Building a Student Intervention System

Classification vs. Regression

The student intervention system is fundamentally a classification problem. We are attempting to identify at risk students and classify them as such in order to begin an intervention process. For this problem either the student is at risk or is not. We are not attempting to predict specific outcomes for each student, merely whether or not they are in need of intervention.

Exploring the Data

Some basic statistics of our dataset

- 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: 67.09%

Training and Evaluating Models

The three models tested, in order, are Logistic Regression, SVM, and K Nearest Neighbor.

- **Logistic Regression**

  The general applications for logistic regression (LG) are binary classifications based on the probability of an outcome. In this instance < .5 LG will predict the student will fail and > .5 LG will predict the student will pass.

  Logistic regression is a fast model and good at defining two different classes (again the classification occurs based on the probability of an outcome.) Furthermore, logistic regression is easy to implement.

  Logistic regression, however, is prone to overfitting if it is not properly regularized and choosing the proper regularization value can be tricky (however, most libraries that implement logistic regression automate much of this process.)

  Since logistic regression is very effective in making binary classifications based on the probability of the outcome it is an excellent model to use in order to classify a student as “at risk” and in need of intervention.

	Training Set Size		
	100	200	300
Training Time (secs)	0.002	0.003	0.004
Prediction Time (secs)	0.000	0.001	0.000
F1 Score Training Set	0.8740	0.86689	0.843267
F1 Score Test Set	0.78740	0.80315	0.794118

- **SVM**

  The general applications of the SVM model are the binary classification of labeled data.

  A strength of SVMs is that SVMs are effective when there are a lot of features to take into account in order to allow the model to learn & make accurate predictions. SVMs are also versatile because they make use of kernel functions.

  SVMs, however, ignore outliers, which might be important when it comes to a student intervention system (and might be a weakness in this instance.) Furthermore, SVMs are complex and this complexity has an impact on performance. The complexity of SVMs can lead to issues when attempting to scale SVMs to larger datasets & if there are resource constraints (such as the constraints in computing costs for the intervention system.) As seen in the table below, SVMs also take more time to train and test (*notice the growth of the training time as the dataset increases in size in the table below.*) Another potential weakness of SVMs is the proper use of kernel functions, which gives SVMs their versatility. Kernel functions are a sufficiently complex topic that they will not be broached in this document.

  This model was chosen because it prioritizes correct classification above all else. Given the desire to correctly classify students in need of intervention, the SVM model seemed to be a viable candidate.

	Training Set Size		
	100	200	300
Training Time (secs)	0.002	0.003	0.006
Prediction Time (secs)	0.001	0.001	0.002
F1 Score Training Set	0.84768	0.8690	0.87212
F1 Score Test Set	0.76056	0.76712	0.73611

- **K Nearest Neighbors**

  The general applications for *k* nearest neighbor (KNN) are classification and regression.

  A strength of KNN is that it is a simple algorithm, which can put less demand on resources at least during the learning phase.

  Picking the *k* in KNN can be tricky. Using the correct *k* (or the number of nearest data points to compare to) can be a somewhat difficult process and using the incorrect *k* can lead to poor predictions. KNN also assumes that near points are similar; this might not be necessarily true and may lead to incorrect classifications. KNN also requires using the correct distance function for a given problem, not all distance functions will work well in various domains and choosing the wrong one can a have negative impact on classification. Finally, even though KNN is a simple model it is a "lazy learner" (which means it defers computations until it absolutely has to), this ends up having a high cost in terms of computation time when making predictions.

  This model was chosen in order to see whether a simple classification model was effective enough to on the data. Given the constraints, using a simple model would be highly advantageous when scaling to larger datasets.

	Training Set Size		
	100	200	300
Training Time (secs)	0.001	0.001	0.001
Prediction Time (secs)	0.002	0.002	0.005
F1 Score Training Set	0.78049	0.81595	0.824268
F1 Score Test Set	0.76510	0.741259	0.732394

