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?

- This question is raising a binary answer as do the student need early intervention or not. So we are searching for a discrete values output and therefore we can say that this is a classification problem.

2. Exploring the Data

Can you find out the following facts about the dataset? Total number of students Number of students who passed Number of students who failed Graduation rate of the class (%age) Number of features Use the code block provided in the template to compute these values.

- We cand find the following fact about the dataset with panda's dataframe or numpy's array. There is 395 student in this dataset, 265 who passed the graduation, and 130 who don't so this gave us a 67.09% graduation rate. And for each student we have 30 features.

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 is the theoretical O(n) time & space complexity in terms of input size? 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.

Model 1 : SVM

SVM is generally between O(n^2) and O(n^3) with n the amount of training instances.

http://stackoverflow.com/questions/16585465/training-complexity-of-linear-svm

Why i choose this model ? Because SVM is very effective when we have a lot of features on a datasets (high dimensional spaces) and when the dataset isn't too large. So with 30 features i think SVM is appropriate. And to add at his strength, SVM is using a subset of training points in the decision function (support vectors) so it is also memory efficient.

For the weaknesses, SVM is inefficient with many training example and will be hard to train (>50K).

Ok i've done some research on it and i was wrong, my bad:

SVMs work fine on sparse and unbalanced data. Class-weighted SVM is designed to deal with unbalanced data by assigning higher misclassification (positive or negative) penalties to training instances.

The F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy which is only good for symmetric data sets where the class distribution is 50/50 and the cost of false positives and false negatives are roughly the same.

	Training set size		
	100	200	300
Training time (secs)	0.002	0.05	0.008
Prediction time (secs)	0.002	0.03	0.005
F1 score for training set	0.8571	0.8662	0.8589
F1 score for test set	0.7625	0.7862	0.7866

Model 2 : ADABoost

AdaBoost's time complexity is trivially O(T f), where f is the runtime of the weak learner in use. http://stackoverflow.com/questions/22397485/what-is-the-o-runtime-complexity-of-adaboost

As we have a lot of features here, i wanted to try an ensemble trees classifier on this dataset. These algorithms handle very well high dimensional spaces. My curiosity make me try some averaging (bagging) methods like RandomForest but it did not perform very well on the testing set so i try boosting methods with ADABoost and it did perform quite well even without tuning parameters (and i think with boosting method we should tune them a little).

But When i saw what the training time was, i understood that this model will not be my final choice even with a quite good F1 score.

	Training set size		
	100	200	300
Training time (secs)	0.059	0.054	0.064
Prediction time (secs)	0.006	0.000	0.010
F1 score for training set	0.9919	0.8989	0.8584
F1 score for test set	0.7377	0.7794	0.7391

Model 3 : KNN

KNN time complexity is close to *O(M·logN).*

http://www.alglib.net/other/nearestneighbors.php

Then i tried kNN. Because kNN is a simple and good classification model. And he is very efficient in term of memory/computing time. It can be explained as a model who classified an object by a majority of vote of its neighbors. (KNN uses the idea of "nearness" to classify new entries.)

Since it's a simple model, i guess too much data or complex datasets with lots of features will make kNN a bad model. (In a very high dimensional space the distance to all neighbors will become more or less the same, and the notion of nearest and far neighbors can become blurred.)

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.001	0.003	0.005
F1 score for training set	0.8387	0.8438	0.8622
F1 score for test set	0.7906	0.7703	0.7659

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? 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). 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 F1 score?

I choose the first model, the Support Vector Machine Classifier because it mades the best F1 score and even if it is not the best model in term of memory/cost efficiency, i find it pretty

reasonable in comparison of the more complex ADABoost. To explain in Laymans Terms what SVM does, i will speak about balls. Now, suppose we want to split two different colors of balls by drawing a line between each color. We now that there are maybe an infinite number of lines that will accomplish this task. SVM just try to find the "maximum-margin" line to make the best balls split. (with some tolerancy about noise or outliers). The model's final F1 score is 0.797202797202797.

Describing the parameter in the SVC :

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. The Kernel parameter specifies the kernel type to be used in the algorithm. The coef0 was chosen in case where the gridsearch select a polynomial or a sigmoid kernel. This parameter will have made the kernel non-symmetric which can be used to overcome one of the important issues with the polynomial kernel. (i should have replaced this one with scoring=f1)

About GridSearchCV : Of course I should have used a two fold CV like the ones in the training set, i did not think about what was the default one (and i didnt know that it was 3 folds), thanks for the precision and the explanation.

About F1 score parameter in GridSearch : I forgot to used this one too, i think it could have end in a better score for the final model.

Final Note :

During this project, i understant that the structure of the data and the data itself was really important. I try a few times to re-shuffle the data, and the models behavior was changing (at a point where i did not know which one to choose for my final model between SVM and kNN). Our choice of classification algorithm might not really matter so much in terms of classification performance because better data often beats better algorithms. And i think this dataset is working well with a lot of models.