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?

Solution:

Since the output labels are *discrete* (yes/no) in this case, it is clearly a *classification* problem. Regression usually deals with predicting continuous outputs (like price prediction).

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.

Solution:

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.

Solution:

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

Preprocessing: We need numerical data for classifier algorithms to be applied. Following are the steps followed in preprocessing.

a. Change yes/no to 1/0
b. For categorical columns, create as many columns as possible values and assign a 1 to one of them and 0 to all others.

Train Test data split

To ensure random samples, first shuffling has been performed. After that, train test data split has been performed.

4. Training and Evaluating Models

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

1. Naïve Bayes classifier
2. Support vector machines
3. Decision tree classifier

For each model:

- What are the general applications of this model? What are its strengths and weaknesses?

Naïve Bayes classifier:

a. Based on probability.
b. Straightforward and simple algorithm. Requires small amount of training data to estimate the parameters. Provides decent results in most of the cases.
c. Loss of accuracy due to class conditional independence. Dependency among variables cannot be modelled.

Support vector machines:

a. Based on margin maximization.
b. Good when there is a good marginal separation. Not so well with large datasets (lots of data and features) because of high running time.
c. Does not work well when there is a lot of noise and overlapping classes.

Decision tree classifier:

a. Based on entropy based trees.
b. Easy to understand, and rules can help us understand the underlying logic behind good classification.
c. May suffer from Overfitting. Tree can be quite large, pruning might be needed.

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

Since the prediction is discrete (pass/fail), we need classification model (no the regression model). I have chosen the above three models because these are diverse in nature. I wanted to compare 3 totally different models by nature (refer point ***a*** in the previous answer for algorithms basic nature). Considering that our data set is small (395 rows, 48 columns after processing), SVM should not be very computationally expensive. Also this much of data should be good enough to train Naïve Bayes algorithm. Decision tree algorithm is rule based, so looking at its performance will be exciting.

- 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: time in seconds)

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

For all the following tables, test size remains constant, irrespective of train size. (As asked)

Gaussian Naïve Bayes (Wihout parameter optimization)					
Training Size	Training Time	Prediction Time (on Train)	Prediction Time (on Test)	F1: Train	F1: Test
300	0.001	0.003	0.002	0.798	0.746
200	0.002	0.001	0.000	0.824	0.788
100	0.002	0.001	0.001	0.433	0.422

SVM (Wihout parameter optimization)					
Training Size	Training Time	Prediction Time (on Train)	Prediction Time (on Test)	F1: Train	F1: Test
300	0.036	0.016	0.006	1.000	0.770
200	0.018	0.009	0.004	1.000	0.771
100	0.005	0.003	0.002	1.000	0.805

Decision Tree (Wihout parameter optimization)					
Training Size	Training Time	Prediction Time (on Train)	Prediction Time (on Test)	F1: Train	F1: Test
300	0.006	0.002	0.001	0.878	0.737
200	0.003	0.001	0.000	0.907	0.765
100	0.002	0.001	0.000	0.887	0.781

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 recorded 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?

For choosing the best model, I run the SVM and Decision tree (Leaving Gaussian NB since it shows lot of variation with change in training size) for different parameter combinations (gridsearch). The results are tabulated as below.

SVM (With parameter optimization)					
`Optimal Parameters: {'kernel': 'rbf', 'C': 5, 'gamma': 0.01}`					
Training Size	**Training Time**	**Prediction Time (on Train)**	**Prediction Time (on Test)**	**F1: Train**	**F1: Test**
300	26.568	0.001	0.006	0.887	0.834
200	8.683	0.020	0.010	0.882	0.831
100	1.703	0.000	0.000	0.931	0.782

