Assignment 3

In brief:

Using the supplied dataset in the form of a CSV file, and the supplied Shiny App, you are tasked to extend this app in order to find the best performing regression model. The app demonstrates the issue of candidate model selection and has been designed around the caret package. This demonstrates the idea of "no-free lunch" - there is no method that works optimally in all situations.

The caret package is a framework for best-practice in regression and classification. It enables you to apply each of 237 different methods from Data Science to your data and optimally sets any hyper-parameters through re-sampling. The candidate models can then be compared fairly in order to select several top models that should then (and only then) be assessed against the official test data.

In detail:

The shiny template has already been developed for you. You will need to familiarise yourself with this code and be comfortable modifying it. The app takes the variable labelled “Y” as the outcome variable. It should be a matter of adding method-tabs (and populating them with plots) using the existing tabs as templates. You may also change the pre-processing of the existing method-tabs. Each model might require different summary and plot facilities that are unique to that method. For example, observe how different the outputs are for the supplied example methods. Use your judgement to show the most relevant information in your app for the methods you add.

You should research the caret documentation https://topepo.github.io/caret/ (especially sections 6 and 7). Notice that caret maintains a list of the methods and their characteristics (tags) which may help in choosing which methods to try. Clearly we are only dealing with regression methods in this assignment. These are made available to you in the tab called "Available methods" - **notice that the "ban" icon indicates that the package is not yet installed**. You will need to deal with installing packages yourself.

Choose additional regression based method types to add to the Shiny App in a manner similar to and consistent with the three models (and the null model) already in place.

Before you submit your revised app, ensure that:

- the code runs after you have done Session/Restart R & Session/Clear Workspace... (in RStudio)
- the default for the radio button selecting the best model is set to your best model.
- the defaults for the pre-processing are sensible & optimal for the particular method. Take care to get the order of the steps sensible & optimal, as well.
- You also need to submit a static report on your process of finding the best model. An example report is included in the assignment. It should include (as a minimum)
    - a screen shot of the model selection visualisation
    - a table of the methods, pre-processing employed, re-sampling results (hyper-parameters and metrics)

- a statement as to what is the best performing model. Assume "best" means "minimum RMSE".
- an explanation on how the best model works - do some research on your best performing method so that you can explain how it works in your own words. Include references to your on-line resources.
- discuss why you think your best method suits the data as well as it does.
- a screen shot of the predicted versus actual visualisation of test data
- statistics for the best performing model based on the test data
- how would the optimum model change if transparency were very important? That is, what would be the best performing transparent model? Justify why you believe this model to be transparent – if necessary prove this by exposing its rules and/or coefficients.

Some hints:

➔ Slower-to-train does not mean better. Avoid slow methods unless you have plenty of spare time. How slow is too slow? This is difficult to say but I reckon a model that takes more than 1 hour to train on this small set of data is not a good use of your limited time.

➔ You may add extra variables (derived from the existing predictors). For example non-linear functions and/or interactions.

➔ If your method crashes, turn off parallel processing in order to catch the error messages and warnings. This will run slower.

➔ Parallel processing means that re-samples can train concurrently. If you are using the university servers, be aware that you can bring the servers to their knees very easily. If you receive emails about compute resources running low, turn off parallel processing for a while.

➔ You should not need to substantially alter the shiny app. If you think you need to, come and speak with me first.

➔ In global.R there is code that converts your pre-processing choices into a sequence of recipe steps in the order that you specify them. If you are unsure what a particular pre-processing choice does, this is where you can work it out. This is also where any preprocessing parameters are set. You may add further recipe steps (in global.R) but I suspect you will not need to.

➔ While bedding things down (i.e. testing & debugging), making a smaller train-set (i.e. train/test slider set to say 30%) may make your training run faster.

➔ A diverse set of methods makes a good candidate model set. The method tags are also relevant to matching the data + preprocessing to the method. For example, if a method does not tolerate missing values then any missing values in the data must be dealt with by the preprocessing.

➔ If a package is required that you have not yet installed, caret will prompt you from the console. **If your app seems to freeze, check the console for a dialogue about installing a missing package.**

- ➔ Searching both possible methods and preprocessing options together can be torturous. It is a good initially idea to seek to create a set of preprocessing steps that suits all the methods you expect to use. Later on, when you have found your top 2 or 3 methods, you can explore how to tune your preprocessing recipe to get a little more out of them.
- ➔ There is an element of luck involved in finding the best method so maximise your luck by trying many methods. You should look at more than 10 (but less than 30) methods before you can be confident that you have found a reasonably optimal model in this assignment.
- ➔ Set your method-specific preprocessing *default* value to your recorded choices so I can reproduce your tabulated results in the report.
- ➔ Be careful not to sneak ahead and peek at the test results to choose your best model. Use the re-sampled results to make this decision (the re-sampled results are NOT training results and are fair to use in order to choose the best model)
- ➔ The code contains comments intended to help you make changes. Look out for these. Note that the **inputId**'s need to be constructed in a standardised way to work properly with this app.
- ➔ The "SavedModels" folder contains files that you can reload later. This is to make your life easier. Bear in mind that some model files get large and may not be able to be included with your finished submission.
- ➔ If you have limited memory on your laptop, consider "forgetting" your trained models so that the memory they used can be freed up for further training.
- ➔ When you deliver your solution include the "SavedModels" folder to save me time BUT ensure you are delivering a only your best (say 5) models to keep the submission small enough.
- ➔ When you think you have finished, re-read this material to ensure you have not forgotten anything.

Compute Resources

You should use your own laptop if it has lots or memory and cores and a stable R/RStudio installation. If you need more resources, please use terminal sessions on **mathstudent.canterbury.ac.nz**

Distance students will not be able to access this server. Distance students should use the MADS terminal servers.

Remember to be considerate of others. If the servers become overloaded they may need to be rebooted.

Submission

The source code for the modified app: server.R, ui.R, global.R

A pdf report (see the example)

The "SavedModels" folder of files. This may get too large to upload into LEARN. If your zip file is too large prioritise the best performing models and upload what you can.

Marking

Since there is an element of luck involved in whether you find a good model, I will also judge you on

- your choice of pre-processing steps (especially their logical order)
- the number and choice of candidate methods coded in the app and discussed in your report.
- the completeness and clarity of the report
- the explanation of how your best method works
- your choice of best transparent method