I began by briefly testing preprocessing methods to reduce collinearity I noticed in the data. I favored PCA as a preprocessing method as it also reduced dimensionality greatly. Because it required 267 components to account for 95% variance in the full dataset, I tested retaining numbers between 220 and 270 components by examining accuracies after training a gradient-boosted model and a random forest model using 10-CV repeated 3 times (10CV3R) on the full dataset. At the range of component numbers tested, it seemed that the the number of components retained did not largely affect CV accuracy on the entire dataset - PCA reduced accuracy for all of the models tested. I decided to use PCA when dimension reduction or collinearity was of particular concern for a method, but to otherwise use the entire dataset.

At this point, I had come across the packages ‘caret’ and ‘caretEnsemble’. The caret package automates parameter tuning for a large number of models. The caretEnsemble package does the same for stacked models, using caret model outputs as inputs to the ensemble model. My strategy then became to test a number of stacked models composed of a variety of different input models. However, I also wanted to consider some other models without stacking because I knew that I did not have the time nor computational power to test a large number of different input model combinations for the stacked models, so I did not want to solely rely on stacking.

To decide which models to use as inputs for the stacked models, I read through the caret documentation and picked out a number of candidate models for whom the prediction methods sounded very different. I wanted to test the accuracy of the stacked models using repeated nested CV rather than splitting the data into training/validation/testing sets to improve the accuracy of my test error prediction.

Because I knew that the full nested CV process would be computationally intensive, I identified models that would benefit from expanded parameter grid searches prior to running the nested CV by examining accuracy estimates after training on the full dataset using 10CV3R. By default, the caret package tests 3 values per parameter on a preset reasonable range. I tested by first expanding that grid to 10 values per parameter for most models, and 5 per parameter on models that had three tuning parameters, then expanding farther if it seemed like doing so might help. In these cases, I expanded the grid search during the nested CV to include a range around the best tuning parameter for the full set. I purposefully did not make any decisions to further restrict the range of parameter space to avoid introducing bias in this step - for example, I allowed k to range from 3 to 25 even though the optimal k was consistently around 20. In the end, I also decided not to select any parameter values at this stage even if the value of the parameter did not seem to affect CV accuracy at all. I repeated the procedure for parameters in the stacked models using the previously identified grids for the input models.

I then used these parameter grids to tune the models for testing accuracy estimates in the nested CV procedure. My final first-level input models (listed here as the name which needs to be entered into the method argument of caret::train()) were knn, xgbLinear, xgbTree, RRF, parRF, nodeHarvest, ada, C5.0, naïve\_bayes, gbm, rf, cforest, rpart, hdda, lda and nnet. For lda and nnet, I decided to preprocess the dataset using PCA (scaled and centered prior to decomposition) that retained 95% variance to reduce collinearity and dimensionality. Candidate ensemble models included RRF, gbm, glm, knn, lda, nnet, rf and svmPoly.

For the nested CV procedure, I included 10 outer folds for prediction accuracy testing, 5 middle folds for training the ensemble models, and 5 inner folds for training the input models. I then duplicated the script twice and changed the seeds used to repeat the procedure three times. Finally, I tested some models on the whole dataset – RRF, rpart, parRF, nodeHarvest, C5.0, gbm, rf, cforest and hdda – using nested CV with 10 outer folds and 5 inner folds, where the inner folds were used for parameter tuning and the outer folds were used for prediction accuracy testing. Accuracy estimates for all nested CV predictions were averaged together for each model to determine which performed best. The best model out of all of these options was a stacked neural network, with a predicted accuracy of 78.99%. To generate unique parameter estimates for the models in the algorithm, I trained each model on the full dataset using 10CV3R. The final parameters for each included model in the ensemble are listed in the table below.

|  |  |
| --- | --- |
| method | Final parameter values |
| knn | k = 17 |
| xgbLinear | nrounds = 150, lambda =0, alpha = .1, eta = .3 |
| xgbTree | nrounds = 50, max\_depth = 125, eta = .1, gamma = 0, col\_sample\_bytree = .6, min\_child\_weight = 1, subsample = 1 |
| RRF | mtry = 50, coefReg = 1, coefImp = 0.5 |
| parRF | mtry = 250 |
| nodeHarvest | maxinter = 4, mode = mean |
| ada | iter = 50, maxdepth = 6, nu = .01 |
| C5.0 | trials = 20, model = tree, winnow = F |
| naïve\_bayes | laplace = 0, usekernel = F, adjust = 1 |
| gbm | n.trees = 50, interaction.depth = 13, shrinkage = .02, n.minobsinnode = 10 |
| rf | mtry = 275 |
| cforest | mtry = 350 |
| rpart | cp = .04 |
| hdda | threshold = .93, model = all |
| nnet | size=1, decay=0.3 |
| lda | none |
| meta nnet | Size=1, decay=0.5 |