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. It is a meta-algorithm, i.e. it combines the predictions of weak learners into a weighted sum to produce the final output of the prediction.

  During the learning process, for each iteration, weak learners focus on instances which were misclassified by previous classifiers. As long as the predictions of each single weak learner are better than random guessing, the classifier is guaranteed to converge to a so called "strong learner" and to output overall better predictions than its single weak learners.

  The literature indicates that it is often able to deliver good results out-of-the-box and it can work well on different types of data. One of its caveats is its sensitivity to noise and outliers. 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. 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	0.006
Training time (secs)	0.004	0.012	0.010
F1 - Test	0.828947	0.828947	0.805195
F1 - training	0.867133	0.88	0.866379
10-fold cv accuracy	0.63	0.63	0.63
Standard deviation	+/- 0.04	+/- 0.04	+/- 0.04

KNeighborsClassifier

	100	200	300
Prediction time (secs)	0.002	0.003	0.009
Training time (secs)	0.001	0.001	0.008
F1 - Test	0.781457	0.811189	0.813793
F1 - training	0.797101	0.841379	0.853933
10-fold cv accuracy	0.59	0.59	0.59
Standard deviation	+/- 0.10	+/- 0.10	+/- 0.10

AdaBoostClassifier

	100	200	300
Prediction time (secs)	0.041	0.026	0.030
Training time (secs)	0.478	0.522	0.783
F1 - Test	0.746269	0.797203	0.746269
F1 - training	1	1	1
10-fold cv accuracy	0.73	0.76	0.72
Standard deviation	+/- 0.08	+/- 0.07	+/- 0.11

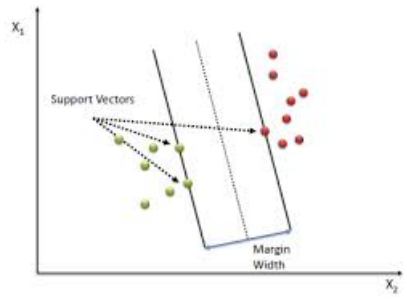
5 Choosing the Best Model

- Based on the performed experiments, SVM proved itself as a good model for the task at hand. In the experiments it presented stable and consistent results across different settings and sizes of the data set. Moreover, the training and prediction times for the SVM model were usually better than the times of the AdaBoostClassifier model. KNN often got the highest F_1 scores and also the best running times for training and prediction. However, its 10-fold cross-validation accuracy was the lowest among the three models and it also had the highest standard deviation for the 10-fold cross-validation. In general, K-nearest neighbors delivered good results for this data set, but it was not chosen due to the concern that small variations in the data set could lead to substantially poorer results.

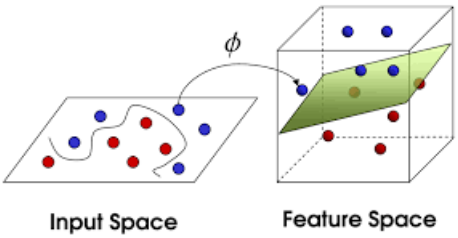
  The AdaBoost model performed inconsistently across different data set sizes and indicated a tendency to overfit rather easily (perfect performance on the training set).

  In conclusion, SVM seemed to be the most appropriate model for this task due to its robustness and consistency of accuracy on different sizes and portions of the data set.

- 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 (see picture). 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. In other words, it performs a margin maximization calculation.



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. SVM is also remarcable for its ability to separate data that at first do not seem to be linear. In the 2-dimensional example from before, imagine now that the input looks like the picture on the left below. In this case there is no evident straight line which is separates the dots. However, which a method called the "kernel trick", one can overcome this limitation by adding a third dimension to data (e.g. $(x \times y)^2$) and thus making the dots separable in a 3-D plane, like the figure below on the right shows.



- The model's final F_1 score stayed about the same on the test set even after some tuning using grid search. (see code output)