Lab 7: Exploring Tree-Based Methods by Predicting Tax Compliance

This assignment asks you to work a bit more closely with the data from Malawi on tax compliance that had a chance to explore it in lab 4, when you practiced clustering with the possessions variables. In this lab, you will instead build a moodel to attempt to predict whether a vendor gave evidence of having paid the market tax or not, based on a large number of demographic variables.

Warm up

Let's warm up with some simple exercises. Open the `lab7.qmd` Quarto file. Update the YAML of your Quarto file with your information, knit, commit, and push your changes. Make sure to commit with a meaningful commit message.

Please make sure to keep committing throughout while working on the assignment!

Data

The data for this lab, contained in `data/vendor_data.RData`,[1] come from a project that Dr. Hoellerbauer is working on that analyzes a randomized control trial (RCT) focused on tax compliance carried out in Malawi. In Malawi, market vendors have to pay a fee (a flat tax) in order to be able to sell goods or services in a market. This fee is collected by fee collectors, who make the rounds each day.

Compliance is not optimal for a variety of reasons, a main one being that markets are underfunded and so are often in poor conditions. This causes market vendors to not want to pay the

[1] This is a `.RData` file, which is a file format native to R that allows us to save one or more R objects in their entirety and then load them into our environment later on. We use the `load()` function to do so in the following way: `load("folder/file_name.RData")`

Note that because saving an R object in an `.RData` file saves the name of the object (usually called a variable in other languages), we do **not** need to assign the `load()` function a return value.

market tax, which in turn makes it harder for the government to improve the markets (this is called a *vicious cycle*). The RCT ran from October 2017 to March 2019 in 128 markets in 8 districts in Malawi. The goal of the RCT was to investigate different ways to break out of this vicious cycle in Malawian markets. [^If you are interested in this topic, I am happy to talk more about it during office hours!]

The data you will be using was collected via an in-person survey of market vendors carried out between October 2018 and January 2019. In total, 12,370 market vendors completed the survey. 2,531 of these vendors were asked a longer form of the survey, with more questions. We will be working with the responses from these individuals for this lab.

Please see the README page of the `data` folder of the GitHub repository for a complete codebook of the data.

Packages

You will need the `tree`, `randomForest`, and `gbm` packages for the fitting decision trees, random forests, and gradient boosting machines. You will need `ROCR` or `pROC` to make ROC curves and calculate the AUC. You will most likely need the `caret` package for some of the cross-validation that you will be doing, if you do not want to do it by hand. We also strongly recommend that you use the `tidyverse` suite of packages for data visualization and data wrangling. The `patchwork` package may also come in useful for combining plots.

The `doParallel` package, which will make the cross-validation grid-search for the GBM model you will fit easier, is already included in the chunk below.

```
# load packages here
library(doParallel) # for parallel processing.
```

Exercises

```
# load data here
```

Data Visualization

Q1

Make a visualization that shows the relationship between our outcome `recent_receipt_7` and *one* of the predictors that you are interested in. Use an appropriate visualization for the comparison you are making. What does your visualization tell you?

Make another visualization that shows the relationship between any two of our predictors. Once again, use an appropriate visualization for the comparison you are making. What does your visualization illuminate?

Train-Test Split

Q2

Drop all rows where `recent_receipt_7` is `NA`. Next, turn`recent_receipt_7`, the outcome variable, and `district` into factors. Then, create a 75/25 training/test split, setting the seed to `20` beforehand.

Also, answer this question in a few sentences:

- What is the trade-off between test set size and training set size?

Model Fitting

Include all variables in your models except for `market`. [2]

Q3: Single Decision Tree.

Use cross-validation to find the optimal cost-complexity parameter for a single classification tree using the `tree` package. Use 5 folds and set the seed to `21` beforehand.

Plot the optimal tree. What does it tell you about who had evidence of paying the tax or not?

Then, plot the ROC curve and calculate the accuracy and the AUC for this classifier *using the test data*.

> **!** `predict.tree()`
>
> When using the `predict()` function, set `type = "vector"` for the ROC curve and the AUC; set it to `"class"` when calculating the accuracy. Note that for the predicted probabilities, the `predict()` method for `tree` objects actually outputs a matrix, with probabilities of each class.

[2] Ideally, we would include information about the market in our analysis, as tax payment dynamics could be different market from market.But we would end up with 128 dummies if we one-hot-encoded the 'market' variable, which balloons the number of possible splitting variables and makes the problem computationally very difficult. It is important to note, however, that by not including 'market', we are implicitly arguing that, conditional on al other variables, markets are the same,

Q4: Random Forest

Use the `caret` package's cross-validation framework to find the optimal random forest by searching over `mtry = 10:16`[3] with 5-fold cross-validation.[4] Inside the `train()` function, also add `na.action = na.roughfix`.[5] Set the seed to `22` before doing cross-validation.

Nicely plot the variable importances for the optimal fit on training data. Plot only the top 10 most influential variables. Which variables seem most important for predicting whether a vendor would be able to present a receipt or not?

