  0.8569235
```

Conclusion

Based on the results, it seems that the Random Forest model performed somewhat better than the Decision Tree model. Therefore, someone who wants to predict whether income is high or low for a given adult could potentially benefit from using my Random Forest model. Obviously, the limitations are that the Random Forest model is only useful for predicting on data that has the same features as my training dataset. Also, the model only works for the feature values that are present in my training dataset. It will not, for example, help predict income category for someone whose native country is not included in my training set.

If I continued this project in the future, I would find a way to better train my Random Forest model, either by feeding it more data or by doing some feature engineering on the training dataset.