P2: BUILDING AN INTERVENTION SYSTEM

ANDY PEREZ

1. Classification vs Regression

This is a classification problem since we are identifying which of two categories our students belong to. They either graduate, or fail to graduate. Although internal to the project there may exist an algorithm that assigns a numerical value related to the likelihood of a student failing, the aim of the project, ultimately, is to spit out whether a student needs or does not need help in order to successfully finish high school.

2. Exploring the Data

Can you find out the following facts about the dataset?

- Total number of students

  395
- Number of students who passed

  265
- Number of students who failed

  130
- Graduation rate of the class (%)

  67.09%
- Number of features (excluding the label/target column)

  30

3. Preparing the Data

Execute the following steps to prepare the data for modeling, training, and testing:

✓ Identify feature and target columns
✓ Preprocess feature colums
✓ Split data into training and test sets

4. Training and Evaluating Models

4.1. **AdaBoost.**

- What are the general applications of this model? What are its strengths and weaknesses?

  AdaBoost is a very versatile, general-purpose model that can be adapted for both classification and regression. In our case we use it with small decision trees, so it inherits the advantages of trees. These advantages include invariance under strictly monotone transformations of the input variables and robustness against irrelevant input variables. In addition, AdaBoost is, unlike the decision trees it uses as weak learners, generally not prone to overfitting; it generalizes well. However, it can give bad results in the presesence of noise, and can sometimes be sensitive to outliers, since the algorithm focuses on difficult instances. In general, it's rarely a bad idea to try out AdaBoost with a proper choice of weak learner.

- Given what you know about the data so far, why did you choose this model to apply?

  As said earlier, it's generally a pretty safe model to use, especially since we're using small decision trees as our weak learners. I know there will be many variables that have almost no final effect on whether a student graduates (we aren't given an optimized list of things that we know are really important), so having our model do feature selection by itself is quite handy. There's a grand total of 30 features, so the model selecting the important ones is, well, important. The inputs are also mixed; some being continuous and others being categorigical. Trees handle this type of data well, and boosting lets us escape the problems with variance and overfitting. Noise may be an issue, but it's not something I suspect there will be much of, nor something I'm entirely sure how to measure without playing around with the data using some models.
- Time consumption and F1 score table (averaged over four runs)

	Training set size		
	100	200	300
Training time (secs)	0.087	0.088	0.098
Prediction time (secs)	0.005	0.008	0.008
F1 score for training set	0.964	0.867	0.848
F1 score for test set	0.716	0.785	0.803

4.2. **Gradient Tree Bosting.**

- What are the general applications of this model? What are its strengths and weaknesses?

  As was the case with AdaBoost with trees, this model inhereits the benefits of decision trees, namely the invariance under transformations and ability to deal with irrelevant variables. As is the case with AdaBoost, it tends to generalize well. It can be sensitive to noise and outliers, but since we can choose our loss function, in the case of noise we can reduce it's impact through the choice of a robust loss function. It generally is expected to give more accurate predictions than AdaBoost. It's a very adaptable model; it has many parameters to play with. It can have a problem with overfitting, but as with many of the other problems this model has, proper choice of parameters can go a long way towards ameliorating it.
- Given what you know about the data so far, why did you choose this model to apply?

  The same reasons given for AdaBoost are applicable here, many of them which are the result of just using any algorithm that boosts on trees. From what I've read though, this model oftentimes performs better, and there are many parameters I can tweak to cater the model to the data. It may be more computationally intensive, but we can't be sure until we try.

- Time consumption and F1 score table (averaged over four runs)

	Training set size		
	100	200	300
Training time (secs)	0.066	0.101	0.120
Prediction time (secs)	0.000	0.001	0.001
F1 score for training set	1.000	0.992	0.971
F1 score for test set	0.786	0.749	0.804

4.3. Random Forests.

- What are the general applications of this model? What are its strengths and weaknesses?

  It is yet another ensemble method with trees, so we have the case that it is invariant under strictly monotone transformations and resistant to the existence of irreleveant variables. The generalization error for the model converges as we run it longer instead of increasing/diverging, hence it does not have an overfitting problem. It's also resistant to noise in the output variable, moreso than the other models described here. It's quite fast, but it may have variance problems, and the convergence may be slower than in the other models.
- Given what you know about the data so far, why did you choose this model to apply?

  The reasons described in the section on AdaBoost are appropriate here as well. This model is more resistant to noise in the output variable, which is an advantage over the other two. It also may have a speed advantage without sacrificing too much predicative power.
- Time consumption and F1 score table (averaged over four runs)

	Training set size		
	100	200	300
Training time (secs)	0.019	0.023	0.021
Prediction time (secs)	0.001	0.001	0.002
F1 score for training set	0.989	0.994	0.955
F1 score for test set	0.700	0.759	0.761

5. Choosing the Best Model

- Based on the experiments you performed earlier, in 2-3 paragraphs explain to the board of supervisors what single model you choose as the best model. Which model has the best test F1 score and time efficiency? Which model is generally the most appropriate based on the available data, limited resources, cost, and performance? Please directly compare and contrast the numerical values recored to make your case.

The final model chosen, after much deliberation, to most accurately predict student success was Gradient Tree Boosting. Out of all the other models considered, it has the least reliance on data size. F1 score is a measure of predicative ablity (a greater score indicates better predictions), and on just a data set of 100 students, gradient tree boosting (henceforth GTB) outperformed all the other models, with an F1 score of .786 compared to .700 and .716 for the others.

When it came to running time for the models, GTB had the quickest predictions, done in fractions of a second. As for how long it took to train these models, while the random forests model was quicker to train, it was much less reliable than the other models, reaching an F1 score of only .761. GTB slows down more notably for larger data sets, but in our case they are not large enough to make the model unappealing.

- In 1-3 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you choose a decision tree or support vector machine, how does it learn to make a prediction).

Our model is based on a simpler model called a Decision Tree, or DT. DTs are made by examining the data, and at every stage seeing which split in some input variable that best divides it into to categories, then repeating the process themselves on the new categories created. For example, in our data, it turns out that whether a child has failed a class or failed one or more classes is the best way to divide the set of students in two by a simple split so that we get two relatively homogenous sets of students. This isn't surprising, as students who have difficulty passing classes typically have difficulty graduating; and those who are passing all their classes do tend to have little difficulty meeting graduation requirements. Out of those who do pass all their classes, we can further divide them into those who get extra instruction or not; For those who have failed at least one class we can divide them by those with 1 or less absences and those with more, and so forth, predicting whether a student passes more accurately as we take more factors into consideration.

The model we are actually using, Gradient Tree Boosting, starts by taking the average proability of a student failing to graduate (.3291or 32.91% in this case) as it's base model. In other words, it predicts everyone will pass, because more people pass than fail. This is obviously not a very useful model; we make this model useful by then finding the small Decision Tree that can be combined with this current model that gets it closest to predicting the correct student outcomes, and reducing the contribution of it by a factor before combining it with the original one. We keep going on in this fashion, creating a sum of Decision Trees that becomes more and more accurate as more trees are added to the sum, until we arrive at a sufficient final model.

In order to make predictions on a given set of students all that needs to be done is to run their stats through the Decision Trees, and sum them to arrive at a final prediction of whether they're expected to pass or fail.

✓ Fine-tune the model. Use gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.

- What is the model's final F1 score?

  The model's final F1 score is 0.823