Building a Student Intervention System

Classification of Students who show signs of needing intervention
By Ryan Talk

Classification vs. Regression

The goal of this project is to identify students who might need early intervention. Thus, we need to put students into two groups – needs early intervention and does not need early intervention. Therefore, this machine-learning problem falls under classification.

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

Preparing the Data

See Code

Training and Evaluating Models:

The three models that I have decided to explore are Support Vector Machines, Naïve Bayes, and K-Nearest Neighbors.

NOTE: To see a comparison of the Test Results for each Classifier, see the Results section at the end of this document.

Support Vector Machines

Support Vector Machines are generally used for binary classification purposes such as the problem at hand. They typically work well in complicated domains meaning the decision boundaries that decide whether or not a student needs intervention may not be linear. The downside of SVMs is that they will be more costly to train and test. The training time of the SVM is cubic with respect to the number of training points and the testing time is linear with respect to the number of support

vector that come from training. Despite the cost, I chose this model as it is generally used for binary classification and will be able to capture the complexity of our data.

Naïve Bayes Classifier

The Naïve Bayes Classifier is generally used for text classification in documents. This assumption of a Naïve Bayes Classifier is that all the features of our data set are conditionally independent. This gives an inherent bias to the algorithm and may keep it from finding complex relationships between the features if any exist. However, this assumption also provides us with the key advantages to Naïve Bayes: it is cheap (linear training time, linear number of parameters, linear prediction time with respect to data to predict) and it is able to handle noisy data.
The main reason I chose this classifier is that it wonderfully fits our budget constraints.

K-Nearest Neighbors

The K-Nearest Neighbors is generally used for classification purposes. It is a non-parametric lazy-learner. This essentially means that it doesn't make many assumptions about the data and that it puts off learning until it is time to make a prediction. This gives us an advantage that we will have 0 training time. This is a very nice benefit in terms of our budget constraint. However there is a tradeoff in prediction time, as the algorithm requires the entire training set. This makes the algorithm memory and computationally intensive. The assumption that is made with nearest neighbors is that points that are close to each other ought to have similar labels. I chose this algorithm in particular, as it seems to emulate the bias in the problem. I am making an assumption that students in similar circumstances will have similar outcomes. In other words, based on a student's environment and how previous students in that environment have fared, future students are likely to have the same outcome.

Choosing The Best Model

Based on the experiments performed in the Results section, I chose the K-Nearest Neighbors (KNN) algorithm as the best model. Although Naïve Bayes would have been the ideal candidate in terms of cost efficiency, it does not perform well in terms of F1 score. The F1 score is a metric that combines precision and recall and represents how well an algorithm does at computing the correct answer; 1 being the best, 0 being the worst. The Support Vector Machine (SVM) and the KNN algorithms had the best test F1 scores of .77852 and .78832 respectively. Therefore, the decision between these two algorithms comes down to cost. SVM has a cubic training time, which means as we increase the amount of training data the amount of time it takes to train drastically increases. In order to optimize our algorithms, we need to select numerous parameters and compare the F1 Score. Thus, we would

need to train the SVM and KNN classifiers multiple times in order to get the best fit. In this instance KNN wins out as there is essentially 0 training time. To give an illustration to this I attempted tuning an SVM classifier with 11 different parameters and that took roughly 95 seconds. Compare that with KNN using 45 different parameters, it took roughly 3.349 seconds. Prediction time for both models is roughly similar with a slight edge to SVM; SVM prediction time is linear w.r.t. the support vectors, whereas KNN is linear w.r.t. the training data.

In Laymen's terms

How does the KNN algorithm classify whether a student needs intervention or not? The KNN algorithm remembers past instances of students who have passed and not passed their final exams as well as what circumstances those students experienced. From this memory, it then compares a new student to students that it remembers. If it sees a student in similar circumstances to one that has failed in the past, it will call for student intervention, otherwise the student is good to go.

Results:

Results: 300 Training points – Prediction Time is time it took to predict 300 labels.

Classifier	Training Time (s)	Prediction Time (s)	F1 Score Train	F1 Score Test
SVM	.008	.006	.87420	.77852
Naïve Bayes	.001	.001	.78759	.69767
K-NN (k=5)	.001	.007	.87361	.78832

Results: 200 Training points – Prediction Time is time it took to predict 200 labels.

Classifier	Training Time (s)	Prediction Time (s)	F1 Score Train	F1 Score Test
SVM	.004	.003	.86956	.78146
Naïve Bayes	.001	.001	.78853	.68217
K-NN (k=5)	.001	.003	.86379	.79433

Results: 100 Training points – Prediction Time is time it took to predict 100 labels.

Classifier	Training Time (s)	Prediction Time (s)	F1 Score Train	F1 Score Test
SVM	.002	.001	.83333	.77419
Naïve Bayes	.001	.001	.20513	.25352
K-NN (k=5)	.001	.001	.84276	.78666