P2: BUILDING AN INTERVENTION SYSTEM

ANDY PEREZ

1. Classification vs Regression

This is a classification problem since we are identifying what categories our students belong to. They are either at risk to fail, in which case they need assistance, or they are not at risk to fail, in which case they do not need assistance. 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 graduate.

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 (the last feature in student_data is whether the student passed, the label column)

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?

  AdaBoost's outlier problem is mitigated since we are doing classification involving two common categories. 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.
- Time consumption and F1 score table

	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 irrelevant variables. As is the case with AdaBoost, it tends go 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 modify it to reduce issues. It generally is expected to give more accurate predictions than AdaBoost. In general, 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.
- 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 oftentime preforms better, and there are many parameters I can tweak to adapt it to any features of the data that appear after use. It may be more computationally intensive, but we can't be sure until we try.

- Time consumption and F1 score table

	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?
- Given what you know about the data so far, why did you choose this model to apply?
- Time consumption and F1 score table

	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