Pulsar Classification Using tidymodels

Your results should be delivered as an HTML webpage generated using R Markdown. Make sure to include all code and results. Provide text to describe your methods and results. This should read like the Methods and Results sections of a paper.

Grading Criteria

- Correctness and completeness of code (16 Points)
- Description of process and results (12 Points)
- Proper implementation of tidymodels (8 Points)
- Webpage formatting (4 Points)

You have been provided with a CSV file obtained from the UCI Machine Learning Repository representing data collected for pulsar candidates. The goal of this exercise and the original dataset is to create a model to differentiate pulsars from non-pulsars using a set of remotely sensed variables collected using a radio telescope. The dataset contains 16,259 examples of non-pulsar signals and 1,639 pulsars. The following 8 predictor variables are provided: mean, standard deviation, excess kurtosis, and skewness for the integrated profile and mean, standard deviation, excess kurtosis, and skewness for the DM-SNR curve. These variables are the first 8 columns of the table. The last column ("class") differentiates pulsars (1) and non-pulsars (0).

Use the tidymodels packages and generate code to:

1. Recode the "class" column as follows: 0 = "Not" and 1 = "Pulsar".
2. Randomly sample 1,600 samples from each of the two classes, or a balanced set of 3,200 samples (this is to speed up the exercise, as it would take much longer to train using the entire dataset).
3. Define a random forest model that uses 501 trees and an optimized *mtry* parameter, is implemented with the "ranger" engine, and is in "classification" mode.
4. Create a training/test split that includes 75% of the 3,200 samples in the training set and reserves the remaining 25% for testing. Stratify on the "class" field to maintain a balanced training and test set. Create tibbles of the separate training and testing sets.
5. Create a preprocess pipeline that normalizes all the predictor variables. No other pre-processing tasks are required.
6. Create a workflow containing the model and pre-processing recipe.
7. Define a set of assessment metrics to calculate that includes overall accuracy, recall, precision, kappa, ROC AUC, and PR AUC.
8. Define 5 folds to perform five-fold cross validation.
9. Tune the model and select the parameters that return the highest ROC AUC.
10. Finalize the workflow and train/fit the model.
11. Create an error matrix and obtain summary metrics from it.

Your Markdown webpage should (1) step through and explain the code and processed used and (2) include a summary of the results obtained.