Building a Student Intervention System

Nubyra Ahmed

April 19, 2016

1 Background

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.

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.

The goal of this project is to model the factors that predict how likely students are to pass their high school final exam. Due to limited resources and budgets, we need to find the most effective model with the least amount of computation costs

2 Regression Vs Classification

The goal is to predict whether a student will pass or fail their high school final exam. The response variable, whether a student will pass or not is binary, in other words it is categorical. Therefore, this is a classification problem. We need to identify to which category a new observation belongs, on the basis of a training dataset containing observations whose category membership is known.

3 Exploring the Data

Below is an overall summary of the data. The appendix section contains information about the features.

1. Total number of students: 395
2. Number of students who passed: 265
3. Number of students who failed: 130

4. Number of features: 30
5. Graduation rate of the class: 67.09%

There are two important issues to note with this dataset. First, the number of observations, relative to the number of features is low. This may result in overfitting our models and unable to generalize, leading to poor predictions about new or unseen data. Second, the data is unbalanced. There are more instances of students who have passed the final exam than students who did not pass. Imbalanced class distributions are a very important problem from both the algorithmic and performance perspective.

Generally, the number of observations needed to accurately predict whether a student would pass or not pass increases exponentially as the number of features increases. This is known as the “curse of dimensionality”. One consequence of this phenomenon is overfitting, where a model is too specific to the training data and unable to generalize to unseen or new data. Since acquiring more observations is not feasible, the issue of overfitting will be tackled through the application of cross validation method. The aim is to choose a simpler and less complex model, over more complicated models, without significantly increasing the out-of-sample prediction error. A standard way of finding the out-of-sample prediction error is to use K-fold cross validation. This is a better representative of the error one would expect when predicting a future response value, rather than just how well a model fits the data at hand.

Accuracy as a performance metric is not an useful measure when evaluating classifiers learned on imbalanced datasets. Assuming, class label distribution for both training and test data is the same, accuracy metric maybe more biased toward the majority class, and have decent accuracy just by predicting everything as “passed”. Therefore, F1 measure is chosen as an alternative metric that is more robust to imbalanced class distribution. The main goal is to improve the recall without hurting the precision. The F1 metric combines the trade-offs of precision and recall, and outputs a single number reflecting the robustness of the learning algorithm, in the presence of minority class.

Some learning algorithms are more sensitive to imbalanced class distributions, such as the logistic regression and the Support Vector Machines. There are measures that can be taken to make these models more robust, or one can choose algorithms that are naturally less sensitive to imbalanced datasets, such as tree-based or ensemble-based methods. For the problem at hand the latter approach is chosen as an alternative way of dealing with model robustness.

Another consideration that must be taken into account when dealing with imbalanced dataset is how the data is split into training and test sets. Simply splitting the data, without any regards to the class imbalance may lead to very different results. In an extreme case one may end up with a training dataset that consist of observations with students who passed the final exam and test dataset with observations consisting of students who failed. This may result in poor performance, since the distributions of the two datasets are polar opposite. In order to tackle this issue, data is split using the stratified shuffle split method. This method ensures that the train and test set contains approximately equal ratio of each class and simultaneously remove any bias that maybe present due to the ordering of the observations.

Note that there are many other techniques that could be applied to deal with imbalanced dataset. Above are only few of such approaches that were applied to tackle this issue.

4 Training and Evaluating Models

As discussed in the previous section the dataset at hand consist of imbalanced class distribution. Keeping this in mind, we have decided to apply tree-based and ensemble-based models that are naturally less sensitive to class imbalance. The three models trained and evaluated using F1 scores are Decision Trees, Random Forest, and Gradient Boosting Machine.

4.1 Decision Trees

Decision Trees (DTs) are non-parametric supervised learning method used for classification and regression. It has a flow chart like structure, where a node represents test on a feature, a branch represents an outcome of the test, and a leaf represents a prediction to the question. The path from root to leaves represents the decision on an observation.

