Homework 5

For this homework you will create a github repo, set up github pages, clone the repo to your computer as an R project, create a `.qmd` file, and push those changes back to github to create a webpage! You'll submit the link to your github pages site (the one that looks like a nice website).

The steps for setting things up exist in the first two homework assignments and are not repeated here.

- Create a new `.qmd` document that outputs to HTML. You can give this a title of your choosing. Save the file in the main repo folder.
- In this document, answer the questions below.

The purpose of this homework is to get practice fitting models using the `caret` package.

Task 1: Conceptual Questions

On the exam, you'll be asked to explain some topics. How about some practice?! Create a markdown list with the following questions:

1. What is the purpose of using cross-validation when fitting a random forest model?
2. Describe the bagged tree algorithm.
3. What is meant by a general linear model?
4. When fitting a multiple linear regression model, what does adding an interaction term do? That is, what does it allow the model to do differently as compared to when it is not included in the model?
5. Why do we split our data into a training and test set?

Task 2: Fitting Models

We'll use the data set called `heart.csv` available here. This data set gives information about whether or not someone has heart disease (`HeartDisease = 1 or = 0`) along with different measurements about that person's health. The data comes from here if you'd like to read a bit more about it.

Quick EDA/Data Preparation

1. Quickly understand your data. Check on missingness and summarize the data, especially with respect to the relationships of the variables to `HeartDisease`.
2. Create a new variable that is a factor version of the `HeartDisease` variable (if needed, this depends on how you read in your data). Remove the `ST_Slope` variable and the original `HeartDisease` variable (if applicable).

3. We'll be doing a kNN model below to predict whether or not someone has heart disease. To use kNN we generally want to have all numeric predictors (although we could try to create our own loss function as an alternative). In this case we have some categorical predictors still in our data set: `Sex`, `ExerciseAngina` `ChestPainType`, and `RestingECG`.

Create dummy columns corresponding to the values of these three variables for use in our kNN fit. The caret vignette has a function to help us out here. You should use `dummyVars()` and `predict()` to create new columns. Then add these columns to our data frame.

Split your Data

Split your data into a training and test set. (Ideally you'd do this prior to the EDA so that info from the EDA doesn't bias what you do modeling-wise, but that isn't usually done.)

kNN

Next, we'll fit a kNN model. The article here gives a great example of selecting the number of neighbors to use with the `caret` package.

You don't have to use all the variables from your dataset when fitting the model. However, you should only use numeric variables.

They use repeated 10 fold cross-validation. Although computationally intensive, doing repeated CV helps to give a more stable prediction of CV error. This is similar to how a mean is less variable than a single value. Since there is some inherent randomness in doing a CV computation, we can get an overall more stable result by averaging a few runs of the CV algorithm!

Train the kNN model. Use repeated 10 fold cross-validation, with the number of repeats being 3. You should also preprocess the data by centering and scaling. When fitting the model, set the `tuneGrid` so that you are considering values of `k` of 1, 2, 3, . . . , 40. (Note: From the help for the `train()` function it says: `tuneGrid` A data frame with possible tuning values. The columns are named the same as the tuning parameters. The name of the tuning parameter here is `k`.)

Lastly, check how well your chosen model does on the test set using the `confusionMatrix()` function.

Logistic Regression

Using your EDA, posit three different logistic regression models. Note: You don't have to use the dummy columns you made here as the `glm()` function (and the `caret` implementation of it) can handle factor/character variables as predictors.

Fit those models on the training set, using repeated CV as done above. You can preprocess the data or not, up to you.

Identify your best model and provide a basic summary of it.

Lastly, check how well your chosen model does on the test set using the `confusionMatrix()` function.

Tree Models

In this section we'll fit a few different tree based models in a similar way as above!

Choose your own variables of interest (as with logistic regression, this models can accept factor/character variables as predictors). Use repated 10 fold CV to select a best

- classification tree model (use `method = rpart`: tuning parameter is `cp`, use values 0, 0.001, 0.002, . . . , 0.1)
- a random forest (use `method = rf`: tuning parameter is `mtry`, use values of 1, 2, . . . , # of predictors (bagging is a special case here!)
- a boosted tree (use `method = gbm`: tuning parameters are `n.trees`, `interaction.depth`, `shrinkage`, and `n.minobsinnode`, use all combinations of `n.trees` of 25, 50, 100, and 200, `interaction.depth` of 1, 2, 3, `shrinkage = 0.1`, and `nminobsinnode = 10`; Hint: use `expand.grid()` to create your data frame for `tuneGrid` and `verbose = FALSE` limits the output produced

Lastly, check how well each of your chosen models do on the test set using the `confusionMatrix()` function.

Wrap up

Which model overall did the best job (in terms of accuracy) on the test set?