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 is a classification problem, because in this problem we are trying to map input (X) to discrete/categorical variable. In a classification problem, we try to put X into a specific class, but in regression we try to predict a continuous value. For example, House_Price is a regression problem because we're trying to predict a value (which is a continuous value). But in this problem we are trying to classify inputs to two classes, YES or NO.

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) **30**

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.

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?

1) Decision Tree:

Applications:

There is a wide range of usage and application for decision tree for example

- Credit card risk analysis
- Modeling calendar scheduling preferences
- Medical diagnosis

We consider decision tree when the target function is discrete value and instances describable by attribute-value pairs. Decision tree has different algorithms to split data such as (ID3, C4.5, C5, CART, …) that Sklearn only support CART. (Based on its documentation)

Strengths and Weaknesses:

- It's vary fast to train and evaluate
- It's easy to interpret
- We can't have two explanatory variables that behave independently. Every variable in the tree is forced to interact with every variable further up the tree
- Poor resolution on data with complex relationships among the variables
- There is no confidence measure.

2) AdaBoost

AdaBoost can use multiple instances of the same classifier with different parameters. It combines weak classifiers to build strong classifiers.

Application:

- Automatic "store front" or "help desk" for AT&T Labs' Natural Voices business, caller can request demo, pricing information, technical support, sales agent and interactive dialogue
- Find faces in photograph or movie (weak classifiers: detect light/dark rectangles in image)

Strengths and Weaknesses:

- Computational cost is expensive
- Fast (but not quite as fast as other methods)
- Provable guarantees
- Tends not to over fit (but occasionally does)

3) SVM

Applications:

- Classify text documents (categorize news articles by topic)

- Recognizing Handwritten Characters

Strengths and Weaknesses:

- Power and flexibility from kernels (kernel trick)
- Fast algorithms, but not so simple to program
- Maximizing the margin
- Many SVM implementations are available.

- Given what you know about the data so far, why did you choose this model to apply?

We know features and class are discrete (i.e. not continuous) and categorical, we also know some of our features are not numerical and we have to convert it to numerical equivalent.

I chose Decision Tree because its learning and prediction time is really fast, and AdaBoost because it has the best F1_score_for_test set and finally I chose SVM because it has the best tradeoff between F1_score_for_test, Training Time and Prediction Time.

- 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.

SVM	Training set size		
	100	200	300
Training time (secs)	0.001	0.003	0.006
Prediction time (secs)	0.001	0.002	0.004
F1 score for training set	0.8518	0.8616	0.8763
F1 score for test set	0.7898	0.7866	0.7659

Decision Tree	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.002
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.7301	0.6612	0.704

AdaBoost	Training set size

	100	200	300
Training time (secs)	0.017	0.020	0.019
Prediction time (secs)	0.001	0.002	0.002
F1 score for training set	0.9014	0.8474	0.8763
F1 score for test set	0.7727	0.7101	0.8321

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

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 recored to make your case.

I chose SVM as the best model. Firstly, because its training time (0.007 s) and F1 Score (76.59%) is more reasonable than AdaBoost, despite AdaBoost's F1 score (83.61%). Additionally, Adaboost is really slow in learning process (0.021) in comparison with SVM (0.007).

Decision Tree in comparison with SVM has better training time (0.002) but there is a large gap between their F1 scores. Decision Tree F1 score is 66.66% whereas SVM is around 76.59%. SVM is not the fastest or most accurate algorithm between these three algorithms, but if we make a tradeoff between F1 score and time efficiency.

I think SVM is the best if we want to consider limited resources, cost, and performance all together

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).

SVM tries to separate/split a list of students by using their attributes (features) such as (Freetime, Absence, Activity, Gender and etc.) between two types. It tries to find either one or a combination of attributes to distinguish between students.

It technically tries to put/draw the entire student on the plot along the x- and y-axes and then make/find a separation line between these two types of students. And then for the students we can categorize them based on their position on the plot. Indeed, the algorithm would like to separate the instances by a maximum margin. This is so that unseen examples are more likely to be classified correctly.

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?

%78.98 which is better than %76.59

F1 score for training set: 0.8071
F1 score for test set: 0.7898

The Best: {'kernel': 'sigmoid', 'coef0': 0.0, 'gamma': 'auto', 'degree': 2}