Project 2: Supervised Learning

Classification vs Regression

Our goal is to be able to identify students who are at risk of failing the final exam. We are treating this as a pass-fail outcome. As such, our model will be a classifier with binary output representing whether or not a given student is predicted to pass the final exam.

Exploring the Data

We will be learning from a set of 395 students, each of whom is described by a set of 30 features. From this sample, 265 students passed and 130 failed the final exam for a graduation rate of 67.09%.

Learning Models

Two factors stand out as most significant in the selection of a classifier: first, the premium placed on performance; second, the small size of the dataset on which to learn. The classifier used for this problem must quickly converge without much training data, and the prediction time and memory must be low. After considering the problem at hand and conducting some research, including consulting the sklearn documentation on choosing an estimator, I've decided on three classifiers to explore further:

Decision Trees

A decision tree classifier immediately stood out to me as a worthy candidate for this situation. Decision trees are used in data mining as they are performant, scalable, and suited to both numerical and categorical data while requiring little data preparation. Perhaps most important when considering that this model must be applied in a non-technical environment, decision trees can be visualized and explained to the layman. A student intervention system might be more effective it can be intuitively understood by the educators and other school personnel responsible for its implementation.

However, decision trees have a tendency to overfit without careful tuning. I will therefore also explore the AdaBoost ensemble method using decision trees as the weak learners, though I suspect that, after tuning, the performance cost of using an ensemble method will outweigh the performance gain.

Naïve Bayes

Naïve Bayes classifiers are commonly used in text categorization and spam filtering. They are easy to implement, benefit from strong performance, and can be trained from a small amount of data. However, the "naïve" assumption of pairwise conditional independence between features clearly does not hold in our situation. Nonetheless, Naïve Bayes classifiers consistently perform well even under such circumstances, and I will be further exploring their suitability to this problem.

Support Vector Machines

Support vector machines (SVMs) are commonly used in image classification (such as handwriting recognition) as well as in biological applications (e.g. protein classification). They scale very well with the number of features. This could be an important factor considering we are starting with a dataset with 30 features and might identify, or want to consider, additional features as we (i.e. the school) learn more about the problem. When referring to the sklearn documentation on choosing an estimator, a linear SVC is the first suggested classifier for our situation (labeled data, less than 100,000 samples). In the future, we could incorporate domain knowledge into the model by using a non-linear, possibly custom-engineered kernel.

Inconveniently, an SVM is a black box and would be more difficult to explain to the school. SVMs can also be computationally expensive, but we can control this through the choice of kernel.

Initial Testing

The default sklearn implementation of each of the above algorithms was tested over 100 iterations. The results are summarized below:

Decision Trees (DecisionTreeClassifier)	Training set size		
	100 samples	200 samples	300 samples
Training time (secs)	0.000578	0.000987	0.001421
Prediction time (secs)	0.000143	0.000162	0.000177
F1 score for training set	1.000000	1.000000	1.000000
F1 score for test set	0.734517	0.749701	0.766689

Naïve Bayes (GaussianNB)	Training set size		
	100 samples	200 samples	300 samples
Training time (secs)	0.000598	0.000669	0.000741
Prediction time (secs)	0.000260	0.000317	0.000377
F1 score for training set	0.704344	0.799889	0.803738
F1 score for test set	0.611525	0.780085	0.828571

Support Vector Machines (LinearSVC)	Training set size		
	100 samples	200 samples	300 samples
Training time (secs)	0.006398	0.014132	0.022624
Prediction time (secs)	0.000151	0.000178	0.000233
F1 score for training set	0.899929	0.827942	0.808301
F1 score for test set	0.764254	0.793889	0.809099

Model Selection

After reviewing the data, I've selected a decision tree classifier as the best choice for this situation, for the following reasons:

1. Strong F1 score with small training sets
2. Short training and prediction time
3. Conceptual simplicity

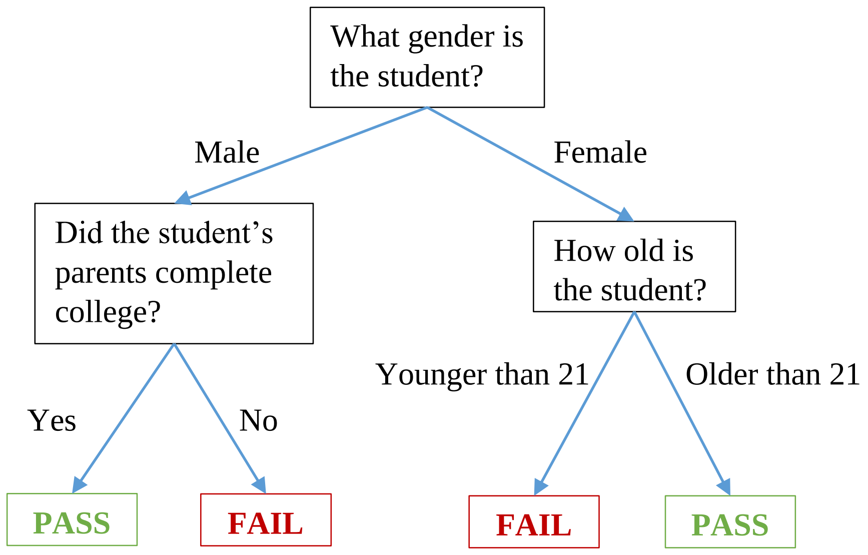
When comparing model performance on training sets of 100 samples, decision trees (F1 score of 0.735) consistently out-perform Naïve Bayes (F1 score of 0.612) and are out-performed by SVMs (F1 score of 0.764). However, SVMs have a training time which is an order of magnitude larger than decision trees (6.40 ms vs 0.58 ms). In fact, decision trees have the shortest training time at 100 samples and the shortest prediction time at any sample size. Also, I believe the F1 score of decision trees will improve significantly after tuning, as the default implementation is clearly overfitting to the training set (perfect F1 score on training set).

