1. Classification vs Regression

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

Classification - Classification is simply the process of taking input and mapping it to some discrete labels, while regression mapping input to continuous value. Since I need to identify students who needs early intervention or not (in the other words, true or false), I can say this type of supervised machine learning problem is “Classification”.

2. Exploring the Data

Can you find out the following 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 (excluding the label/target column): 31

Use the code block provided in the template to compute these values.

3. Preparing the Data

Execute the following steps to prepare the data for modeling, training and testing:

- Identify feature and target columns
- Preprocess feature columns
- Split data into training and test sets

Starter code snippets for these steps have been provided in the template.

See ipython notebook.

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 F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.
- Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

Note: You need to produce 3 such tables - one for each model.

Supervised learning models (classes) which I chose for this problem

- Logistic Regression (sklearn.linear_mode.LogisticRegression)
- AdaBoost (scikit.ensemble.AdaBoostClassifier)
- Support Vector Machines (scikit.svm.SVC)

Logistic Regression (sklearn.linear_model.LogisticRegression)

- What are the general applications of this model? What are its strengths and weaknesses?
    - The general application of the model:
        - **Classification**
    - The strengths of the model:
        - **Less computation resources** are required for training and prediction compared with more sophisticated models such AdaBoost, SVM.
    - The weaknesses of the model:
        - **A bigger amount of training data** is required to avoid overfitting compared with more sophisticated models such as SVM.
- Given what you know about the data so far, why did you choose this model to apply?
    - Since the problem is binary classification and the input data contains different types of features, logistic regression can be applied. In addition logistic regression would be a better choice if there are strict resource limitations because logistic regression requires less computational resources for training and prediction, and can avoid overfitting by L1 or L2 regularization.
- A table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

Logistic Regression scikit.linear_model.LogisticRegression	Training set size		
	100	200	300
Training time (secs)	0.001792	0.002542	0.002101
Prediction time (secs)	0.000341	0.000350	0.000226
F1 score for training set	0.824427	0.842857	0.838407
F1 score for test set	0.757576	0.770370	0.773723

AdaBoost (scikit.ensemble.AdaBoostClassifier)

- What are the general applications of this model? What are its strengths and weaknesses?
    - The general application of the model:
        - **Classification**
    - The strengths of the model:
        - **More accuracy and less overfitting than** its basis learner such as decision tree (a wide range of base learners can be used).
    - The weaknesses of the model:
        - **Much more computational resources** are required than basis learner.
- Given what you know about the data so far, why did you choose this model to apply?

    - Since the problem is binary classification and the input data contains different types of features, AdaBoost with decision tree as a basis learner can be applied. Decision tree itself also can be applied to this problem, but it tends to overfit to training data and takes parameter tuning to avoid overfitting. However, if decision tree works as a basis learner of AdaBoost, it can effectively avoid overfitting and utilize decision tree for learning.
- A table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

AdaBoost scikit.ensemble.AdaBoostClassifier	Training set size		
	100	200	300
Training time (secs)	0.088589	0.106152	0.096139
Prediction time (secs)	0.003511	0.004123	0.005298
F1 score for training set	0.960630	0.856089	0.846337
F1 score for test set	0.688525	0.787879	0.770370

Support Vector Machines (scikit.svm.SVC)

- What are the general applications of this model? What are its strengths and weaknesses?
    - The general application of the model:
        - **Classification**
        - **Regression**
        - **Outliers detection**
    - The strengths of the model:
        - **Is effective in high dimensional spaces.**
        - **Is memory efficient** because of support vectors.
    - The weaknesses of the model:
        - **Requires more computational and storage resources** with the number of training vectors.
- Given what you know about the data so far, why did you choose this model to apply?
    - Since the problem is binary classification and the input data is high dimensional vectors (almost 50 features), SVM is one of the best choices because SVM is effective in high dimensional spaces. In addition, scikit.svm is libsvm-implementation, which means it works efficiently compared with pure python-based models.

