1. Classification vs Regression

Classification.
What we want to predict here is whether students will “pass” or “fail” the class.
So we chose column “passed” as a target label to train and test the model.
This target label (“passed” or “failed”) is not continuous but descrete value.
Problems about predicting descrete values are classification.
Therefore I decided to solve this problem as one of classification problems.

2. Exploring the Data

Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 31
Graduation rate of the class: 67.09%

3. Preparing the Data

Done in in the IPython Notebook.

4. Training and Evaluating Models

I chose these 3 models: Decision Tree, SVM, Gaussian Naive Bayes

Decision Tree:

The problem here is classification, especially binary classification (“yes” or “no”). So, with criterion “entropy” DT learner is supposed to work best because the information gain is caluculated on log base 2.

strengths:

able to handle both numerical and categorical data.
simple to understand and to interpret.
bigger classifier available (e.g. emsamble methods)

weaknesses:

prone to overfit.

SVM:

The dataset here is small (395 data points) and SVM is said to be good at handling small datasets, so I chose this model.

strengths:

Still effective in cases where number of dimensions is greater than the number of samples.

weaknesses:

If the number of features is much greater than the number of samples, the method is likely to give poor performances.
don't work well in very large datasets (because the training time happens to be cubic in the size of the datasets).

Gaussian Naive Bayes:

It is said that this model requires a small amount of training data to estimate the necessary parameters. So on the same reason as the SVM model, I chose this model.

Naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering.

strengths:

can be extremely fast compared to more sophisticated methods.
require a small amount of training data to estimate the necessary parameters.

weaknessess:

a bad estimator, so the probability outputs are not to be taken too seriously.

I've written strengths and weaknesses above acording to the sklearn documentation and UDACITY's lecture videos.

Decision Tree

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.003
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	1.000	1.000	1.000
F1 score for test set	0.732	0.773	0.803

SVM

	Training set size		
	100	200	300
Training time (secs)	0.001	0.004	0.007
Prediction time (secs)	0.001	0.002	0.005
F1 score for training set	0.848	0.887	0.876
F1 score for test set	0.779	0.792	0.784

GaussianNB

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.001	0.001	0.000
F1 score for training set	0.831	0.810	0.804
F1 score for test set	0.672	0.726	0.763

5. Choosing the Best Model

I chose the Decision tree model as the best model.
It has the highest F1 score for entire testing set size (0.803).
It also has the fastest prediction time (0.000 secs).
Although it doesn't have the fastest training time and have the second fasteset training time (0.003 secs), I think the differences between its training time and that of the GaussianNB model (, which has the fastest training time here) is quite small and negligible. Because almost no disadvantage in times and have the highest F1 score for test set, I conclude that the DT model is the best model here.

The final model chosen above (DT model) is supposed to work as follows:

It separates the data points into two subsets and then separates each subsets into two sub-subsets and then separates each sub-subsets into sub-sub-subsets… until each subset contains only one class (only "yes" or "no" in this case).

In this separating process, it determines a rule for each separation. This rule is some kind of question like "Does the student want to take higher education?", "Does the student have an extra educational support?", "Does the student study weekly more than 10 hours?", "Is the number of school absences of the student lessthan 5?".

It determines these rules so that it maximizes certain criterion. This criterion measures how much information is gained through separation. This is caluculated as certain value of the parent subset minus the sum of the same value of the child subsets. This certain value is an impurity value and measures how impure each subset is. For example, a subset which contains 2 "yes" and 2 "no" gets an impurity value 1 (completely impure). On the other hand, a subset which contains only "yes" or only "no" gets an imputiry value 0 (completely pure). For each separation, the DT tree model calcurates the impurity of the parent and the sum of the impurity of the children and measures how much information is gained (an impurity value of the parent minus an impurity value of the children) through the separation, then separates the subset so that the information gain value becomes as big as possible. Though these separating processes, the DT model learns the set of rules that maximize the information gain value.

And using the set of rules learned, DT model predicts whether given students will pass or not. For example, DT model learns that the student “who wants to take higher education” and “who has an extra educational support” and “who studies weekly more than 10 hours” and “whose number of school absences less than 5” passes. Then it predicts that a student who wants to take higher education and has an extra educational support and studies 12 hours weekly and was absent from school 2 times will pass. But it predicts a student who studies 5 hours weekly will fail. It also predicts a student who studies 11 hours weekly but was absent from school 10 times will fail.

The model’s final F1 score is 0.8 for test set.