Some advantages of DTs other than being able to handle imbalanced datasets are that they are simple to understand and prediction procedure is transparent. They can handle both numerical and categorical data, including missing and sparse data. Some disadvantages of decision trees are that they are notoriously noisy and tend to create complex trees that do not generalize the data well. They tend to be unstable. Any small change in the variation of a dataset may result in a completely different tree.

Assuming a balanced binary tree is constructed, training time cost is $O(nd \log n)$ and the prediction time is logarithmic in the number of data points used to train the tree, that is $O(\log n)$, where n is the number of observations and d is the number of features. The depth is $O(\log n)$, therefore space complexity is $O(\log n)$ also.

For the student intervention dataset, algorithm for DT was applied using python's scikit-learn module with default parameter setting:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=0, splitter='best')
```

Following are the F1 scores and time complexity for different training set sizes and corresponding results for the test dataset:

	Training Set Size		
	100	200	300
Training time (secs)	0.001	0.002	0.002
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	1.000	1.000	1.000
F1 score for test set	0.516	0.661	0.610

As expected, training and prediction time is very fast across all training sets. Also, not surprisingly, under the default setting, due to not setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree to mitigate the issue of overfitting, all training datasets were overfitted, indicated by their perfect F1 score. This has resulted in poor prediction on the test set with F1 score of about 66% being the highest for sample size 200.

4.2 Random Forest

Random Forest (RF) can be used for both regression and classification problems. It is a type of ensemble learning that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control overfitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement. Also, in order to reduce noise, each tree is constructed by using a randomly chosen subset of the original number of features. Random Forest is also helpful for identifying which features are important in the underlying data being modeled. Random Forests are reasonably fast to train, but prediction time can be slow. To construct more accurate ensembles more trees are need. Also, results of learning are harder to explain and glean insight, compared to a single decision tree.

Assuming that approximately symmetric trees are built, a RT has a space complexity of $O(\sqrt{d}n \log n)$, where d is the number of features and n is the number of observations. The training complexity is $O(N_{tree}\sqrt{d}n \log n)$, where N_{tree} denotes the number of trees built. The training complexity is greater than the prediction time by a factor of N_{tree}.

F1 scores and time complexity for different training set sizes and their results for the test dataset are obtained using scikit-learn module's default parameter setting.

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

	Training Set Size		
	100	200	300
Training time (secs)	0.019	0.015	0.015
Prediction time (secs)	0.001	0.001	0.001
F1 score for training set	1.000	0.993	0.998
F1 score for test set	0.678	0.727	0.744

Compared to training a single DT, RF takes much longer, but prediction accuracy is much larger. Under the default setting, as training size increases F1 score decreases slightly, with an increase in prediction score, suggesting that RF maybe able to mitigate the issue of overfitting. The highest F1 score for test set is 74.4% for training size 300.

4.3 Gradient Boosting Machine

Gradient Boosting Machine (GBM) can be used for both regression and classification problems. GBM builds decision trees in a stage-wise manner, where later trees use the residual from the earlier trees to reduce prediction error. The final model is a linear combination of the individual tree models, where weight given to each individual tree is determined through a learning rate. The issue of overfitting is addressed through the number of trees that are grown iteratively, depth size, minimum number of observations required at a leaf node, and learning rate. To derive an optimal model and avoid overfitting, GBM requires tuning of additional parameters such as the learning rate. Due to its iterative nature training time can be longer than other ensemble methods such as Random Forest, which is traditionally known to be easier to parallelize. However, recent advances suggest that there are efficient ways of training GBMs, such as XGBoost. Note that XGBoost was not applied for this project.

GBM has time complexity of $O(N_{tree} dn \log n)$, with prediction time of $O(dn \log n)$ and space complexity of $O(dn \log n)$. F1 scores and time complexity for different training set sizes and their corresponding test predictions are obtained using scikit-learn module's default parameter setting.

