P2 Student Intervention System
Draft Submission for Feedback

1. Classification vs Regression

This problem is a classification problem as the output is a discreet value, in this case whether the student passes (label is “yes”) or they fail (label is “no”).

In comparison, regression problems relate to continuous values of data, such as length or temperature.

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 : 135
- Graduation rate of the class (%) : 67%
- Number of features (excluding the label/target column) : 30

3. Preparing the Data

See submitted iPython notebook

4. Training and Evaluating Models

Model 1: Decision Tree

Decision trees can work by determining a set of decisions that can be made to predict the likely outcome of data. For example, if the biggest differentiator between passing and failing is the number of absences they have, then the first decision about could be based on that feature, and so on through the remaining features.

I chose this model to understand whether there are a number of features that are indicative of a student passing or failing, as if this was the case a decision

tree should be able to create a good model based on these features. Decision trees work very well in supervised classification problems like this one.

Training set size ->	100	200	300
Training time (secs)	0.00089	0.00120	0.00223
Prediction time (secs)	0.00027	0.00013	0.00014
F1 score (training set)	1.0	1.0	1.0
F1 score (test set)	0.7	0.733	0.693

Model 2: K-Nearest Neighbour (KNN)

KNN models are usually very good at predicting based on similar data, so for example trying to predict whether a particular student will pass is based on other students with similar features, such as their parents' education or how much time they study. The "K" is a value that determines how many neighbours are used to predict. This value can vary depending on the dataset. An advantage of this data set is that training should effectively be very easy as the data is simply stored, but the disadvantage is determining which are the nearest neighbours at prediction time. This model also stores all of the data

I picked this model as I my intuition suggests that the likelihood of a pass for a particular student is likely to be similar to students with similar features. In this test, I have used K=3 (so, use the 3 nearest neighbours)

Training set size ->	100	200	300
Training time (secs)	0.00061	0.00046	0.00056
Prediction time (secs)	0.00134	0.00167	0.00196
F1 score (training set)	0.859	0.88	0.888
F1 score (test set)	0.739	0.791	0.822

Model 3: Gaussian Naive Bayes

They can be good to determine linear models that fits parts of the data, but are able to merge these different linear models to provide a much more accurate classification algorithm. Another advantage is that the data is not stored and only the model is kept. However, a potential weakness of this model is the number of linear algebra calculations it needs to make in order to determine an optimal model for prediction.

I chose this model as there are a large number of features and I wanted to understand whether a number of linear regression models, when merged together, would be a good prediction model. This is the model that was used in the house price example (albeit in a regression problem).

Training set size ->	100	200	300
Training time (secs)	0.00048	0.00053	0.00073
Prediction time (secs)	0.00023	0.00023	0.00027
F1 score (training set)	0.827	0.78	0.804
F1 score (test set)	0.693	0.758	0.765

5. Choosing the Best Model

From testing different models, the best f1 score (which is a measure of a model's ability to predict correctly) was achieved using K-Nearest Neighbour (KNN). This achieved a f1 score of .822 vs .765 for the next best model. In terms of fit to the available data, K-Nearest neighbour works best with a small training set. Based on just 100 training examples, it's f1 score for the test set is 0.739 compared with 0.7 for DecisionTreeClassifier and 0.693 for GaussianNB. From the metrics, it is obvious that the Decision Tree Classifier suffers from overfitting to the training data which larger training data sets.

In terms of resource utilisation, Gaussian Naives Bayes provides the best CPU profile. Although it training time of .00073 is higher than the other models (0.0056 for KNN and 0.00223 for DecisionTreeClassifier), the most common scenario would not be to train but to determine the pass or fail of a student. In this query-type workload, it's CPU load is 0.00023 compared with 000027 (DecisionTreeClassifier) and 0.00134 (KNN). In addition, both DecisionTreesClassifiers and GaussianNB create a model of the data, which is effectively of an order of magnitude of 1 (it is small), whereas KNN keeps a

copy of the data (so an order of magnitude in relation to the data) so that it can perform calculations at query time.

Based on these metrics, I would propose that we use the Gaussian Naive Bayes model.

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?

Gaussian Naive Bayes works by performing statistical analysis of the data. For example, it tries to determine the probability of an outcome based on the available data. An example could be the probability of passing based on whether the student's Mother works full time. During training, GaussianNB looks at each of the available features and tries to determine the probability for each and their affect on the student's result.

In theory it is possible to determine whether some training examples are more indicative of a generalised model for whether a student passes or fails.