1. Classification vs. Regression

   - Given that the goal is to identify students who might need early intervention, this is a classification problem because we are mapping our input (the student data) to discrete labels (needs early intervention or doesn't need early intervention) as opposed to a continuous output (as in regression).

2. Exploring the Data

   - There are 395 total students. 265 students passed. 130 students failed. The graduation rate of the class is 67.09%. There are 30 features.

3. Training and Evaluating Models

   - Support Vector Machine

     The general applications of a support vector machine are image classification and text/hypertext catogorization. In general, they are useful when there are fewer features with respect to samples.

     It is particularly strong in high dimensional spaces. Furthermore, it is a versatile algorithm – i.e., a *kernel* can be described for the decision function. Lastly, SVMs tend to be memory efficient because they use a subset of training points in the decision function.

     Support vector machine's main weakness is they are inefficient to train (this is largely dependent upon the kernel and regularization parameter). Furthermore, when the number of features is much greater than the number of samples, it yields a poor performance. Also, SVMs use a five-fold cross-validation to provide probability estimates, which is expensive.

     Given what I know about the data so far, I chose to apply a support vector machine model because the data contains more samples than features, thus playing toward SVM's strength.

	Training set size		
	100	200	300
Training time (secs)	0.002	0.004	0.011
Prediction time (secs)	0.001	0.003	0.007
F1 score for training set	0.8591	0.8693	0.8692
F1 score for test set	0.7839	0.7755	0.7586

   - Decision Tree Classifier

     The general applications of decision trees are problems with many different features because decision trees perform an implicit feature selection where the highest decisions made are biased toward what it believes to be the most important features.

     Decision trees are known for being particularly fast, especially with large datasets. Furthermore, decision trees are known to be robust. Also, decision trees implicitly perform variable screening or feature selection. Lastly, decision trees require low effort from users with respect to data preparation, and decision trees are simple to understand and interpret.

     Decision trees' main weakness is the tendency to create over-complex trees that don't generalize well – i.e. overfitting. Furthermore, decision trees tend to have high variance and low prediction accuracy.

     Given what I know about the data so far, I chose to apply a decision tree model for speed and to see how the decision tree would fare given the many features from which to classify.

   - Random Forest Classifier

	Training set size		
	100	200	300
Training time (secs)	0.004	0.003	0.003
Prediction time (secs)	0.001	0.001	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.7069	0.7031	0.7350

The general applications of the random forest classifier are areas with large datasets and a large number of features.

The advantages of random forests are they readily handle larger numbers of predictors, they have fewer parameters, they are fast to train, and they don't require cross validation (because they generate an internal unbiased estimate of the generalization error, or test error, as the forest building progresses). Similar to decision trees, random forests require low effort in terms of data preparation.

Random forests' main disadvantage is since more accurate ensembles require more trees, creating predictions once the model is trained causes the model to become slower. Furthermore, random forests are far less comprehensible compared to decision trees. They also are prone to overfitting for noisier datasets (although less so than decision trees).

Given what I know about the data so far, I chose to apply a random forest model because I wanted to expand on the ability of the decision tree.

	Training set size		
	100	200	300
Training time (secs)	0.008	0.008	0.015
Prediction time (secs)	0.001	0.003	0.004
F1 score for training set	1.0	0.9962	0.9903
F1 score for test set	0.7333	0.7826	0.7556

- AdaBoost Classifier

  The general applications of the Ada Boost Classifier are problems that consist of many layers of classifiers – e.g., facial detection using Viola-Jones.

  Its strengths are it requires much less tweaking of parameters than other classification algorithms. AdaBoost works by using an iterative process across a pre-defined set of "weak" classifiers, choosing the best at that round of boosting.

  Its weaknesses are sensitivity to noisy data and outliers. Specifically, AdaBoost tends to have a similar overfitting problem as with Random Forests and Decision Trees.

  Given what I know about the data so far, I chose to apply an AdaBoost model because I wanted to also compare how a boosting model would fare with a base of a decision tree.

	Training set size		
	100	200	300
Training time (secs)	0.044	0.055	0.059
Prediction time (secs)	0.005	0.007	0.007
F1 score for training set	0.9538	0.8865	0.8662
F1 score for test set	0.72	0.7971	0.7794

4. Choosing the Best Model

- Based on the experiments detailed above, I chose a support vector machine as the best model. Although the decision tree had the best F1 score for the training set (1.0), the support vector machine and AdaBoost classifiers had the best F1 scores for the test sets (.7727 and 0.7655, respectfully, averaged across training set sizes of 100, 200, and 300). As for time efficiency, the support vector machine was much faster than random forests and AdaBoost (0.0057 for the SVM vs 0.0103 for random forests and 0.0527 for AdaBoost, averaged across training set sizes of 100, 200, and 300), but not as fast as the decision tree (0.0033, averaged). For the purposes of this problem, the support vector machine was negligibly slower than the decision tree (by about 0.002 seconds), but had a much higher F1 score on the test set (0.7727 for the SVM vs 0.715 for the DT). This tends to indicate that the decision tree may have experienced overfitting in the training set, thus contributing to an F1 score of 1.0 for the training set, but a lower score for the test set.

  Based on the available data, the support vector machine appears to be the most appropriate model because it consistently performs the best or second best out of each classifier. Based on the limited resources, cost, and performance, either the decision tree or support vector machine would be the best model because both perfomed much faster than AdaBoost or random forest. Therefore, choosing the support vector machine is a solid tradeoff between speed and efficiency because it is both fast and efficient.

  Support vector machines work by focusing only on the data points that are the most difficult to tell apart because the idea is that if a SVM is good at the most challenging comparisons (points that are closest to each other), then the classifier will be even better at easy comparisons (points that are furthest from each other). More specifically, support vector machines find the best separating line (or plane) that "splits" the data into the specified labels, or *classifications*. In our case, the labels are "pass" and "fail".

  First, the support vector machine starts by searching for the closest points (where each point has an associated "attendance", "age", "sex", etc.) to one another, or *support vectors*. Once it has these, it draws a line (or plane) connecting them, and then draws the best separating line (or plane) that is perpendicular to this connecting line. Therefore, when it comes across a new sample – in our case, a student – the SVM is able to make a prediction due to the previously made line and the fact that spillover is unlikely.

  Using 75% of the entire dataset as the training set and optimizing across gamma, C, and the kernel, the model's final F1 score was 0.8434 and the F1 score for the test set was 0.7862. The parameters grid search found to be best were an rbf kernel, gamma set to 0.001, and C set to 10.0.