Support Vector Machines scikit.svm.SVC	Training set size		
	100	200	300
Training time (secs)	0.001420	0.002859	0.005231
Prediction time (secs)	0.001458	0.002281	0.004221
F1 score for training set	0.828947	0.861184	0.852878

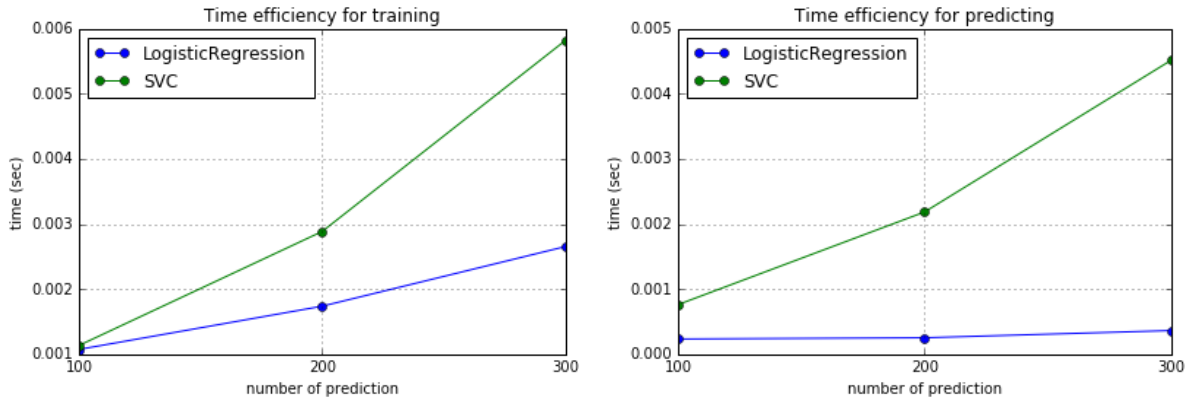
F1 score for test set	0.8	0.851351	0.835616

5. Choosing the Best Model

Based on the experiments you performed earlier, in 2-3 paragraphs explain to the board of supervisors what single model you choose as the best model. Which model has the best test F1 score and time efficiency? Which model is generally the most appropriate based on the available data, limited resources, cost, and performance? Please directly compare and contrast the numerical values record to make your case.

I conclude the best model is logistic regression in terms of both effectiveness and computation cost. Based on the experiments, I found that 1) logistic regression marks the highest time efficiency and the second best test F1 score, 2) AdaBoost marks the lowest time efficiency and worst test F1 score, and 3) SVM marks second highest time efficiency with available data set and best test F1 score. Thus, the results show that there are no model which has both the best test F1 score and time efficiency. Below, I will explain why SVM is better than logistic regression.

Firstly, in terms of time efficiency, I conclude logistic regression is the best model because I assume that the student intervention system need to diagnose over 100,000 students every week or month. Note that SVM will rapidly lose time efficiency when training data size increases (see below figures). The figures compares average computation time out of 100 iterations for training (left-side) and predicting phase (right-side). With 300 data set, this difference seems to be small but lead budget problem if the number grows rapidly.



Secondly, in terms of both effectiveness and time efficiency, I still conclude logistic regression is the best model. Note that effectiveness is mandatory to reach a 95% graduation rate by the end of the decade. To reach 95% graduation rate by only the student intervention software, model's F1 score must be around 95% but currently there is no such great model. Therefore, I suggest that we should develop a pilot system with logistic regression first, and change the model to more sophisticated one if more budget is allocated for the student intervention system.

In 1-3 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 learn to make a prediction).

Logistic regression works in a 3 steps. 1) It learns parameters of a function. This function maps input multiple values (e.g. sex and age) to a 1-dimensional value from training data and the value is

supposed to be positive if corresponding target value is “yes” (passed final exam) or negative if the target value is “no” (failed final exam). 2) Then, It converts the positive value to 1 and the negative value to 0 with special functions named sigmoid and sign. 3) Finally, converted value is 1 or 0 and they mean “yes (will pass exam)” and “no” (will not pass exam) respectively.

In logistic regression, the most important step is of course the 1) learning step. In this step, we need to optimize two important configurations such as penalty and C to learn better function which generally works well with unseen inputs. To optimize these settings I conducted several time tests and finally found almost best configurations.

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.

Table below shows 4 parameters which I used with GridSearchCV and 2 important parameter to be tuned also shown as input value of param_grid.

Parameters for GridSearchCV	**Input Value**
estimater	LogisticRegression()
param_grid	{‘C’: [0.01, 0.02, …, 0.99, 1.0], ‘penalty’: [‘l1’, ‘l2’]}
scoring	‘f1’
cv	3

As a result of above inputs, GridSearchCV returned best params:

```
{'C': 0.08000000000000002, 'penalty': 'l1'}
```

What is the model’s final F1 score?

F1 score for training data: 0.823
F1 score for test data: 0.813