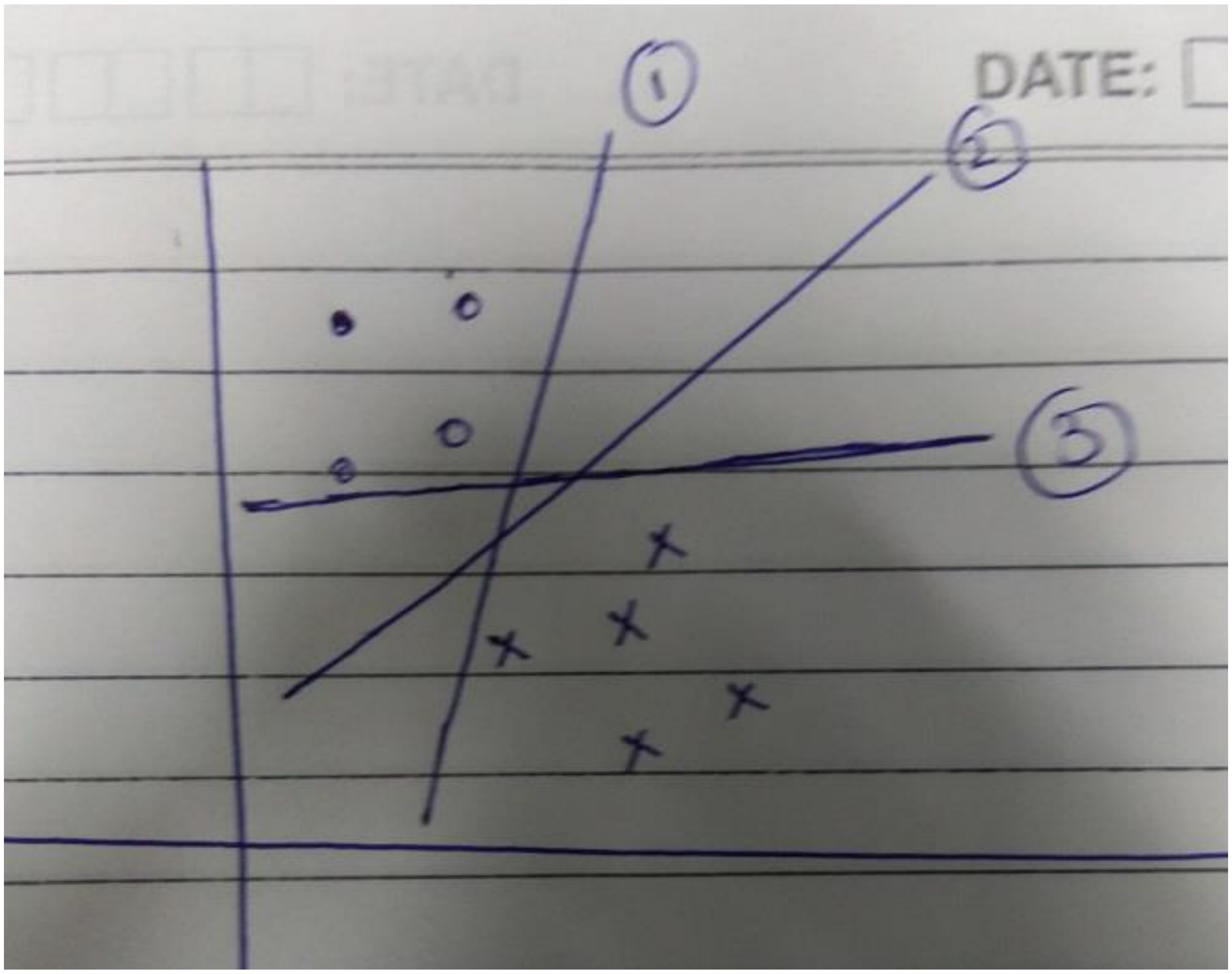
Decision Tree (With parameter optimization)					
`Optimal Parameters: {'max_depth': 1}`					
Training Size	**Training Time**	**Prediction Time (on Train)**	**Prediction Time (on Test)**	**F1: Train**	**F1: Test**
300	0.259	0.000	0.000	0.812	0.805
200	0.216	0.016	0.000	0.825	0.805
100	0.159	0.000	0.001	0.822	0.805

If we compare these two tables, we see that **SVM** gives the best F1 (Test) score of 0.834, whereas Decision Trees has best F1 (Test) score of 0.805. Prediction time for both the algorithms are very low (so, small difference does not matter much). SVM takes much higher time for training. But, since our data set is very small this is something we can tolerate. Also model is trained only once and predictions are made many times. So, prediction time is of prime importance. As said, prediction time is very low for both the algorithms, it is better to use SVM, since it gives substantially high F1 (Test score). So I choose SVM as best model, with parameter as specified in the above table. And since training time for training size = 200 is substantially lower than training time for training size=300, also the drop in F1 score is very low, I will prefer training with training size=200. And hence final model's **F1 score is 0.831** (not 0.834).

