Project 2: Supervised Learning

Building a Student Intervention System

1. Classification vs Regression

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

A1. :

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%
- Number of features : 31

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’

Preprocess feature columns

As we can see, there are several non-numeric columns that need to be converted! Many of them are simply yes/no, e.g. internet. These can be reasonably converted into 1/0 (binary) values.

Other columns, like Mjob and Fjob, have more than two values, and are known as *categorical variables*. The recommended way to handle such a column is to create as many columns as possible values (e.g. Fjob_teacher, Fjob_other, Fjob_services, etc.), and assign a 1 to one of them and 0 to all others.

These generated columns are sometimes called *dummy variables*, and we will use the pandas.get_dummies() function to perform this transformation.

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

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem. For each model:

- What are the general applications of this model? What are its strengths and weaknesses?
- Given what you know about the data so far, why did you choose this model to apply?
- Fit this model to the training data, try to predict 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.

Produce a 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 that enable quick solving of the quadratic programming problem arise from the model. Kernel SVM is available in many toolkits. For images classifications SVMs achieve significantly higher search accuracy than traditional query refinement schemes.

Weaknesses:

The model is not easy to interpret. The classification is into two class and the solution is uncelebrated (we cannot estimate the degree of certainty of a given solutions). All input data must be labeled (no unsupervised learning).

Why did you choose this model to apply?

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.

Training set size: 100
Training SVC...
Training time (secs): 0.003

Prediction time (secs): 0.001
F1 score for training set: 0.9343065693430657

Prediction time (secs): 0.002
F1 score for test set: 0.7801418439716312

Training set size: 200
Training SVC...
Training time (secs): 0.007

Prediction time (secs): 0.004
F1 score for training set: 0.910958904109589

Prediction time (secs): 0.003
F1 score for test set: 0.7746478873239436

Training set size: 300
Training SVC...
Training time (secs): 0.015

Prediction time (secs): 0.009
F1 score for training set: 0.9135254988913526

Prediction time (secs): 0.003
F1 score for test set: 0.7464788732394366

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:

Less susceptible to overfitting than other learning algorithms, the best out-of-the-box classifier.

Weaknesses:

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

Why did you choose this model to apply?

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.

Training set size: 100
Training AdaBoostClassifier...
Training time (secs): 0.010

Prediction time (secs): 0.001
F1 score for training set: 0.8169014084507042

Prediction time (secs): 0.002
F1 score for test set: 0.7887323943661971

Training set size: 200
Training AdaBoostClassifier...
Training time (secs): 0.006

Prediction time (secs): 0.001
F1 score for training set: 0.8243243243243242

Prediction time (secs): 0.001
F1 score for test set: 0.7971014492753624

Training set size: 300
Training AdaBoostClassifier...
Training time (secs): 0.005

Prediction time (secs): 0.001
F1 score for training set: 0.8278867102396513

Prediction time (secs): 0.001
F1 score for test set: 0.7887323943661971

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.

Strengths:

More complex classifier (a larger forest) getting more accurate nearly monotonically.

Weaknesses:

For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favored over larger groups.

Why did you choose this model to apply?

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.

Training set size: 100
Training ExtraTreesClassifier...
Training time (secs): 0.096

Prediction time (secs): 0.011
F1 score for training set: 1.0

Prediction time (secs): 0.009
F1 score for test set: 0.7714285714285714

Training set size: 200
Training ExtraTreesClassifier...
Training time (secs): 0.086

Prediction time (secs): 0.010
F1 score for training set: 1.0

Prediction time (secs): 0.008
F1 score for test set: 0.7285714285714285

Training set size: 300
Training ExtraTreesClassifier...
Training time (secs): 0.092

Prediction time (secs): 0.010
F1 score for training set: 1.0

Prediction time (secs): 0.008
F1 score for test set: 0.8082191780821918

5. Choosing the Best Model

- Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?

The training time of SVM achieve the shortest time with 0.01 sec, where the Adaboost and the Extremely Randomized Trees achieved 0.06 0.1 respectively. This is expected, considering the fact that SVM is usually fast model.

Prediction in all cases is ~ 10 times lower than the corresponding training time (SVM 0.003 Adaboost 0.005 Extremely Randomized Trees 0.010).

F1 score for the SVM model depict decreasing trend as Training set size increase, where F1 score for the Adaboost and Extremely Randomized Trees depict increasing trend.

Considering the above, and the given problem limitations, the most appropriate algorithm is the Adaboost. The main reason for that is the nature of the data set, which is expected to be large and even larger in the feature. SVM is the fastest model, but the performance of the mode decrease for larger training size. Adaboost provides a stable or even increasing performance as the training set increase and reasonable computation time that is longer than the SVM but shorter than the Extremely Randomized Trees.

- In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).

AdaBoost uses a number of training sample to pick a number of good 'classifiers'. AdaBoost will look at a number of classifiers and find out which is the best predictor of a label 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.

- Fine-tune the model. Use Gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.
- What is the model's final F_1 score? F1 score for test set: 0.79

Reference

http://wikipedia.org/

http://stackoverflow.com/

http://scikit-learn.org/