P2 - Building an Intervention System.

Julio Rodrigues (juliocezar.rodrigues@gmail.com) January 2016

1 Classification vs Regression

1. **Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?**

This is a classification problem, since we are trying to classify students into two categories: passed and not passed.

2 Exploring the data

1. Total number of students: 395
2. Number of students who passed: 265
3. Number of students who failed: 130
4. Number of features: 30
5. Graduation rate of the class: 67.09%

3 Preparing the data

(see code)

4 Training and Evaluating Models

1. What are the general applications of this model? What are its strengths and weaknesses?
2. Given what you know about the data so far, why did you choose this model to apply?
3. Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.
4. Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

Chosen learning models:

- AdaBoost

  AdaBoost is one of the most popular boosting algorithms and can be used for different kinds of problems. The literature indicates that it is often able to deliver good results out-of-the-box. Some other advantages of this approach are its speed, its effectiveness (given its simplicity) and also its flexibility, since it can work well on different types of data. One of its caveats is the fact that it can easily overfit on noisy data sets. Moreover, it can underfit if the weak classifiers are too weak.

  AdaBoost was chosen as part of this experiment because it seemed a good fit for the kind and amount of data available and because of the advantages described above. It proved itself as a good model and sometimes outperformed all other tested models. One noticed weakness during the tests was the fact that its performance can vary considerably from one run to the other or even among single cross-validation runs. There were indications of a strong tendency to overfitting, as well, specially when trained on small amounts of data.

- K-Nearest Neighbors

  KNN classification is an instance-based algorithm. It classifies new data points based on its similarity between instances of the training data. Some of the main advantages of KNN are the absence of training (lazy learning) and the fact that it does not make any assumption about a possible underlying function to the problem we are trying to predict. One of its main disadvantages is that it can perform very poorly depending on the distribution/structure of local data (noise, sparseness, etc).

  KNN was used here rather as a reference and as a base comparison to the other parametric models. I also wondered how effective an instance-based approach would be when trying to predict student performance, considering that it is an aspect that can be highly "vicinity-dependent".

- SVM

  SVM seems to be one of the most popular machine learning algorithms. It offers many advantages; it is very versatile, since it can handle both linear and non-linear data (mainly by applying different kernels), it performs well even in high dimensional spaces, and since it uses only a few *support vectors* to make predictions, SVM is quite tolerant to some amount of noise in the data. One of the main disadvantages seems to be the necessity of adjusting several combinations of parameters in order to guarantee its good performance and avoid overfitting.

  SVM was the chosen model especially for its robustness and generally good performance in practice. This model was expected to easily accommodate the number of features in the data and also to keep a stable performance even among varying sizes of the data set.

SVC

	100	200	300
Prediction time (secs)	0.002	0.005	0.010
Training time (secs)	0.003	0.007	0.014
F1 - Test	0.768212	0.797297	0.783784
F1 - training	0.904459	0.890323	0.876068
10-fold cv accuracy	0.63	0.63	0.63
95% CI	+/- 0.12	+/- 0.12	+/- 0.12

KNeighborsClassifier

	100	200	300
Prediction time (secs)	0.002	0.005	0.012
Training time (secs)	0.001	0.002	0.003
F1 - Test	0.772414	0.779412	0.780142
F1 - training	0.88	0.879195	0.880899
10-fold cv accuracy	0.57	0.57	0.57
95% CI	+/- 0.24	+/- 0.24	+/- 0.24

AdaBoostClassifier

	100	200	300
Prediction time (secs)	0.029	0.032	0.034
Training time (secs)	0.448	0.515	0.603
F1 - Test	0.783217	0.771429	0.771429
F1 - training	1	1	1
10-fold cv accuracy	0.68	0.66	0.67
95% CI	+/- 0.22	+/- 0.16	+/- 0.15

5 Choosing the Best Model

- Based on the performed experiments, SVM proved itself as being the appropriate model for the task at hand. In the experiments it often outperformed all other models, but more importantly, it presented stable and robust results across different settings and sizes of the data set (see confidence intervals). Moreover, after performing a grid search with cross-validation (default 3-folds), it presented an even better performance. The training and prediction times for the SVM model are also notably better than the times of the AdaBoostClassifier model, the next best model in comparison. KNN presented F_1 scores comparable to SVM with even better prediction and training times, it had however the poorest performance when considering only the cross-validation tests. Among the three compared models, KNN also presented the highest standard deviation between single cross-validation runs. K-nearest neighbors was not chosen also due to the concern that small variations in the data set could lead to substantially poorer results.

- SVM is an algorithm that learns from already available examples (called *training set*). If we imagine for a moment that we would be using only two characteristics of the students to predict their chances of passing, we could represent our students, i.e. our training-set, as dots on a two-dimensional Cartesian plane (here I would draw it). Provided with these examples, SVM uses them to calculate a line on this plane which separates the dots that represent students which passed from the dots representing students who did not pass. The choice for the best separating line is based on calculations which do not simply find a random separating line, but a line which separates the dots from each other with the largest margin possible (here I would draw the central separating line, the two other equidistant parallel lines on each side and a perpendicular line indicating the distance from the central line to the other two supporting lines). Once our model "learns" where to draw this line,

all the model has to do when confronted with a new input to be predicted is to check on which side of the line this input falls and classify it accordingly.

- The model's final F_1 score after some tuning using grid search is 85% on the test set. (see code output)