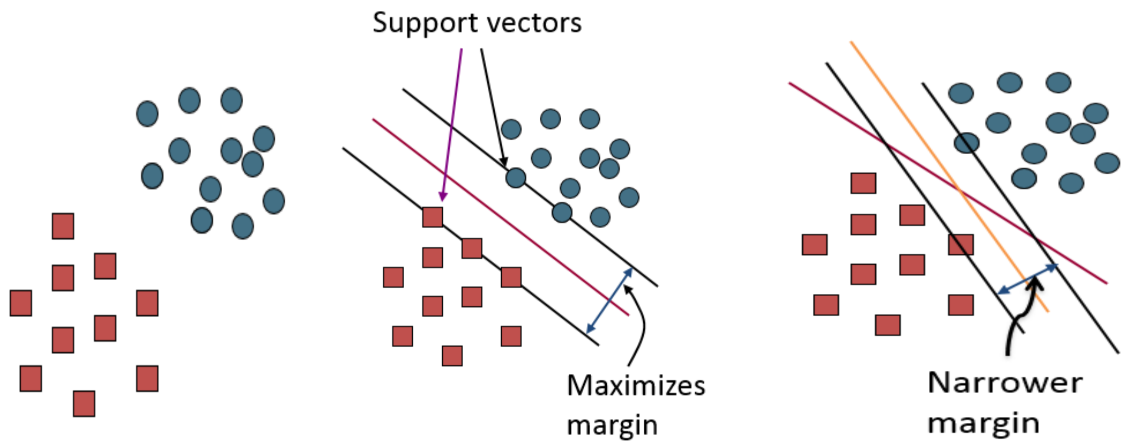
My explanation to board of directors (about SVM: in layman terms):

We will be doing predictions using SVM. SVM is a pretty simple and effective model.

I will draw the following picture and ask them to pick a line of their choice which they think does the best job of separating dots to crosses.

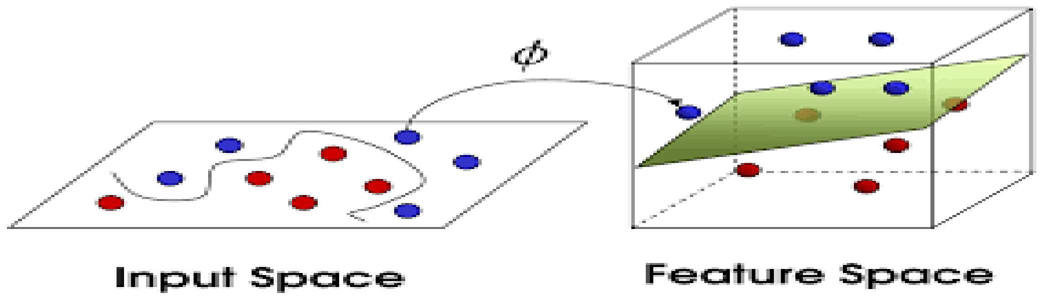


Once they choose the line 2, i'll tell them that SVM also thinks like you. This algorithm draws the line 2 for you. The reason it does that is because line 2 has maximum margin. To explain the concept of margin and support vectors, I'll use the following pictures. So, SVM picks the separator with maximum margin.



But drawing a straight line (or any linear boundary) is not always possible. I'll ask them to draw a straight line to separate the following two classes of data (LHS of following figure). Can a straight line separate these two classes?

Once they say no, I will ask them to draw any line (not necessarily straight) to separate these classes. How does SVM get this line? By projecting points in a higher dimensional space. Not clear? Wait.



For explaining the kernel trick (to explain that even if we don't have a linearly separable data, we can get a linear separation in higher dimensional space):

Consider lot of empty blue and red balls are floating in water, not clearly separable. To separate, SVM injects some liquid to blue balls so that they sink down the water, while it lets the red balls untouched (floating). Now it picks the floating balls (red) easily there by separating it from blue balls.

// I'll Play this short video for explaining the kernel trick. (https://www.youtube.com/watch?v=3liCbRZPrZA&feature=youtu.be)

You can choose among different sets of available kernels and set parameter such as c (controls the cost of misclassification on the training data), gamma (kernel coefficients) etc. Predictions are made on the basis of which side of the hyper-plane the point lies in.