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?

[A]: It's classification problem, because the output is a discrete value: 'Yes' or 'No'.

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 (%)
- Number of features (excluding the label/target column)

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

[A]:

Total number of students: 395

Number of students who passed: 265

Number of students who failed: 130

Number of features: 30

Graduation rate of the class: 67.00%

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?
- 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.

[A]:

1. Logistic regression:

Pros: It is a pretty well-behaved classification algorithm and is widely used. We can set baseline with it for a problem. We expect the problem to be linear separable. But we can turn non-linear separable problem into linear separable. The final output can be interpreted as a probability, there exist efficient implementations which can be distributed for big data scenarios.

Cons: It has limited expressive power compared to SVM. We need to manually transform features to apply to non-linear separable problems. Transformations are needed for categorical features. It's fundamentally a binary classifier, use one-vs-all to deal with multi-class problems.

Given the simplicity and generality, we can choose it to start with and set a baseline.

	Training set size		
	100	200	300
Training time (secs)	0.001	0.002	0.004
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	0.857142857143	0.838028169014	0.838137472284
F1 score for test set	0.761194029851	0.779411764706	0.791044776119

2. Support vector machine:

Pros: High accuracy, nice theoretical guarantees regarding overfitting. It can handle non-linear decision boundaries by using kernel functions. It can handle large feature spaces which makes them one of the favorite algorithms in text analysis which almost always results in huge number of features where logistic regression is not a very good choice.

Cons: It is not efficient to train. Sometimes it's not easy to find the appropriate kernel function. The final output is incomprehensible. Same as LR, It's fundamentally a binary classifier, use one-vs-all to deal with multi-class problems.

The number of our dataset is moderate, for the sake of accuracy we should try SVM to see if it gives us better result.

	Training set size		
	100	200	300
Training time (secs)	0.003	0.003	0.009

Prediction time (secs)	0.002	0.002	0.005
F1 score for training set	0.859060402685	0.869281045752	0.869198312236
F1 score for test set	0.783783783784	0.775510204082	0.758620689655

3. Decision tree:

Pros: Requires little data preparation. It does not expect linear features or even features that interact linearly. Since decision trees are designed to work with discrete intervals or classes of predictors, categorical features are not really an issue with decision trees. Models obtained from decision tree is fairly intuitive and easy to explain. It can handle high dimensional spaces and large number of training examples. It can handle multi-class problems well.

Cons: It may overfit the data. We can make a decision tree model on your training set which might outperform all other algorithms but it'll prove to be a poor predictor on test set. We'll have to rely heavily on pruning and cross validation to get a non-over-fitting model with Decision Trees. The learning algorithm can't guarantee to return the globally optimal decision tree.

We don't know if our problem is linear separable or not, we should try decision tree. It handles both cases well. There is no class dominates the training set, so learning algorithm won't create a biased tree.

	Training set size		
	100	200	300
Training time (secs)	0.003	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	0.666666666667	0.723076923077	0.730434782609

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.

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).

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?

[A]: I choose logistic regression model. This problem needs a binary classifier and LR works well in the experiments. LR has best test F1 score - 0.791044776119 and relatively low training time - 0.004 for training set size 300.

Decision tree model has lowest training time and significant F1 score for training set, but apparently it overfits the data. So the test F1 score for DT is not very good. For SVM, it has highest training time and prediction time, and the test F1 score decreases as more data is used to train the model. Based on the experiments, I would choose LR. But I think after some parameter tuning, DT and SVM could acheive similar test F1 score as LR.

Explain LR: There is a function ranging from 0 to 1, it has many unknown parameters. We want to know all the parameters and use this function to predict answer. If function ouput >= 0.5, it means 'passed', otherwise it means 'not passed'. Now we use the training data to learn these parameters. It's an iterative learning process which the goal is to minimize the error between the real ouput and

the function output. The error function we want to minimize has a bow shape. For each iteration, we move closer to the certer of the bow. At last, we will reach the center and get all the parameters for the predictor function.

Final F1 score: 0.791666666667