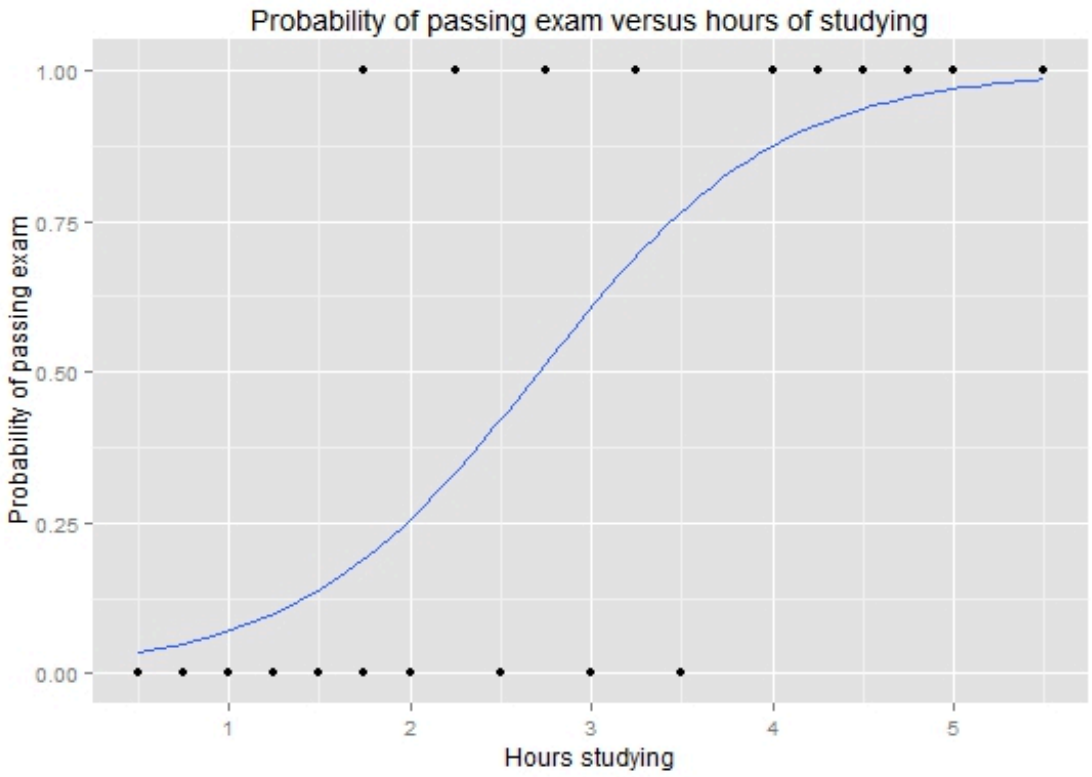
Best Model

The best model given the constraints is Logistic Regression. The chosen model has the highest F1 score (LG F1 score of ~.79 vs SVM of ~.74 and KNN of

~.73) while simultaneously having the lowest training/prediction times. LG's training/prediction times on the whole training set were a combined .004 seconds, while SVM's were a combined .008 seconds and KNN's were a combined .006 seconds.

As briefly discussed above in the section "Training and Evaluating Models", LG makes binary classifications (yes or no) based on the probability of an outcome. What does this mean? Simply, the LG model will look at absences, age, travel times, study times, etc. and give a probability of the student passing or failing. So, if the probability is greater than .5, LG will predict that the student will pass and if the probability is less than .5, LG will predict that student will not pass.

A good example from Wikipedia that is relevant to this topic is displayed in the image below:



source: https://en.wikipedia.org/wiki/Logistic_regression - Example:_Probability_of_passing_an_exam_versus_hours_of_study

What we see in the above image is the probability curve of the likelihood of a group of students passing a test given the number of hours the students study. Every data point that is greater that .5 on the blue curve will be given a value of 1.00. While every point that is less than .5 on the same curve will be

given a value of 0.00. This is the binary classification referenced in the “Training and Evaluation” section.

When the LG model is learning it will compare it’s prediction of pass or fail with the actual result from the dataset it’s learning from. LG does this in order to find the best way to predict whether a student will pass or fail. So LG will go through the data constantly comparing it’s predictions with the actual results and make adjustments until the best method of predicting is found.

After the learning phase is complete we test the method the LG model found against a small portion of the data that was removed from the dataset before we began the learning process. The purpose of this is to make sure that we haven’t manipulated the model to return results that make the model seem accurate (more simply, we haven’t fixed the predictions to agree with the real results.) This works by passing this portion of the data through the same method, looking at absences, age, travel times, etc. and then giving the probability of whether or not the students represented in this portion of the data pass or fail. If this test produces a reasonable accuracy score we can begin the process of introducing new data and using the method the LG model found to make real world predictions.