Eda Gür

50488

**Comp491 HW7 Report**

I have used a random forest algorithm for the given binary classification problem. As stated in the homework documentation, I have applied cross validation on the possible values for the parameter which is the number of trees in the growing forest. We have learnt about this algorithm when studying combining multiple learners. This algorithm chooses D different features each time and trains accordingly and with these trees that are created, we get the forest. I chose this algorithm because I thought that it would be reliable to have a multiple learner algorithm. When combining learners, the false predictions can be “neutralized” by true predictions: f\*(random forest) = f1+f2+…+f#trees

I have retrieved the data matrices as given in the example algorithm. Then, I converted the label data which was in frame format to numeric format. Factors 0-1 translate to numeric 1-2. I retrieved the positive and negative indices from as.numeric( y\_train ). After that, I set my fold count (nfold) to 5 and possible set of number of trees (ntree) to {100,200,300,400,500}. I have wanted to test for a greater number of trees, but the memory blew up on every computer I tried. So, I set the limit to 500. I have created a 5\*5 empty auroc matrix to store the auroc value of each fold-ntree pair.

Then, I started my cross-validation by using a for loop for each fold in 1:nfold. We have learnt in class that when creating folds, we should be careful about the ratios of class samples in each fold. For example, in this question negative to positive class ratio (prior probability) in this question is approximately 5.5 to 1. We should preserve this ratio in all folds. In order to achieve this, I have separated the training data of the positive and negative classes. I took one-fifth of both the negative instances and the positive instances and r-binded them to a validation set for each fold. I have created the accompanying label data. The remaining are my training set for the fold and I made the accompanying label data for them, too. After that, it is time to check for each possible value in the ntree set. This results in an inner for loop. In this for loop, I use the rfsrc method of the randomForestSRC library to train my algorithm:

rfsrc(formula = y ~ ., data = cbind(training, y = as.factor(training\_label)), ntree = ntree)

Then, I use the predict function to predict the classes for the validation set. I supply the resulting state from the previous function into this predict function:

predict(state, newdata = validation)

I create the roc curve using the prediction results (prediction$predicted[,2] ) and the factorized form of the labels of the validation set. I find the AUC value and place it in the AUROC matrix.

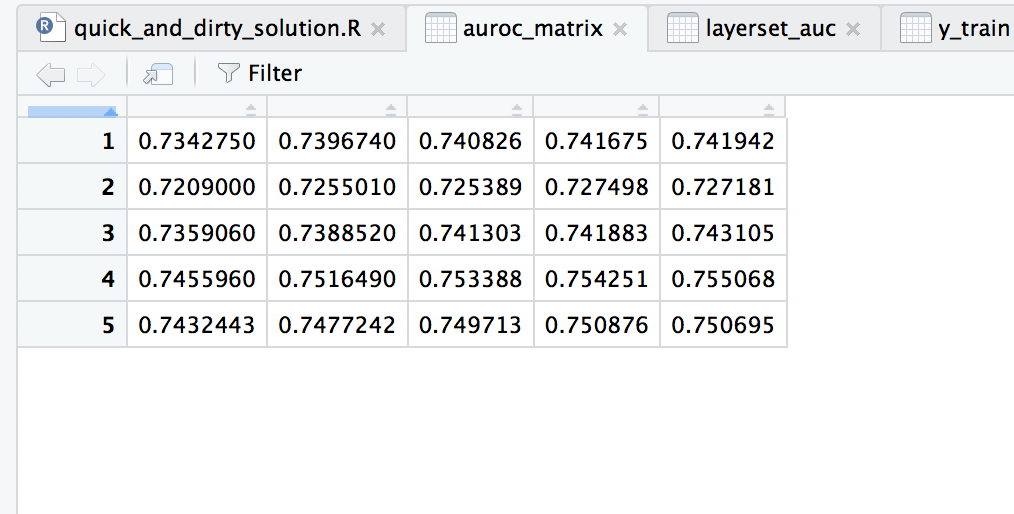


Figure 1: The Auroc Matrix for the 5-Fold Cross-Validation of the nTree Set

Above is the resulting Auroc matrix from my computations. The values are relatively similar for each coulumn (possible ntree value), but ntree=500 gives the highest average Auroc value. So I set my ntree parameter to 500. Using this ntree\* value, I train and predict on the whole training set. Below are my resulting ROC curve and AUROC value (0.7507).

