Building a Student Intervention System

By : Shmuel Naaman

1. Classification vs Regression

The goal is to identify students who might need early intervention - In this case we are solving a “classification problem”. Each student in the data set will be categorize according to the parameters as "need early intervention" or does not "need early intervention". The algorithm will fit each student into one of these categories.

2. Exploring the Data

Some statistics facts about the dataset

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

3. Preparing the Data

Identify feature and target columns

Feature column(s):-

['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']

Target column: ‘passed’

Split data into training and test sets

So far, we have converted all *categorical* features into numeric values. In this next step, we split the data (both features and corresponding labels) into training and test sets.

Training set: 300 samples

Test set: 95 samples

4. Training and Evaluating Models

Three supervised learning models that are available in scikit-learn, and appropriate for this problem.

- General applications of each model, strengths and weaknesses.
- Reason for choosing each model
- Fit this model to the training data, prediction labels for both training and test sets, and measure the F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

The table showing training time, prediction time, F_1 score on training set and F_1 score on test set, for each training set size.

Support Vector Machines Classification O(n3):

General applications:

Given, data points that belong into two clusters, each data point can be represented by a n-dimensional vector. It is possible to find many n-1 dimensional hyper planes that will separate the data points into two clusters. Support Vector Machines Classification will find the hyper plane that represents the hyper plane with maximum separation or margins. SVM is helpful in text categorization, images classifications, highly efficient in protein classifications and classification of hand writers' characters.

Strengths:

Several specialized algorithms enable quick solution of the quadratic programming problem. Kernel SVM is available in many toolkits. SVM is able to separate feature spaces that are not linearly separated. SVM is effective for problem with high dimensional space. SVM is effective for cases where number of dimensions is greater than the number of samples. SVM is Memory efficient. [http://scikit-learn.org/stable/modules/svm.html]

Weaknesses:

Parameters of a solved model are difficult to interpret [https://en.wikipedia.org/wiki/Support_vector_machine]. The classification is into two class and the solution is un-celebrated (we cannot estimate the degree of certainty of a given solutions). All input data must be labeled (no unsupervised learning). If the number of features is much greater than the number of samples, the method is likely to give poor performances

Reasons for choosing this model.

The structure of the data set is a classic case for SVM, each label can be describe with a vector of values. The problem that we want to solve is classification. I hope that the training will be done in a short time. The solution that I need is binary and I do not care about the certainty for each student.

Implementation:

	Training size		
	100	200	300
Training time	0.002	0.008	0.014
F1 score training	0.945	0.92	0.9
Prediction time	0.001	0.002	0.003
F1 score test	0.829	0.825	0.83

AdaBoost Adaptive Boosting Classification

General applications:

A Meta-algorithm that can be used in conjunction with other types of learning algorithms and improve their performance. The output of the other learning algorithms ('weak learners') is combined as a weighted sum to create the boosted classifier. AdaBoost used as a Standard algorithm for Face and object Detection,

Strengths:

AdaBoost is less susceptible to over fitting than other learning algorithms. AdaBoost is the best out-of-the-box classifier. AdaBoost is simple to implement. AdaBoost is can perform feature selection

Weaknesses:

AdaBoost is sensitive to uniform noisy data and outliers. AdaBoost depends on data and weak learner and can fail if weak classifiers are overfit or underfit.

Reasons for choosing this model.

As part of the exploration I wanted to test a predictive model that will be less susceptible to overfitting and since Adaboost is consider the best out-of-the-box classifier, I wanted to check if a boosting model will do better.

Implementation:

	Training size		
	100	200	300
Training time	0.005	0.006	0.006
F1 score training	0.855	0.84	0.80
Prediction time	0.001	0.001	0.001
F1 score test	0.83	0.83	0.83

Extremely Randomized Trees

General applications:

Ensemble learning method for classification, that construct a multitude decision trees at training, outputting the class of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. The algorithm, inducing random forest "bagging" idea and the random selection of features, in order to construct a collection of decision trees with controlled variance. Random forests use tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. Extremely randomized trees or ExtraTrees are trained like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way. Hydrological problem. Computational intensive problems. Modelling of large datasets and input selection.

Strengths:

More complex classifier (a larger forest) getting more accurate nearly monotonically. Combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. effective option to balance accuracy and computational efficiency in data-driven modelling.[http://www.hydrol-earth-syst-sci.net/17/2669/2013/hess-17-2669-2013.pdf]

Weaknesses:

The larger the number of trees in the forest the better, but also the longer it will take to compute. The results will stop getting significantly better beyond a critical number of trees.

Reasons for choosing this model.

The fact that we deal with students and as we know humans are not easy to predict, a model that will include randomization in the process of creating classifiers might captured something that could not be captured by the well formulated models.

Implementation:

	Training size		
	100	200	300
Training time	0.083	0.072	0.094
F1 score training	1.0	1.0	1.0
Prediction time	0.007	0.007	0.008
F1 score test	0.818	0.78	0.77

5. Choosing the Best Model

The training time for SVM is 0.014 sec, Adaboost 0.006 and for Extremely Randomized Trees 0.094.
Prediction time for SVM is 0.003 sec, Adaboost 0.001 and for Extremely Randomized Trees 0.008.
F1 score for the SVM is 0.83, Adaboost 0.83, and for Extremely Randomized Trees 0.77. (Running the model again provides different numbers, but the difference is not significant, ~0.8 for all models).

Considering the above, and the given problem limitations, the most appropriate algorithm is the Adaboost. Adaboost performed best when considering the computation time and the performance is similar for all the three models. Since the problem required efficient model that is accurate Adaboost provide both.

- Explaination in layman's terms how the final model chosen is supposed to work .

Layman's Adaboost:

Description of the algorithm. A Meta-algorithm that can be used in conjunction with other types of learning algorithms (in this case decision trees) and improve their performance. The output of the other learning algorithms ('weak learners') is combined as a weighted sum to create the boosted classifier.

Description of how the algorithm is trained. AdaBoost uses a number of training samples to pick a number of good 'classifiers'. In this case we consider classifiers as the features that describe each student. Good features (classifiers) will be features that distinguish between students that need intervention to students that do not need interventions. AdaBoost will look at a number of classifiers and find out which is the best predictor of a label (intervention) based on the sample. After it has chosen the best classifier it will continue to find another until some threshold is reached and those classifiers combined together will provide the end result.

How the algorithm makes predictions. The output of the learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. For each student the model weight the corresponding feature according combined weighted sum that provids the final classification whether the student need or not intervention.

Fine-tune the model using Gridsearch
The model's final F_1 score: 0.82

Reference

http://wikipedia.org/
http://stackoverflow.com/
http://scikit-learn.org/