Student Intervention System

Classification vs Regression

Identifying student who might need early intervention is a classification problem, because the goal of the prediction output is discrete: whether the student will pass or not.

Exploring the Data

- 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: **30** (excluding the label)

Training and Evaluating Models

Support Vector Machine

SVM is typically a good choice for complex domain with many features/dimensions and small data set. Our data set has 30 features initially, and 48 features after feature expansion, which makes SVM an ideal algorithm to evaluate.

Strengths:

- Kernel trick provides a flexible way to inject domain knowledge.
- Performs well in complicated domain with many features / dimensions.

Weaknesses:

- Doesn't scale well for large data set.
- Does not perform well when the data contains a lot of noise (a lot of overlapping between classes)
- If the default Kernel isn't the best fit, finding alternative kernel require more domain knowledge as well as some experience in finding best Kernel.

SVM	Training set size		
	100	200	300
Training time (secs)	0.002	0.003	0.007
Prediction time (secs)	0.001	0.002	0.009
F1 score for training set	0.8811	0.8759	0.8528
F1 score for test set	0.7947	0.8497	0.8553

Naive Bayes

Naive Bayes classifiers is a scalable algorithm for problems with a lot of features and large data set. The requirement to minimize computational cost makes this a good model to explore.

Strengths:

- Highly scalable

Weaknesses:

- Needs a big data set to make a reliable model (doesn't work so well with small data set)

Gaussian Naive Bayes	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	0.4578	0.7578	0.7822
F1 score for test set	0.2683	0.7419	0.8296

Decision Tree

Decision Tree might be a good fit for this because it performs well with large data set. Once the model has been built, it's pretty intuitive to understand and we can trace back the steps that leads to the result.

Strengths:

- Performs well with large data set.
- Steps that leads to result can be traced back and easily understood (white box model)

Weaknesses:

- There are some problems that cannot be represented very well, and cause decision tree to grow very large. For example: XOR problem.

Decision Tree	Training set size		
	100	200	300
Training time (secs)	0.001	0.002	0.002
Prediction time (secs)	0.000	0.000	0.001
F1 score for training set	0.9	0.8696	0.865
F1 score for test set	0.7031	0.8261	0.831

Choosing the Best Model

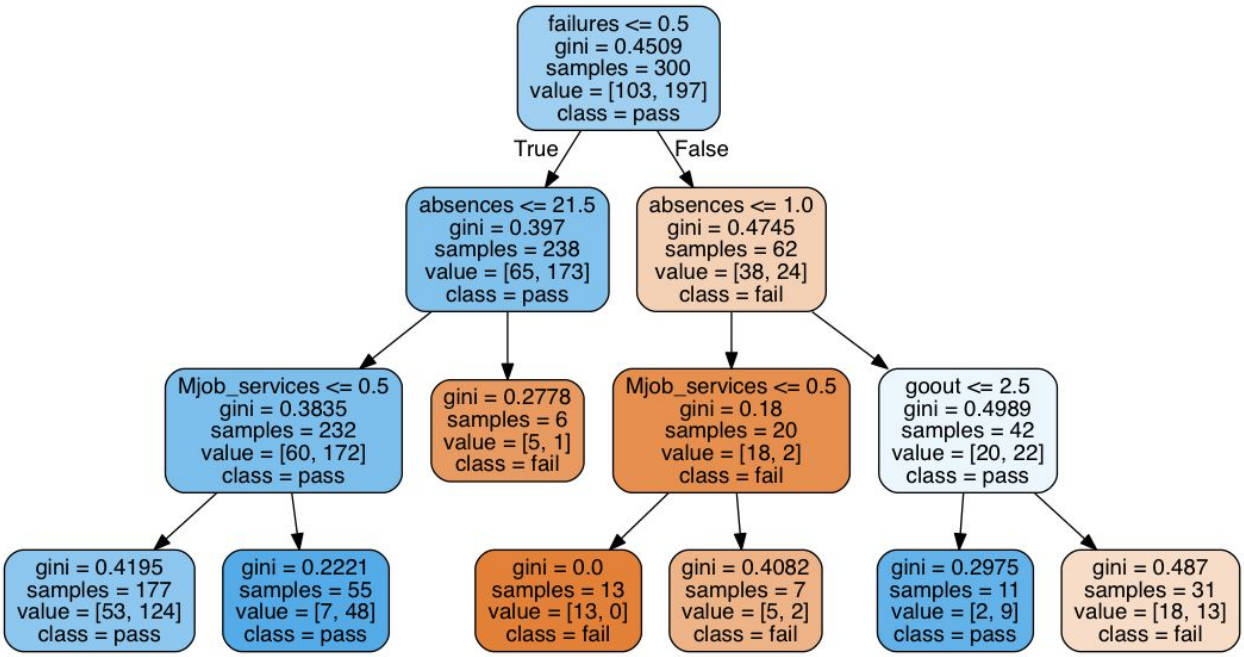
Based on the model evaluation, SVM has the highest overall F1 score for all training size (0.795, 0.85, 0.855). However the training and prediction time for SVM increase rapidly as the size of data set grows. This means the cost of using SVM will increase rapidly as we have more data to analyze and make prediction on.

On the other hand, Gaussian Naive Bayes has the best training and prediction time (a constant 0.001 second for training and almost 0 second for prediction). However it doesn't do so well when the training data set is small. Its F1 score for 100 data is just 0.26.

This makes Decision Tree the preferred choice because its training and prediction time is close enough to that of Gaussian Naive Bayes. Its performance is close enough to that of SVM (with F1 score of 0.703, 0.826, 0.831). With some tuning, Decision Tree's performance can still be improved.

How does Decision Tree work?

Here is the visualization of the decision tree from our model:



Decision Tree uses a few selected features in the data set to break down the data into smaller subsets. Each feature in the decision tree splits the data into 2 sub-trees. Depending on what the answer for a particular data point regarding that feature, you go to one of the sub-trees and answer the question for the feature on that sub-tree, until you reach a leaf node which tells you what the class / prediction result for the data point in question.

For example, starting at the root node (failures <= 0.5), if the student past failure is <= 0.5 (0 times), then we go to the left branch (absences <= 21.5). If the student absences is greater than 21.5, our decision tree predicts that the student will fail. In a sentence we can summarize this as: student who has never failed in the past, but have been absent more than 21 times will fail.

After doing more tuning using Grid Search Cross Validation, our Decision Tree achieves the following result:

Final F1 score for training set: **0.8227**
Final F1 score for test set: **0.8611**