Decision trees are also very intuitive. The decisions made at each node can be expressed and understood in human terms (e.g. are the student's parents employed), which will be invaluable in communicating the final model to the school personnel who will be responsible for its application. A teacher can understand that, for example, younger female students who are daughters of unemployed or uneducated parents and commute long distances to school are at higher risk. These are risk factors that can be recognized in a student without needing to resort to a computer. The same cannot be said of the black-box decision surfaces generated by SVMs.

Understanding Decision Trees

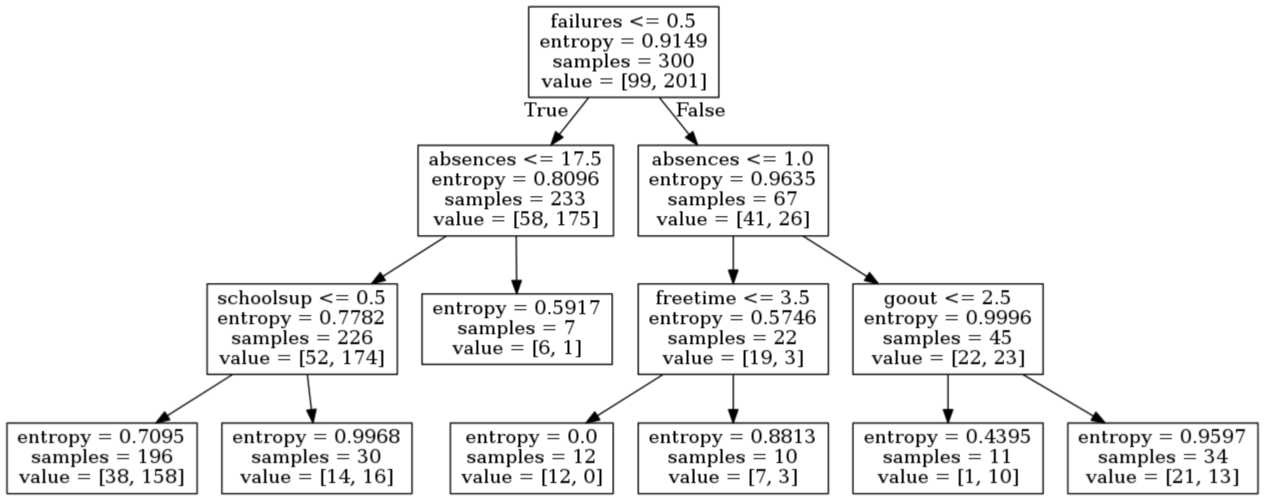
Like all classifiers, decision tree classifiers try to discover a set of rules that can be used to accurately predict the type ("label") of a sample based on its characteristics ("features"). In this case, the features include gender, age, family size, and parents' education. The label is a binary pass/fail prediction.

Decision trees accomplish this classification by asking a series of questions about the features of the data and splitting the data into different groups based on the value each sample has for that feature. For example, we ask "what is the gender of the student" and separate the samples into two groups: male and female. We choose the best question to ask by identifying the feature that, when the data is grouped based on each samples value for that feature, best orders the data into groups that each contain mostly samples of just one label (i.e. each group mostly contains student that passed or it mostly contains students that failed). The process is then repeated on each group until we've identified a series of features on which we can reliably split the data into uniform – or, more realistically, *mostly* uniform – groups of just one label. At this point, given a new sample, we can ask the same series of questions about that sample, assign the sample to one of these groups, and expect that the label for that sample should be the same as the others in the group. An extremely simple example decision tree is given below.



Model Refinement

After tuning, the final model had an F1 score of 0.791 on the testing set. It is shown below:



Looking at the patterns discovered by the classifier, I see many that make intuitive sense. For example, the first feature considered is whether a student has failed a class in the past, and the second feature considered is how many absences a student has. For those who have failed a class before, we also consider amount of free time and time spent going out with friends. Students who have failed a class in the past, have multiple absences, and spend a lot of time out with friends are likely to fail again (>20% of students who failed fall into this category).

Students who haven’t failed a class before but have many absences (at least 18) are also likely to fail, though this sample size is very small (only seven samples). The other significant factor for these students is whether they are receiving additional education support.

Examining the precision and recall scores of the model reveals an “over-optimistic” tendency to label students as passing. In fact, one in every six students labeled as passing actually failed (recall score of 0.733). Given the serious implications of failing, it would be better to err on the side of caution and bias the model to generate more false negatives (students labeled as failed who actually passed) than false positives (students labeled as passed who actually failed).

Though the above model would serve as a great starting point for a student intervention program, it should be refined further with additional data, tuning, and possibly pruning.