Building a Student Intervention System

Supervised Learning Project Open the template iPython notebook student_intervention.ipynb and follow along.

Project brief

As education has grown to rely more and more on technology, more and more data is available for examination and prediction. Logs of student activities, grades, interactions with teachers and fellow students, and more are now captured through learning management systems like Canvas and Edmodo and available in real time. This is especially true for online classrooms, which are becoming more and more popular even at the middle and high school levels.

Within all levels of education, there exists a push to help increase the likelihood of student success without watering down the education or engaging in behaviors that raise the likelihood of passing metrics without improving the actual underlying learning. Graduation rates are often the criteria of choice for this, and educators and administrators are after new ways to predict success and failure early enough to stage effective interventions, as well as to identify the effectiveness of different interventions.

Toward that end, your goal as a software engineer hired by the local school district is to model the factors that predict how likely a student is to pass their high school final exam. The school district has a goal to reach a 95% graduation rate by the end of the decade by identifying students who need intervention before they drop out of school. You being a clever engineer decide to implement a student intervention system using concepts you learned from supervised machine learning. Instead of buying expensive servers or implementing new data models from the ground up, you reach out to a 3rd party company who can provide you the necessary software libraries and servers to run your software.

However, with limited resources and budgets, the board of supervisors wants you to find the most effective model with the least amount of computation costs (you pay the company by the memory and CPU time you use on their servers). In order to build the intervention software, you first will need to analyze the dataset on students' performance. Your goal is to choose and develop a model that will predict the likelihood that a given student will pass, thus helping diagnose whether or not an intervention is necessary. Your model must be developed based on a subset of the data that we provide to you, and it will be tested against a subset of the data that is kept hidden from the learning algorithm, in order to test the model's effectiveness on data outside the training set.

Your model will be evaluated on three factors:

- Its F1 score, summarizing the number of correct positives and correct negatives out of all possible cases. In other words, how well does the model differentiate likely passes from failures?
- The size of the training set, preferring smaller training sets over larger ones. That is, how much data does the model need to make a reasonable prediction?

- The computation resources to make a reliable prediction. How much time and memory is required to correctly identify students that need intervention?

Deliverables

1. Classification vs Regression

This is a classification problem. We are interested in the identifying students who are likely to fail so we can intervene with the necessary help. Thus the model would need to predict whether a student would pass or fail the high school final exam, which a classifier algorithm would be most suitable for.

2. Exploring the Data

- 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: 67.00%

3. Preparing the Data

```
Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob',
'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup',
'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic',
'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']

Target column: 'passed'
```

See Appendix for feature descriptions

4. Training and Evaluating Models

The following were the three supervised learning models chosen for this problem: Gaussian Naive Bayes, Random Forest Classifier, Support Vector Machine.

Gaussian Naive Bayes

A Naive Bayes learner has a space complexity of O(dc) where d is the number of attributes and c is the number of classes, and a training time complexity of O(nd+cd) where n is the number of instances. Since training is linearly more complex for this algorithm, I expect the training and prediction times to be very low for this small dataset.This algorithm is widely used in classification problems such as spam detection, content aggregation and facial recognition. It's strengths include its learning speed, simplicity and independence from dimensionality. Some of it's weakness include its poor performance if independence assumptions do not hold and has difficulty with zero-frequency values. Gaussian Naive Bayes was chosen for it's strengths to classify data quickly,

which is desired in this scenario to save on computation time.

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	0.400	0.811	0.786
F1 score for test set	0.396	0.750	0.791

While training and prediction times are low, F1 scores appear to increase with an increase in training size.

Random Forest Classifier

A Random Forest has a space complexity of O(√f N log N) with f is the number of features and N the number of elements in the dataset, under the assumption that a reasonably symmetric tree is built and that the evaluation of a single test takes constant time in the size of the dataset. The training complexity is given as O(M √f N log N), where M denotes the number of trees. The training complexity is greater than the prediction by a factor of `M`, such that training time would be ten times (the numner of default trees in the algorithm) that of prediction time.

Random Forest learners have been implemented in numerous data mining applications in fields from agriculture, genetics, medicine, physics to text processing - even the Xbox Kinect. Random forest strength is that it can scale well as runtimes are quite fast, and they are able to deal with unbalanced and missing data. Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy. This algorithm was chosen as it is an ensemble of decision tree classifiers, which might suit the dataset well given that majority of the features appear to be mutually exclusive from each other.

Training set size	100	200	300
Training time (secs)	0.006	0.007	0.010
Prediction time (secs)	0.001	0.001	0.001
F1 score for training set	0.976	0.992	0.997
F1 score for test set	0.744	0.721	0.785

The classifier was not tuned and default parameters used, but it can be seen that the learner is overfitting on the training dataset with high F1 scores on the training set.

