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)      0.001      0.004      0.006
Training time (secs)        0.003      0.005      0.008
F1 - Training            0.835443   0.843137   0.866379
F1 - Test                0.802548   0.810458   0.805195


KNeighborsClassifier
                              100        200        300
Prediction time (secs)      0.002      0.006      0.009
Training time (secs)        0.001      0.002      0.002
F1 - Training            0.825175   0.809689   0.853933
F1 - Test                0.758621   0.785714   0.813793


AdaBoostClassifier
                              100        200        300
Prediction time (secs)      0.022      0.022      0.029
Training time (secs)        0.408      0.442      0.731
F1 - Training                   1          1          1
F1 - Test                0.746479   0.721805   0.744526
```

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 (F_1 around $\approx 81\%$ for all training set sizes). Moreover, the training and prediction times for the SVM model were usually better than the times of the AdaBoostClassifier model and it also had better prediction times than the KNN model.

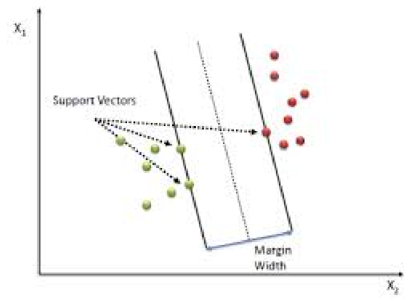
  KNN reached the highest F_1 score on the 300-test-set. Its scores on the test set were .76, .79 and .81 for sizes 100, 200 and 300, respectively. In general, K-nearest neighbors delivered good results for this data set, but it was not chosen due to the concern that small variations in the data could lead to substantially poorer results.

  The AdaBoost model had overall poorer results across all different data set sizes and indicated a tendency to overfit rather easily (.75, .72 and .74 for 100, 200 and 300, respectively, and perfect scores on the training set).

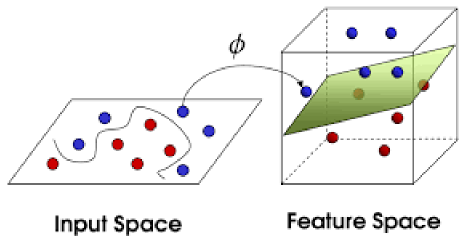
  In conclusion, SVM seemed to be the most appropriate model for this task due to its robustness and consistency of accuracy on different sizes and portions of the data set.

- Explanation of SVM in layman terms:

  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, say number of absences and free time, we could represent our students, i.e. our training-set, as dots on a two-dimensional Cartesian plane where blue dots could represent students who passed (few absences and little free time) and the red dots students who had plenty of absences and free time and did not pass (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 the algorithm "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 remarkable for its ability to separate data that at first does not seem to be linear. In the 2-dimensional example from before, imagine now that the data looks more like the picture on the left below. In this case there is no evident straight line which separates the dots (it might mean that in this scenario the relation between absences and free time is not so linear and plenty of free time per se, for example, doesn't necessarily mean that the student will fail). However, with a method called the "kernel trick", one can overcome this limitation by adding a third dimension to the data, for example, additionally to the plain values of x and y, we add another attribute with the result of $1/(x \times y)^2$. What this is roughly doing, is positioning the data points representing low absence and low free time higher than the dots representing high absence and high free time values in the 3-D space. Now, after applying this "trick", we can separate the dots, like indicated in the figure below on the right.



- The model's final F_1 score after some tuning using grid search was $\approx 80\%$, about the same score we had before on the test set. The chosen kernel was *rbf*, the *C* parameter was 128 and *gamma* was 0.0001.

```
Parameters for the best estimator:
SVC(C=128, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.0001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
F1 score for training set: 0.831223628692
F1 score for test set: 0.797297297297
```