Lab 2 - Machine Learning for Social Science

To be handed in no later than October 9th, 10:00. The submission should include code, relevant output, as well as answers to questions. We recommend the use of RMarkdown to create the report.

Part 1: Social Network Ad Purchase (cont.)

In the first part of this lab, we will again consider the *ad purchase* data that we used in lab 1 (containing information about individuals' purchasing behavior under exposure to online ads). As in lab 1, we will build classification models to predict ad purchase (0/1) based on a set of covariates about individuals (including `Age`, `Gender` and `Salary`). In contrast to lab 1, we here focus on *non-parametric methods.*

1. Begin by importing the file "`Kaggle_Social_Network_Ads.csv`". Format the outcome variable `Purchased` as a `factor` variable (this is required for the subsequent analysis using the `caret` package). You may also delete the `user_id` column, as it will not be used.

2. The first non-parametric method you will use to predict ad purchase is the *k-nearest neighbors* algorithm. As we talked about in the lecture, it has one key parameter, k, that the researcher must set. Use the `caret` package to implement a 10-fold cross-validation procedure that examines the test accuracy of *kNN* under different choices of k (hint 1: set `method="knn"` to estimate a *kNN* with `caret`| hint 2: use the `tuneGrid` argument to specify your grid of k values). Consider the following values of k: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50\}$. Because *kNN* is sensitive to the scale of variables, we must standardize our variables prior to estimation (hint: you can add the following argument to `train()`: `preProcess=c("center","scale")`, to do so). From the results, plot the *test accuracy* against k (hint: results can be extracted from the `train` object by using `$results`). Interpret this plot. How does the performance—for the best k—compare to that of *GAM* and *standard linear regression* (i.e., your results in lab 1)? Which do you prefer? Why?

3. Suppose we now learn that those who collected the data have retrieved some additional information (in the form of digital traces) about these 400 individuals. This extra information consists of 20 variables $X_1, X_2, \ldots, X_{20}$. The people providing us with the data suggest that *some* of these variables could be very relevant for predicting `Purchase`, but they also note that many of them likely are irrelevant. The problem is, they do not know which is which. This new data set is stored in the .csv file "`Kaggle_Social_Network_Ads_Augmented.csv`". Import it to R and process it the same way you did in #1.

4. Repeat task #2 for this new data set (from step #3). How does the prediction accuracy change? How do you explain this difference (or lack thereof)?

5. Motivated by the results obtained in #4, we want to explore an alternative method. Based on the properties—which we discussed in the lecture—of *decision trees*, do you think it has the potential to perform better than *kNN* on the data set from #3? If yes, why?

6. Estimating decision trees is what you will do now. In `R`, we can use the package `rpart` for this. Decision trees have two main parameters that a researcher must choose prior to estimation: (*i*) the minimum number of observation in each leaf node (in `rpart`: `minbucket`), and (*ii*) the complexity parameter α (in `rpart`: `cp`). In practice, the former is often set heuristically (using some rule-of-thumb) and the latter using cross-validation. Please do the following:

a. Answer why it generally is *not* a good idea to set `minbucket` to either a *very large* number or a *very small* number (relative to the size of your data).

b. Use the `caret` package (`method="rpart"`) to implement a 10-fold cross-validation procedure to assess how the test accuracy changes as a function of the complexity parameter (ISL: α; `rpart`: `cp`). Hint: use the `tuneGrid` argument to specify your grid of *cp* values. Consider the following values for *cp*: $\{0, 0.01, 0.025, 0.05, 0.10, 0.25, 0.5, 1.0\}$. (note: `minbucket` is here set to a default value=10). Plot the *test accuracy* against *cp* (hint: extract results as you did in #2 and #3). Interpret this plot. How does the accuracy for the best *cp* compare to *kNN*? What do you attribute this difference to?

c. Use the `rpart.plot` package to plot the decision tree you found to be the best in *b* (hint: you can extract the model with the highest accuracy from a `caret` model using `$finalModel`). First report how many *internal* and *terminal* nodes this tree has. Then, provide a interpretation. Does any of the "new" variables ($X_1, \ldots, X_{20}$) show up in the tree?

d. Use `caret`'s `varImp()` function to calculate *variable importance* scores. Interpret.

Part 2: Bernie Sanders and Donald Trump tweets (cont.)

In the second part of this lab, we will reconsider the *Bernie vs. Trump* twitter data set that we used in lab 1. As in lab 1, we shall build classification models to predict who authored a given tweet based on its linguistic content, and to identify which words are the most discriminative between Bernie and Trump.

1. Import the file "`trumpbernie2.csv`" to `R` (note: to reduce computational time, I have reduced the number of columns for this lab), and format the outcome variable `trump_tweet` as a factor variable (again, this is needed for the `caret` package to recognize it as a *discrete* variable.)

2. Use `caret` to implement a 5-fold cross-validation procedure that estimates the *test error* for a *decision tree* with different *cp* (ISL: α) values. Consider the following *cp* values: $\{0, 0.001, 0.01, 0.1, 0.25\}$. Report the accuracy of the best configuration. How does this compare to the performance of a *standard logistic regression* (LR) and a *ridge regression* (RR)? You do not need to estimate the LR and the RR yourself—I provide their performance here: Accuracy(LR)=55%. Accuracy(RR)=86%. Why do think this is the case? Doing the following two things might help you answer this question:

   a. Plot the tree which achieved the highest test accuracy (hint: again, you can extract the best model using `$finalModel`, and plot it using `rpart.plot`). Is it a big or a small tree?

   b. Calculate variable importance. How many variables have non-zero values?

3. Given the results in #2, we want to explore other alternative methods. Using once again the `caret` package, you shall now implement a 5-fold cross-validation procedure that fits a *random forest* model (`method="rf"`) with different *mtry* values. *mtry* controls the number of variables that are considered at each split (corresponding to parameter m in ISL). Specifically, you shall consider the following values for *mtry*: $\{1, 5, 10, 25, 50, 200\}$ (hint: again, you specify this grid using the argument `tuneGrid`). Another important parameter of random forest models is the *number of trees* to use. Here you shall use 200 (hint: you may enter `ntree=200` as an argument in `train()` to do so). The estimation may take a couple of minutes to finish. Once it has finished, please do the following:

   a. Plot *accuracy* against *mtry*. Interpret this plot. Does decorrelating the trees seem to help performance?

   b. Relate the performance of the best random forest model to the best decision tree model. Do you find a meaningful improvement using random forest? Why do you think that is (or isn't)? Examining the variable importance scores may provide some clues; how many non-zero variables do you find for the random forest?

c. Relate the performance of the best random forest model and the ridge regression from #2. What do you think this difference indicates about the data, and more specifically the relation ship between X and y?

d. Consider the variable importance scores in a bit more detail: does the variables that appear in the top 20 align with your expectations? Note that variable importance does not reveal the *direction*; e.g., whether a certain word is associated with Trump or with Bernie. For this, we can use the `randomForest`'s function `plotPartial()` which produces a partial dependence plot for a variable of choice. Select three variables (words) which you find interesting, and create three separate partial dependency plot. In order to interpret the y-axis of the plot, use the argument `which.class` to specify which class ("0" or "1") should represent the positive class.