```
GradientBoostingClassifier(init=None, learning_rate=0.1, loss='deviance',
                max_depth=3, max_features=None, max_leaf_nodes=None,
                min_samples_leaf=1, min_samples_split=2,
                min_weight_fraction_leaf=0.0, n_estimators=100,
                presort='auto', random_state=0, subsample=1.0, verbose=0,
                warm_start=False)
```

	Training Set Size		
	100	200	300
Training time (secs)	0.050	0.065	0.083
Prediction time (secs)	0.000	0.001	0.001
F1 score for training set	1.000	1.000	0.967
F1 score for test set	0.650	0.688	0.742

Compared to RF, GBM takes longer to train as training size increases. Under the default setting, without any parameter tuning, overfitting is still an issue as evidenced by the F1 scores for the training datasets. Also, prediction performance on test data is approximately the same as Random Forest, with highest F1 score of 74.2% is achieved for training size of 300.

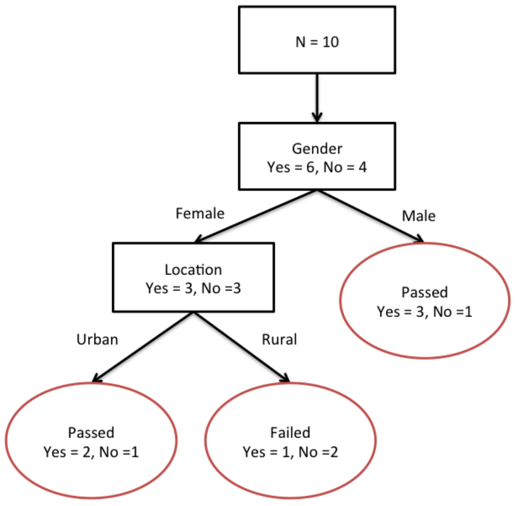
5 Choosing the Best Model

The best model, in terms of performance metric is RF and GBM with nearly the same highest F1 score of approximately 74.0%, while DT performed the worse, with highest F1 score of 66.0%. Both RF and GBM achieved relative performance gain of approximately 12.0% over DT. In terms of time complexity, DT is the fastest. For a training size of 300, DT is approximately 7 times faster than RF and 40 times faster than GBM, while RF is approximately 5 times faster than GBM. For all three models, prediction time is approximately the same. Keeping in mind that we have limited available data and resources, in terms of cost and performance a reasonable balance is achieved using Random Forest, which provides a performance gain of 12% over DT and is reasonably fast to train, with prediction time being approximately the same. Below is a simple explanation of Random Forest.

A decision tree is a set of rules used to classify data into categories. In our case, we want to classify students as either they will pass or fail their final exam. DT looks at the factors in a data set, determines which are most important, and then comes up with a set of rules or questions which best partitions the data. The tree is created by splitting data up by variables and then counting to see how many are in each bucket after a split. For a bucket after a split on a variable, lets say there are M observation, if all the M observations belong to the same class or M is very small, than this bucket becomes an endpoint, labeled with the most frequent class. If M for a bucket is too large and it contains more than one class, then we would find the "best" feature to split on based on the observations in that bucket. This process is continued until all the observations are split into buckets that are very small or entirely one class, than we have generated a set of rules that look like a tree. Lets take a look at a simple dataset with N = 10 observations and two features (gender and location). The response is whether students passed or did not pass their final exam. The sample dataset is show below:

passed	gender	location
yes	female	urban
no	female	rural
yes	male	urban
no	male	urban
yes	female	rural
no	female	rural
yes	male	urban
yes	male	urban
yes	female	urban
no	female	urban

Below is a Decision Tree generated from the sample dataset:



