CS699 A1 – Spring 2025

Homework 6

Due: 3/24

# For Problem 1 and Problem 2, you must use the method that we discussed in the class.

**Problem 1 (10 points).** This question is about the AdaBoost algorithm that we discussed in the class.

The following table shows a test result for one model:

|  |  |  |  |
| --- | --- | --- | --- |
| TID | Weight | Act | Pred |
| 1 | 0.1 | P | P |
| 2 | 0.1 | N | P |
| 3 | 0.1 | N | N |
| 4 | 0.1 | N | N |
| 5 | 0.1 | P | P |
| 6 | 0.1 | N | P |
| 7 | 0.1 | P | P |
| 8 | 0.1 | P | P |
| 9 | 0.1 | N | P |
| 10 | 0.1 | P | P |

1. Calculate the error of this model.
2. Calculate the normalized, updated weights.

**Problem 2 (10 points).** This problem is also about the AdaBoost algorithm that we discussed in the class.

Suppose that the algorithm built 5 models and each of the 5 models classified an unseen object. The following table shows the errors of the five models and their predictions.

|  |  |  |
| --- | --- | --- |
| Model | err(Mi) | Pred |
| 1 | 0.17 | N |
| 2 | 0.05 | P |
| 3 | 0.13 | N |
| 4 | 0.2 | N |
| 5 | 0.08 | P |

Determine the final classification (the classification of the composite model).

**Problem 3 (20 points).** We discussed in the class two types of ensemble methods – bagging and boosting. There is another type of ensemble method, *stacking*. There are different packages in R which we can use to build a stacked model. For this assignment, you are required to use the *caret* package. If necessary, you must study and learn how to use the stacking method implemented in the *caret* package.

1. Study the stacking ensemble method and write a description of the method. You should not copy from a documentation, a web page, or a response from an AI tool (such as ChatGPT). You must write the description in your own words. Your description must be at least two-page long and must be written using font size 12pt and single spaced.

Ensemble learning is all about combining multiple models to get better predictions. There are different ways to do this, like bagging and boosting, but stacking takes a slightly different approach. Instead of just averaging or weighing models in a simple way, stacking uses another model, called a meta-learner, to figure out the best way to mix their predictions. This method is effective because it lets different models complement each other, which usually leads to better accuracy.

The idea behind stacking is straightforward. First, multiple base models, like decision trees, support vector machines, or neural networks, are trained on the same dataset. Instead of making a final decision based on these models directly, their predictions are used as input features for another model, the meta-learner. This meta-learner is trained to make the best possible final prediction by learning from the strengths and weaknesses of the base models. Since the meta-learner is trained on the outputs of the base models, it figures out how to balance their predictions in a way that maximizes overall performance.

To implement stacking in R, the caret package is a great option. The process starts with loading the necessary libraries, like caret, randomForest, and e1071. After that, we prepare the data, usually by splitting it into training and testing sets using createDataPartition. Once the data is ready, we train the base models using the train function. Each base model is trained separately using different methods, like random forests (rf) and support vector machines (svmRadial). After training, we use caretStack to combine the base models and fit a meta-learner, which is often a logistic regression model. Once trained, this stacked model can be used for predictions, and we can evaluate its performance using functions like confusionMatrix.

For example, let’s say we have a dataset and want to predict a target variable. Instead of relying on just one model, we train a random forest model and a support vector machine model separately. After getting their predictions, we pass those predictions as inputs to another model, like logistic regression, which learns how to best combine the results from the random forest and SVM models. This approach usually improves accuracy compared to using just one model.

Stacking has some big advantages. It usually leads to higher accuracy since it combines the strengths of multiple models. It’s also flexible because we can use different types of models and blend them in different ways. Another major benefit is that it makes the model more robust—since stacking leverages diverse predictions, it can help reduce overfitting and make the model perform better on unseen data.

However, stacking has some downsides. One of the biggest challenges is that it takes a lot of computational power. Training multiple models and then a meta-learner takes more time and resources compared to just using a single model. Another issue is tuning—each base model and the meta-learner need to be fine-tuned separately, which can be tricky. Also, since stacking involves multiple layers of models, it can be harder to interpret compared to simpler ensemble methods like bagging and boosting.

Stacking is useful when different models do well on different parts of the dataset, but no single model is perfect across the entire dataset. If multiple models give different but useful predictions, stacking helps figure out the best way to combine them. It’s also great when regular models struggle to capture complex patterns in data. For example, if a dataset has non-linear relationships or interactions that individual models can’t fully understand, stacking can help by combining multiple perspectives. Stacking is also popular in competitions or real-world applications where even small accuracy improvements matter a lot.

Another time stacking is useful when working with high-dimensional data that has different types of features. Some models might be good with numerical data, while others handle categorical data better. By stacking these models together, we can take advantage of each model’s strengths and create a more effective prediction system.

In summary, stacking is a powerful ensemble method that can significantly improve model performance by intelligently combining multiple predictors. The caret package in R makes it relatively easy to implement, although it does take some extra tuning and computational power. If used the right way, stacking can be a great way to boost accuracy, especially when working with complex datasets where no single model performs well enough on its own. Even though it requires more effort, the accuracy improvements make stacking a valuable technique in machine learning.

1. Write a R code, *hw6.R*, which builds and tests a stacked model (or a stacked metamodel) following the instruction below:
   * Use the *drug\_consumption\_cannabis.csv* dataset.
   * Split the dataset into a training set and a test set.
   * Build *naïve bayes*, *nnet*, *knn*, and *svmRadial* models from the training set and test them on the test set. Make sure that you build and test each model separately and also make sure that you do parameter tuning.
   * Collect the confusion matrix, sensitivity, and the specificity of each model.
   * Build a stacked model using the above four algorithms as base algorithms and using *glm*

as the meta learning algorithm.

* + Test the metamodel on the test set and collect the confusion matrix, sensitivity, and the specificity.
  + In your homework submission file, include all five confusion matrices and the following performance comparison table:

|  |  |  |
| --- | --- | --- |
| Model | Sensitivity | Specificity |
| rpart | 0.7983871 | 0.8537549 |
| nnet | 0.7096774 | 0.8537549 |
| knn | 0.6532258 | 0.8814229 |
| svmRadial | 0.6935484 | 0.9011858 |
| metamodel | 0.7258065 | 0.9011858 |

Is the performance of the metamodel better than the performances of individual models?

It is not completely better. The SVM Model performs better than the MetaModel.

# Submission:

Include answers to Problem 1, Problem 2, and Problem 3 in *hw6.docx* or *hw6.pdf*. Combine this file and *hw6.R* into a single archive file and name it *LastName\_FirstName*\_*HW6.EXT*. Here, “*EXT*” is an appropriate file extension (e.g., *zip* or *rar*).