1. Problem description

This is classification problem. The goal of the task is to map students into 2 categories: pass or not pass. So in other words, we are going to classify students by this discreet parameter (true or false).

2. Exploring the data

Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 30
Graduation rate of the class: 67.09%

3. Preparing the Data

Identify feature and target column: done
Preprocess feature columns: done
Split data into training and test sets:
Training set: 300 samples
Test set: 95 samples

4. Training and evaluating models

Due to the nature of the problems (classification) the following algorithms have been chosen:

1. Decision tree (http://scikit-learn.org/stable/modules/tree.html)

Algorithm complexity in Scikit-leart realization:
$O(n_{features} n_{samples} \log(n_{samples}))$

General application of the model is to create a model that predicts the value of target variable by decision rules from features.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

The method has been chosen due to the simplicity of understanding and applicability to the classification problems.

Results

	Training set size		
	100	200	300
Training time, s	0.002	0.002	0.004
Prediction time, s	0.001	<0.001	0.001
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.772727272727	0.784615384615	0.740157480315

2. Support vector machines (SVM, http://scikit-learn.org/stable/modules/svm.html)

Algorithm complexity: between $O(n_{features} \times n_{samples}^2)$ and $O(n_{features} \times n_{samples}^3)$

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

The method has been chosen due to the effectiveness in high dimension space and classification nature of the problem.

Results

	Training set size		
	100	200	300
Training time, s	0.002	0.004	0.010
Prediction time, s	0.001	0.002	0.002
F1 score for training set	0.815789473684	0.864686468647	0.847965738758
F1 score for test set	0.853503184713	0.851351351351	0.84

3. Stochastic gradient descent (http://scikit-learn.org/stable/modules/svm.html)

Complexity: If X is a matrix of size (n, p) training has a cost of $O(kn\bar{p})$, where k is the number of iterations (epochs) and $\bar{p}$ is the average number of non-zero attributes per sample.

The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

The method has been chosen due to the foreseen lower complexity (training time) of the algorithm and regularization ability.

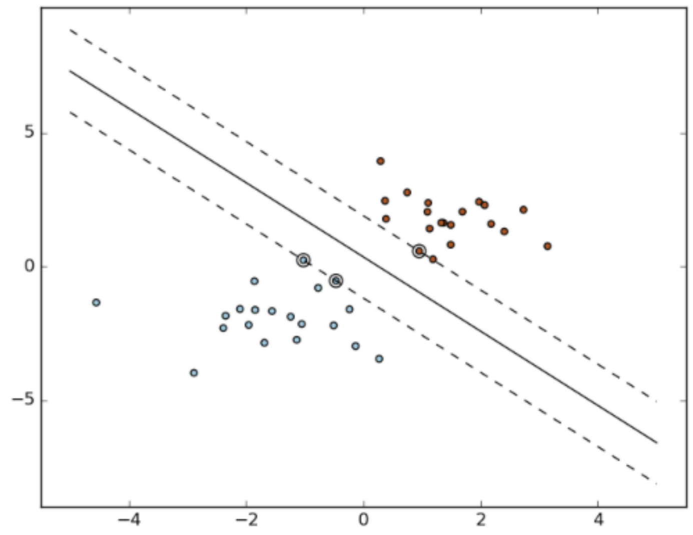
Results

	Training set size		
	100	200	300
Training time, s	0.001	0.001	0.034
Prediction time, s	<0.001	<0.001	<0.001
F1 score for training set	0.748201438849	0.635944700461	0.70737913486
F1 score for test set	0.813333333333	0.623853211009	0.714285714286

5. Choosing the Best Model

Decision tree model has been initially tested. But split of training/testing data set is not optimal for this model. Based on variation of training data set 100,200,250 points, local extreme of F1 score for test data set near 200 points in testing data set could be found. Later 2 other algorithms have been tested for the same initial training/testing data set: Support Vector Machines and Stochastic Gradient Descent. Support Vector Machines reach highest F1 score on testing data set 0.84 (in case of 100 and 200 data points F1 score on the same size testing data is slightly higher, but remaining points should be added to testing data set to use all information and make proper comparison). SGD shows lower F1 score in all cases. For 300 data points in training data set the training time is highest in case of SVM, but it could be decreased without damaging prediction accuracy by reducing training data set size.

SVM algorithm is trained by creating the surfaces in n-dimensional space among points in training data set. It tries to found the equation for the surface on the largest distance from different classes of data points.



For example on the graph above the bold line split data points by 2 classes above and below the line. On the prediction step, the same surface (line in the example above) is using to predict to which class the data point is belong. If the point is above the line, it belong to class 1, in other case – to class 2.

GridSearchCV was used and C for SVM algorithm was varied as parameter (I’ve tried other parameters: cache_size, kernel, max_inter and no increase of F1 score has been achieved).

SVM model's final F1 score for test set: 0.84.