The 10 observations are split on gender resulting in two buckets, namely male and female. If observations are male than they are predicted to pass since "yes" response is the majority class. For the female bucket there are 3 passed and 3 not passed, This bucket is split further on location, resulting in two new buckets, namely rural or urban. If the observations are rural than they are predicted to fail, since majority class is "no". If the observations are urban than, prediction is pass, since majority class is "yes". To sum up, this DT predicts that if a student is mail they are likely to pass. If they are female and are from an urban area they are likely to pass.

But our actual dataset at hand has a lot of features, not just two features. We don't know which set of features are the best features to build a DT on that will provide us with the most accurate prediction of whether a student will pass or fail their final exam. So our solution is Random Forest where lots of DTs are built. This is achieved by taking a random set of features from all the available features and a random sample of the dataset at hand and a DT is built. This process is repeated many times using a different random set of features and a random sample of data each time. Once many DTs are built, a prediction for a given observation is made from each DT. The final prediction for a an observation is made based on a majority vote. For instance, if we build 1000 DTs, and 700 of the DTs predict that a student will pass and 300 DTs predict otherwise, than based on majority vote, the final prediction is that a student will pass.

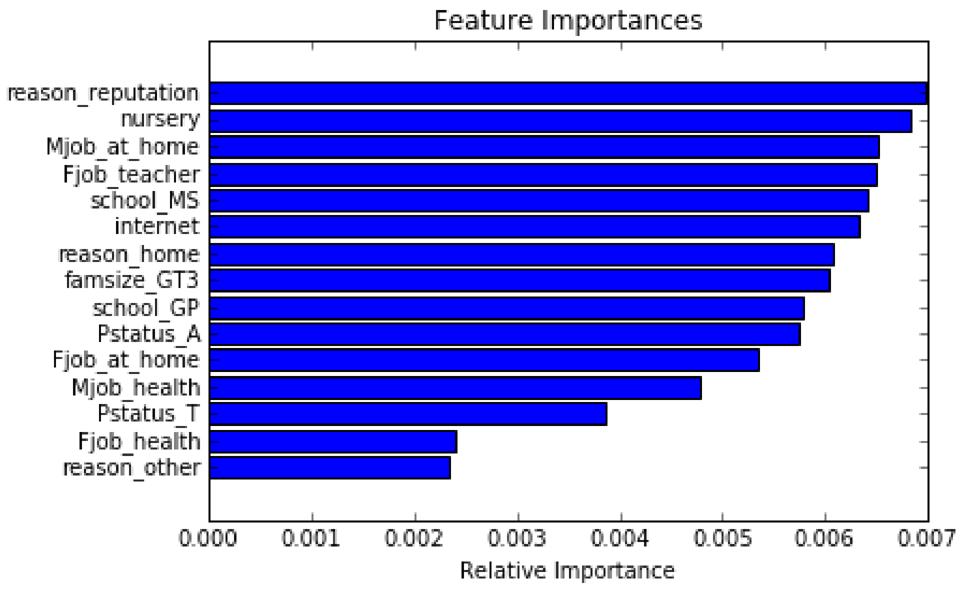
In order to avoid overfitting and find an optimal model, Random Forest is fit to the full training data with various combination of parameter settings. The following parameters with their respective indicated range is applied to find the optimal model:

1. Max Depth Size: [1, 2, 3, 4, 5]
2. Minimum Samples per leaf: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
3. Number of Trees: [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]

The RandomizedSearchCV procedure in the scikit-learn module is applied to fit the various parameter settings. In order to avoid overfitting, the parameters of the estimator used to apply RF are optimized by 3-fold cross-validated search over parameter settings. In contrast to GridSearchCV, not all parameter values are applied, rather a fixed number of parameter settings is sampled from the specified distributions. This approach is taken due to the amount of time required to fit all possible combination of parameter settings, in our case this is 650. The number of parameter settings to fit to the training data is set to 20. The best estimator derived via the randomized search approach is:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=3, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=700, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

The F1 score for train data is 82.7% and test data is 80.0%. Following is a bar chart of the top 15 important features identified by the optimal Random Forest model:



Appendices

Attributes for student intervention dataset:

- 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 - a 5th to 9th grade, 3 - a secondary education or 4- a higher education)
- Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - a 5th to 9th grade, 3 - a secondary education or 4 - a 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 - less than 15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - greater than 1 hour)
- studytime - weekly study time (numeric: 1 - less than 2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - greater 10 hours)
- failures - number of past class failures (numeric: n if between 1 and 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)