Then, plot the ROC curve and calculate the accuracy and the AUC for this classifier *using the test data*.

> **!** `predict.train()`
>
> The way in which `caret::train()` passes data to the machine learning algorithm chosen by the user can complicate prediction in some cases, especially when there are factor level predictors. Therefore, instead of calling `predict(trained_object$finalModel)`, which will call the `predict()` method for whatever the class of the `finalModel` is (in this case `"randomForest"`), you should use `predict.train()`. In other words, call `predict()` *directly* on the output of the `train()` function.
> Set `type = "prob"` for the ROC curve and the AUC; set it to `"raw"` when calculating the accuracy. In both cases, add `na.action = na.roughfix` to the `predict()` function call. This is an imperfect solution but prevents `NA`'s from being dropped.

Q5: Gradient Boosting Machine

Use the `caret` package's cross-validation framework to do 5-fold cross-validation to find the optimal GBM by searching over the parameter grid in the following code chunk. Set the seed to `23` before doing so. Inside the `train()` function, also add `na.action = na.pass`.[6] **Please note that the grid-search cross-validation could take a considerable amount of time.** To help with this, the code chunk below makes it so that iterations are run in parallel – at the same time. This can speed up computation considerably.

[3] Ideally, for a random forest, you would search from using only one variable at each split to all variables. However, this could take a considerable amount of time. That is why for this lab you are searching over a more constrained space. This can still take several minutes.

[4] In the example code, you saw how to use 'caret' for GBMs, but it works the same way for random forests, you just have to set 'method = "rf"' in the 'train()' function.

[5] Tree algorithms generally do fine with 'NA' values, which is a huge advantage of trees. However, the way trees are implemented in 'randomForest' cannot handle missing values. 'na.action = na.roughfix' does a very rough imputation of missing values, by default replacing them by median/mode and then trying to calibrate them somewhat. This adds computation time.

[6] Unlike 'randomForest', 'gbm' can handle 'NA' values fine; we need to do this to override the 'train()' default, however, which is to drop missing values.

Also, you may see a list of warnings printed saying `Warning: variable num: varname has no variation.`. This warning appears because in some of the folds, a certain variable has no variation. You can ignore this for now.

```
# set up parallel processing
gbm_clusters <- makeCluster(detectCores() - 1)
registerDoParallel(gbm_clusters)

# grid to search over
expand.grid(n.trees = c(3000, 5000),
            interaction.depth = c(1, 2, 3, 4),
            shrinkage = 10^(-3:-1),
            n.minobsinnode = 10)

### ALL OTHER CODE FOR FITTING GBM HERE

# stop cluster when done; this is very important!
stopCluster(gbm_clusters)
```

Nicely plot the partial dependence plot between the outcome and `customers_pr_day_trim_99`. What does it tell you about the relationship between the number of customers a vendor reports having each day and the probability that the vendor presented a receipt?

> **!** Hint
>
> Pay extra attention to the values output by `plot(..., return.grid = TRUE)`. Compare it to the output of `predict()` called on your `"train"` object.

Then, plot the ROC curve and calculate the accuracy and the AUC for this classifier *using the test data*.

> **!** `predict.train()`
>
> As with `randomForest`, you should call `predict()` directly on the output of the `train()` function.
> Once again, set `type = "prob"` for the ROC curve and the AUC; set it to `"raw"` when calculating the accuracy. In both cases, add `na.action = na.pass` to the `predict()` function call.

Introspection

Q6

Is the peformance of any of any of the modeling approaches good, do you think? What might this mean?

Which modeling approach resulted in the best fit? What were the optimal tuning parameters of the best-fitting model? What does this tell you about the relationship between the predictors and the outcome?

Q7

In this homework, we have been working to predict when a Malawian market vendor will pay the market tax or not. From the perspective of the government, this would be a great tool, because they could potentially identify shirkers ahead of time. But what do you think could be some of the downsides of this prediction task? *In your opinion*, is prediction of this kind mostly good idea?

Final Steps

When you have made your final edits, remember to render, stage the changed files, then commit, and push to the repo.

Don't forget to submit your pdf on Canvas.

Grading

Please put your names on your answer document and add the date. You won't get points for doing this, but you will *lose* points for not doing so.

Remember to comment your code. You don't get points for commenting, but you will *lose* points for not commenting your code.

For clarity's sake, you should label your code chunks (this helps make debugging easier). You will *lose* points for not naming chunks.

If an exercise asks a question, you should answer it in narrative form and not just rely on the code. You will be *penalized* otherwise.

Please suppress warnings and messages. You will *lose* points if you do not suppress warnings and messages.

If you are asked to make a visualization, make sure that it is presentable, professional in appearance, has a title, and has properly labeled axes. You will *lose* points if you do not do this.

Points Breakdown

The different components of this lab are worth the following points:

- Q1: 8 pts
- Q2: 2 pts
- Q3: 6 pts
- Q4: 6 pts
- Q5: 6 pts
- Q6: 2 pts
- Q7: 2 pts
- Workflow (includes making sufficient commits): 4 pts

Total: 36 pts