Student Intervention System

Author: Jonathan Lee

Table of Contents

Classification vs Regression

Trying to determine which students may need early intervention is a classification problem because the output that we are trying to predict is a discrete value of either pass or fail. Regression is used in a situation where there is be a continuous output which is not the case in this current study.

Exploring the Data

Query	**Result**
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 (excluding the label/target column)	30

Preparing the Data

In this arena, the features and target columns were separated. The features were separated and named into X_all. The target variable was named y_all. The feature columns with multiple outcomes were also split into individual columns called dummy variables and their value is either one or zero. Also, any yes/no string has been converted to a 1 or 0 integer. Then finally, this data was randomized and split into training and test sets.

Training and Evaluating Models

In this section, three different supervised classification models were chosen: Support Vector Machine, Decision Tree Classifier, and Naive Bayes. Let's dive into each model.

Support Vector Machine

SVMs can be used for both classification and regression applications. Some classification applications include text categorization, image classification, and handwriting recognition. Some advantages of SVM include good performance in high dimensional spaces where there is a clear margin between datasets. Also, a kernal trick can be applied to help create a larger margin of separation between data. It can also give SVMs the ability to fit non-linear data. Some disadvantages is that the performance becomes degraded in applications that contain large number of features and dataset. Also do not perform well when the two classes begin to overlap.

As one already knows that this system in a classification problem since the outcome has two classes: passed == ‘yes’ or passed == ‘no’. Therefore, by using SVM algorithm, a hyperplane is drawn in such a way that maximizes the margin between the points (called support vectors) in both the ‘yes’ or ‘no’ zone. To make a prediction, we simply input the features of a student which we are trying to predict and whichever side of the hyperplane this student will land on is the side that is chosen.

Test Results

	Sample Size		
	300	**200**	**100**
F1 score Test	0.8535031847	0.8427672956	0.8571428571
training time	0.009347	0.002868	0.005019
prediction time (train)	0.014756	0.005701	0.00113
prediction time (test)	0.00144	0.001955	0.000971
total time	0.025543	0.010524	0.00712

Decision Tree Classifier

Decision trees can also be used for both classification and regression. There are countless applications which could employ decision trees. Some simple examples could include, predicting which customers to send credit card applications too, which customers can be added to the insurance policy with what price, or even which customers may qualify for a student/business loan. In the previous project, Decision Trees were used for regression. In this project, we shall utilize the classification aspect of Decision Trees. Some advantages of decision trees is due it it's simplicity and ease of use. They are very easy to understand. The main disadvantage is that they are susceptible to overfitting when there are a large amount of features.

Since this current study is a classification problem, it seems that using Decision Tree Classifier is another obvious choice. Again, we attempt to classify the students into two classes either pass or fail. In this decision tree algorithm, it will attempt to ask the right questions which will maximize the informational gain until we break the data into two separate classes.

Test Results

	Sample Size		
	300	**200**	**100**
F1 score Test	0.776119403	0.7669172932	0.7462686567
training time	0.005188	0.001442	0.000813
prediction time (train)	0.000752	0.00024	0.000091
prediction time (test)	0.000333	0.000208	0.000077
total time	0.006273	0.00189	0.000981

Naive Bayes

This model is famously chosen to be used in spam filtering by classifying email as either spam or not spam. It is also used in text classification. Some advantages of this model are that it only a small amount of data to train and the model is simple and very fast. Some disadvantages include the fact that it is oversimplifying the data in the “naive” assumption that all the features are independent of each other. Also, it is a bad estimator.

Since spam filtering is also a classification problem, it seems that this is another obvious choice to see if this model can help classify if students pass or fail. Also, since naive bayes classifier’s speed and efficiency makes this a great choice for an application where resources are limited. This algorithm will look at all the features and attempt to classify based on the conditions of the input features whether the probability will be higher for the student to pass or not pass and classify accordingly.

Test Results

	Sample Size		
	300	**200**	**100**
F1 score Test	0.7121212121	0.7384615385	0.768115942
training time	0.001262	0.002961	0.000748
prediction time (train)	0.000361	0.002136	0.00019
prediction time (test)	0.000173	0.003387	0.000337
total time	0.001796	0.008484	0.001275

Choosing the Best Model

We have been rating the models based on the F1 score, which is the harmonic mean of precision and recall. The higher the F1 score signifies that both precision and recall should be both high and relatively close with each other. A model high precision and low recall or low precision and high recall will generate low F1 scores, in addition to, a model with both low precision and low recall. Only a model with high precision and high recall can generate high F1 scores. SVM provides the highest F1 score around 0.85. Decision tree comes in second with around 0.776. However, Naive Bayes performed the worse with 0.71.

With regards, to training and prediction time, SVM performs the worse with 0.025543 seconds (training + prediction of training set + prediction of testing set). Decision Tree comes in second with a total time at 0.006273 seconds. Naive bayes is the fastest with 0.001796.

Therefore, for low budget and limited resource point of view, the best tradeoff is to use the decision tree. The F1 score is not the best but it is between SVM and Naive bayes. The training and prediction time combination is also not the best but is between SVM and Naive bayes. Therefore to get the best of both worlds, I believe we should choose the model that has decent training and prediction time and performance which happens to be a decision tree classifier.   Also, as we'll find out later, we can tune this decision tree very nicely to achieve good performance similar to SVM.

The decision tree algorithm focuses on asking the best questions aimed to split the data into separate classes. It is like playing the game called twenty questions. We attempt to ask a series of questions in a binary search format until we get our desired answer. The first question asked would ideally narrow down the possibilities by half. If we are able to narrow down the possibilities by half after each question, eventually we'll pinpoint the answer. For example, in the data set of students we want to ask questions that can help split the data into two groups (a group of students that passed and a group of students that failed). A bad question to ask would be a question that does not help narrow down the data. In fact, the worse question to ask would be a question that does not change the dataset at all. The best question to ask would be a question that perfectly separates the pass and fail students. This algorithm will automatically find the best question to ask at each stage of the decision tree. The training will end when the data has been perfectly split into two classes.

Conclusion

By using GridSearchCV, we are able to tune the model to get the following result for the test set:

F1_score = 0.818791946309

Here, the max_depth sweep is set to 8 and the min_split is set to about 20. This is necessary to prevent overfitting of the training data. When the training data is fit to give F1_score = 1.0, then it did not generalize well with data.

The result is definitely better than the F1_score = 0.776119403, which we were getting earlier on the untuned decision tree. The performance is also relatively close to the F1 score of the SVM, which happened to be 0.8535031847.