Support Vector Machine

Support Vector Machine (SVM) has a space complexity of O(n^2) and training time of O(n^3) where n is the training dataset size. The training time is longer for this learner compared to the prediction time in a polynomical fashion, such that the prediction time is shorter by a factor of `n`. This learning algorithm is often used in image recognition, text processing, and bioinformatics classification. Some of SVM's strengths are that it works by employing kernels and on the basis on hinge loss that enable it to learn non-linear decision boundaries by finding the maximum margin, or hyperplane, in

the data. It's downfall include how memory intensive its learning can be, how prone it is to overfitting if given too many features, and how it can only be used in classification problems effectively. SVM was chosen as a learner for this problem given that a feature length of 30 meant that the decision boundary was likely nonlinear, which would make the use of kernels very effective.

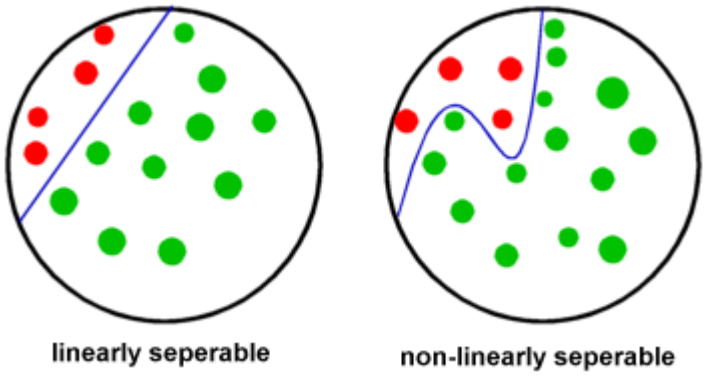
Training set size	100	200	300
Training time (secs)	0.001	0.003	0.007
Prediction time (secs)	0.001	0.001	0.002
F1 score for training set	0.882	0.881	0.846
F1 score for test set	0.775	0.784	0.833

As the training set size increases the training and prediction time understandably increases, but so does the F1 score for the test set.

5. Choosing the Best Model

Based on the constraints provided, the best model to chose from is the Support Vector Machine (SVM). The other learners tested performed poorly with increasing training set sizes compared to SVM. Although SVM has a costly training time complexity where the training time essentially doubles with each additional hundred dataset, it has a high performance that can handle large datasets well that are also unbalanced. Given the ambitious goal of reaching a 95% graduation rate, this processing time is a small price to pay.

Support Vector Machines are based on the concept of decision lines that define decision boundaries. A decision line is one that separates between different sets of objects. In other words, given labeled training data as is in this supervised learning case, the algorithm outputs a clear divide that categorizes new examples. SVM chooses the best decision line or divide where the distance between that line and the nearest observations of differing classes are the largest. This is formally known as achieving the largest margin. The left circle in the following image illustrates this well.



Furthermore, SVMs can employ the use of kernels to fit the data in a higher dimensional space to convert a linear classifier into a more complex nonlinear decision line. Think of this decision boundary as being oddly shaped much like the borders of West Virginia in the US as opposed to being a straight line like the y-axis separating negative and positive x values on an x vs y graph. The

right circle in the above image illustrates this non-linear decision line. The use of kernels lends the model the necessary flexibility to learn from complex datasets and enhance its predictive outcomes.

The chosen SVM model was tuned using Grid Search and Stratified Shuffle Split because the data set is small and unbalanced. Also, in such a case where the data is unbalanced, an F1 score is a better metric than accuracy. The parameters optimized were `gamma` and `C`. I attempted to use optimize over a more exhaustive grid with multiple kernels, but due to timeout issues with ipython notebook, I stuck with the two aforementioned parameters. Nevertheless, satisfactory F1 score of 0.791 on the test sets where obtained, which was better than the default model.

```
Training time (secs): 0.012

Prediction time (secs): 0.004
F1 score for training set: 0.86320754717

Prediction time (secs): 0.001
F1 score for test set: 0.851063829787
```

References

- Naive Bayes Notes
- Random Forest Based Feature Induction
- Random Forest Decision Tree Application
- A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System
- Supoprt Vector Machine
- Support Vector Machines Notes

Appendix

Attributes for student-data.csv:

- school - student's school (binary: "GP" or "MS")
- sex - student's sex (binary: "F" - female or "M" - male)
- age - student's age (numeric: from 15 to 22)
- address - student's home address type (binary: "U" - urban or "R" - rural)
- famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
- Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
- Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
- Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or

police), "at_home" or "other")
- reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
- guardian - student's guardian (nominal: "mother", "father" or "other")
- traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- failures - number of past class failures (numeric: n if 1<=n<3, else 4)
- schoolsup - extra educational support (binary: yes or no)
- famsup - family educational support (binary: yes or no)
- paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- activities - extra-curricular activities (binary: yes or no)
- nursery - attended nursery school (binary: yes or no)
- higher - wants to take higher education (binary: yes or no)
- internet - Internet access at home (binary: yes or no)
- romantic - with a romantic relationship (binary: yes or no)
- famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- freetime - free time after school (numeric: from 1 - very low to 5 - very high)
- goout - going out with friends (numeric: from 1 - very low to 5 - very high)
- Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- health - current health status (numeric: from 1 - very bad to 5 - very good)
- absences - number of school absences (numeric: from 0 to 93)
- passed - did the student pass the final exam (binary: yes or no)