1. Classification vs Regression

Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?

This is a classification problem because the students are being grouped into those who do need early intervention and those who do not. The output is discrete with no in-betweens.

2. Exploring the Data

Can you find out the following facts about the dataset?

- Total number of students
    - 395 students, each in a row with a header row
- Number of students who passed
    - 265 students have “yes” as label
- Number of students who failed
    - 130 students have “no” as label
- Graduation rate of the class (%)
    - 67.09% of the students passed
- Number of features
    - 30 with the last column being label

Use the code block provided in the template to compute these values.

3. Preparing the Data

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

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

Starter code snippets for these steps have been provided in the template.

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem. For each model:

1:

- What are the general applications of this model? What are its strengths and weaknesses?
- Gradient Tree Boosting or Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions. GBRT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems. Gradient Tree Boosting models are used in a variety of areas including Web search ranking and ecology.
- The advantages of GBRT are:
    - Natural handling of data of mixed type (= heterogeneous features)
    - Predictive power
    - Robustness to outliers in output space (via robust loss functions)
- The disadvantages of GBRT are:
    - Scalability, due to the sequential nature of boosting it can hardly be parallelized
- From scikit-learn Gradient Tree Boosting documentation: http://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting

- Given what you know about the data so far, why did you choose this model to apply?
- I wanted to use ensemble methods with boosting because there are 30 features that should have different weights applied to them. Because this method is a good off the shelf procedure, I wanted to start off with this to get a benchmark of results.

- Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

- Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

GBRT	Training set size		
	100	200	300
Training time (secs)	0.057	0.075	0.095
Prediction time (secs)	0.000	0.001	0.001
F1 score for training set	1.0	0.982206405694	0.968824940048
F1 score for test set	0.77037037037	0.805555555556	0.817518248175

Note: You need to produce 3 such tables - one for each model.

2:

- What are the general applications of this model? What are its strengths and weaknesses?
- Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. For example in the case of spam filtering the ordering of words is not considered.
- Naive Bayes methods require a small amount of training data to estimate the necessary parameters, and they can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.
- On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator.
- From scikit-learn Naive Bayes documentation: http://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes

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

- Now I that i have a baseline for model performance, I wanted to try Naive Bayes to see how much the assumption of dependence between all 30 features holds true. From the looks of it most of the features probably don't affect the target much, but a couple features stands out like if the student has failed classes before. I've decided to use multinomial naive Bayes because the data contains many different kinds of data, including continuous and discrete data.
- result table:

MultinomialNB	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	0.789115646259	0.814285714286	0.798143851508
F1 score for test set	0.762589928058	0.763358778626	0.787878787879

3:

- What are the general applications of this model? What are its strengths and weaknesses?
- Decision Trees are usually used to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Advantages of decision trees are simple to understand and to interpret, requires little data preparation, among others.
- Disadvantages include prone to overfitting, small variations in the data might result in a completely different tree being generated, and by itself cannot guarantee a globally optimal tree.
- From scikit-learn Decision Trees documentation: http://scikit-learn.org/stable/modules/tree.html

- Given what you know about the data so far, why did you choose this model to apply?
- Looking at the list of features I suspect that certain features will influence the target much more than some others, so I wanted to see if decision trees can use that to its advantage and come up with a good model.
- result table:

DecisionTreeClassifier	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.002
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.754098360656	0.684210526316	0.734375

5. Choosing the Best Model

Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?

I chose gradient boosting classifier because it had consistent the highest score out of the box for all training sizes, ranging from 0.77 to 0.81, whereas multinomial naive Bayes hovers from 0.76 to 0.78 and decision tree from 0.68 to 0.75. It also has a consistent score as training size decreased. The time it takes is higher than the other two but in this scenario it is acceptable at less than a tenth of a second for 300 training samples. The model that is most appropriate would be the one that balances all of the constraints with desired output. If the amount of available data increases in the future then a model that's less robust to overfitting but higher performing can be used. If the amount of allocated resources decreases then a faster model can be used.

In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a decision tree or support vector machine, how does it make a prediction).

Ensemble methods combine predictions of many estimators to try to make a better prediction. It is similar to a voting process in the sense that many aspects of the data can be taken into consideration. Boosting in this case will focus on the ones it got wrong to try to get them right within the specified number of

iterations. This way the model would have a good sense of how much does each feature weight into the final outcome, and have enough opportunities to get the non-obvious ones right.

This is similar to decision trees in that they both go from small subsets of the data to find a solution that will work with the whole data. Decision trees start by picking the best attribute that will split the data roughly in half, and then go into the ones that have the most amount of information gain, and keep splitting the data until it gets the right answer. Ensemble learning picks small subset of data, derive rules from those, and move on to another set of data. Afterwards it will use average or some other methods to combine those rules to formulate a more complex rule that will work with the whole dataset.

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 0.0.834532374101 at 150 estimators and a max depth of 1.