P1: Student Intervention System

1) Classification vs Regression

The problem of identifying students that might need early intervention (students that are likely to fail), is a classification problem. The output (the target: “passed”) is binary “yes” or “no”. The problem is also discreet, most of the features are binary or trinary, the highest numeric feature range is 0 to 93, while a relatively large number compared to binary is still discreet.

2) Exploring the Data

- 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

- Identified the target column as “passed”, the rest are part of the feature set
- Decomposed some more complex features into binary set
- Split the data into training (300 items) and testing (95 items) sets

4) Training and Evaluating Models

Gaussian Naive Bayes

We selected Gaussian Naive Bayes as the first model. Naive Bayes works very well as a classifier, it work well in many applications, particularly known for document classification as well as email spam filtering.

Naive Bayes was selected for student intervention model because we are dealing with a classification problem with relatively small data, and this model can achieve good results even with a limited data set (in this instance a dataset of 395 examples) and can even deal with some missing attributes. Even though Naive Bayes assumes there is no interdependence between features it still works with dataset features that do have strong dependencies as those dependencies cancel each other out. This model works better with potentially overlapping data where classes might overlap. In addition Naive Bayes scales linearly O(n) with data as we are working with limited resources and want to keep processing costs to a reasonable minimum. The initial assumption is that some of the features in our dataset might be interdependent and do overlap in some way.

	Training set size		
	100	200	300
Training time (sec)	0.002	0.001	0.001
Prediction time (sec)	0.001	0.000	0.001
F1 score for training set	0.8467	0.8406	0.8038
F1 score for test set	0.8029	0.7244	0.7634

Support Vector Machines (SVM)

SCM are used for classification and regression analysis, applications in many fields including bioinformatics, text, image recognition.

We selected SVM because it is generally known to be one of the top classification models and can be very effective in high dimensional space (we have 30+ features in the dataset), if there is a clear margin of separation in our data we can get a very accurate prediction.

	Training set size		
	100	200	300
Training time (sec)	0.002	0.005	0.008
Prediction time (sec)	0.001	0.003	0.005
F1 score for training set	0.8777	0.8679	0.8761
F1 score for test set	0.7746	0.7815	0.7838

Classification Decision Tree

Decision tree learning is commonly used in data mining where based on input variables the target value is predicted. Decision trees are also used in decision analysis (formal study of decisions).

We selected this back up model for its simplicity and ability to work well with classification problems as well as having fast performance, decision tree learning scale logarithmically with data

O(log n) as we are taking limited resources into consideration. In addition to boolean data Decision Trees can handle numerical and categorical data (we have all three in our dataset).

	Training set size		
	100	200	300
Training time (sec)	0.001	0.002	0.002
Prediction time (sec)	0.000	0.000	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.6207	0.7556	0.6830

5) Choosing the Best Model

The model selected out of the 3 models is the classification decision tree. Decision tree F1 score on test dataset of 300 training items was 0.6830; however we can see that the F1 score for test dataset of 200 training items was 0.7556 and by looking at the F1 score for training set for all sizes one can see that the score is pefect 1.0 this means the training dataset was perfectly fitted to the data and while we'd love to see a score of 1.0 for the test dataset, the same score for the training dataset (while test score is much lower) means the model is badly overfitted. But overfitting can be mediated using model fine-tuning (we will do this in the last section).

Time efficiency for the decision tree is 0.002 for training and < 0.000 for prediction vs much higher for SVM 0.008 and 0.005 respectively, 4 and >5 times longer than decision tree times. While SVM has a F1 score of 0.7838 or 0.0282 more than decision tree F1 score, the added accuracy is not justifiable given the increased processing time (in addition SVM processing time will grow n^2 vs log n for decision tree, that is a much faster growth rate, the processing time compared to dataset size will grow much faster for SVM). Decision tree model has the best test F1 score and time efficiency balance.

Naive Bayes model performed well with an F1 score of 0.7634 and 0.001 training time and 0.001 prediction time, a score 0.0078 larger than the decision tree F1 score. The decision tree processing time is however better even though it's training time is double that of the Naive Bayes, the training will only need to be done one time, the processing time that counts in this instance is the prediction time, as we will be running the model for each of the existing and new students (perhaps multiple times a year for each student, to catch troublesome developments early). In addition Naive Bayes is linear (will grow n to the dataset) where decision tree is logarithmic (will grow log n), linear grow will have longer processing times.

In addition if we can attach weights to Naive Bayes features we might be able to fine-tune the model to give better results, but unfortunately we do not know if particular features are more dominant than others or what weights to attach (in the future we might be able to do this given some study into feature importance). On the other hand we can see clearly that decision tree is badly overfitted (from the fact that testing dataset score is 1.0), but still delivered similar results to other models. Decision tree looks to be a low hanging fruit for fine-tuning, we should be able to increase the decision tree F1 score above of what we have seen from the other models while having much better time efficiency.

Given available data, limited resources, cost and performance considerations decision tree model is the most appropriate of the three.

In simple terms the decision tree works by splitting the dataset in two according to a particular variable (feature) and then splitting it again according to another variable until each split only has one type (class) of data (in this instance until we know that the student “passed” or not).

As an example let's say that the student passes or does not pass depends only on, if the student has an internet connection (and the internet connection was what made the student pass), the dataset in this case will be split in two by the decision tree model, one side will be all the students that have internet connections and therefore passed and on the other side all the students that don't have internet connections and therefore did not pass. New students will travel down this decision tree until they reach a prediction of passing or not passing.

- After some fine-tuning the final decision tree F1 score is 0.8