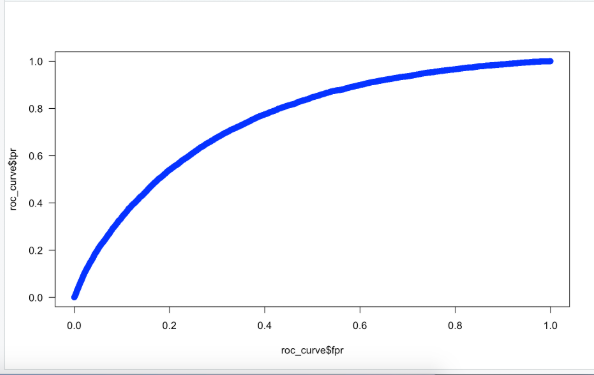


Figure 2: The ROC curve graph for the training dataset

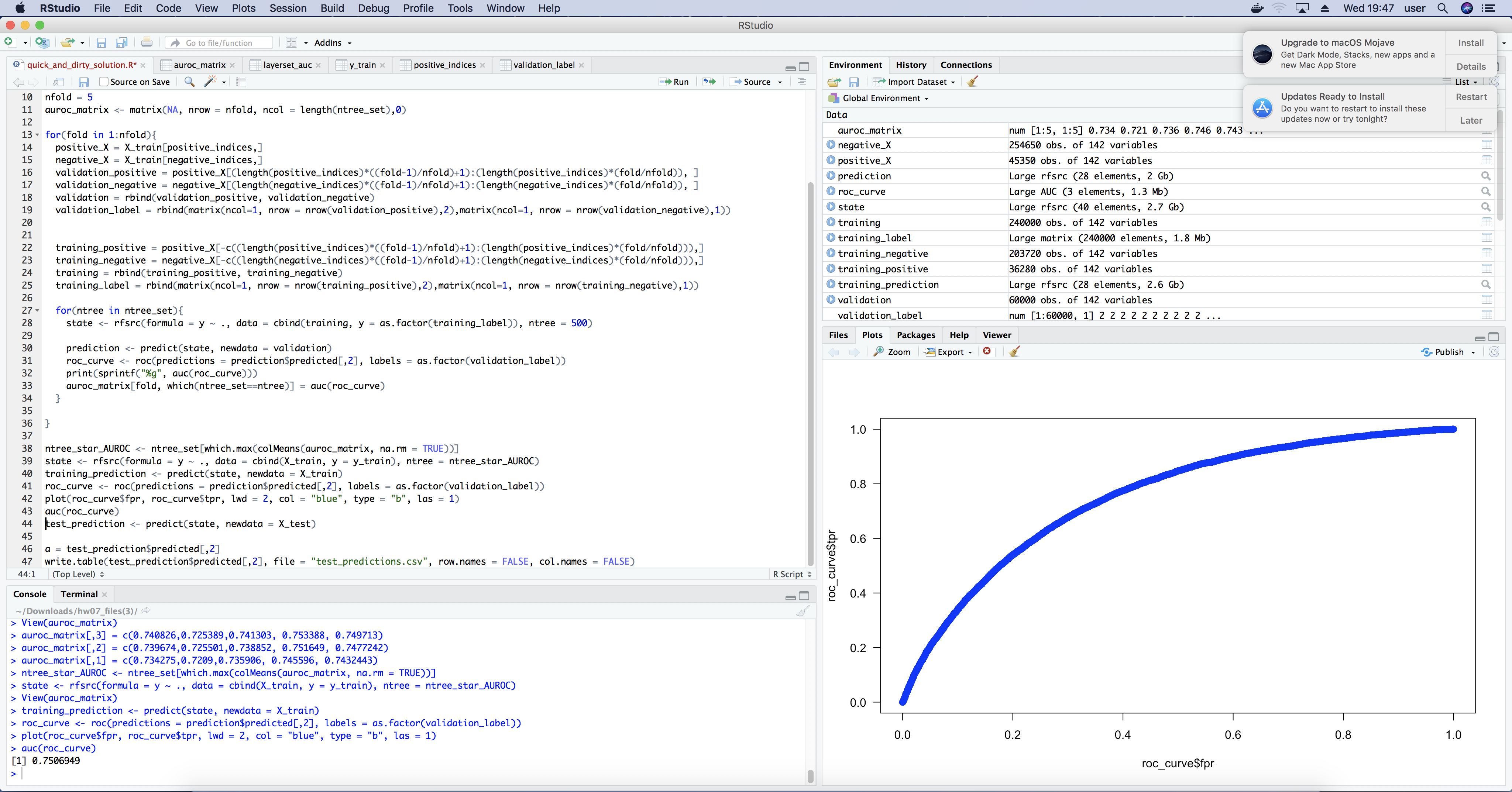


Figure 3: Console Output for the Area Under the ROC curve of the training dataset

I think I achieve a satisfactory result and with the state I receive from my training, I can predict the probabilities for the actual test data. The prediction gives me a list. From this list, I retrieve the probabilities with test\_prediction$predicted[,2] and write the results to a csv file.