# Student Intervention

Release 0.2

Russell Nakamura

March 19, 2016

## Contents

The repository for the source of this document is here.

This is Project Two from Udacity's Machine Learning Nanodegree program. It uses the UCI Student Performance data-set to model the factors that predict the likelihood that a student will pass his or her final exam. The project begins with an exploration of the data to understand the feature and target variables, followed by the selection of three machine-learning algorithms to potentially model the student-performance. Finally one of the models is chosen and the optimal parameters discovered using an exhaustive grid-search.

## 1 Classification vs Regression

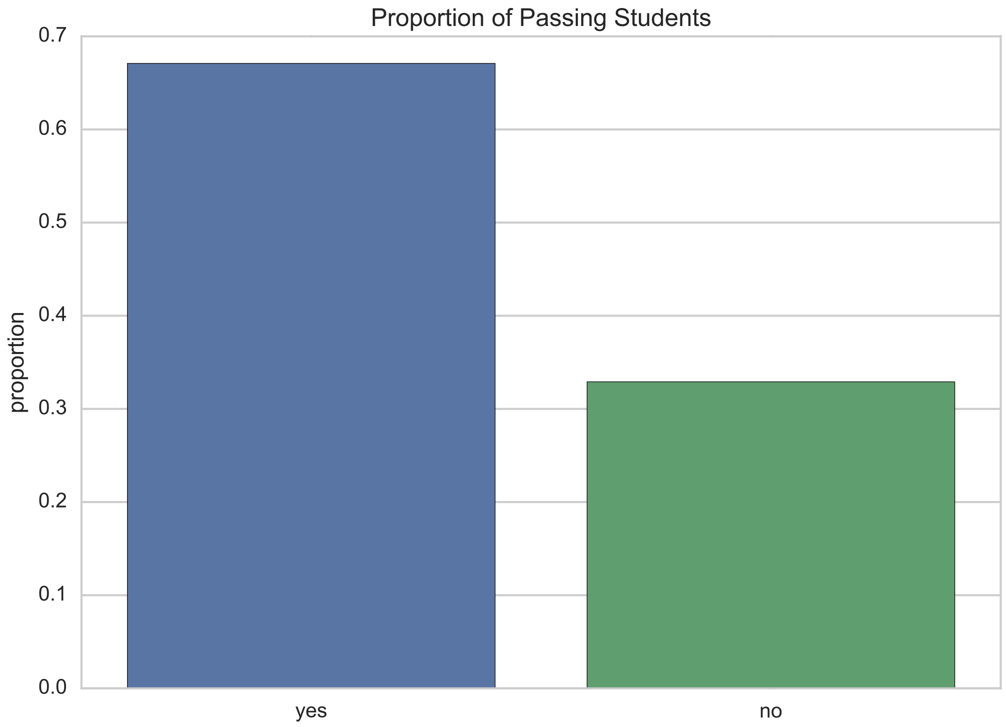
Identifying students who might need early intervention is a *classification* problem as you are sorting students into classes (*needs intervention*, *doesn't need intervention*) rather than trying to predict a quantitative value.

# 2 Exploring the Data

| Statistic | Value |
|---|---|
| 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% |

## 2.1 Passed

The target variable is ‘passed’.



As noted above, 67% of the students passed.

# 3 Preparing the Data

In this section, we will prepare the data for modeling, training and testing.

## 3.1 Identify feature and target columns

The target (as noted previously) is the ‘passed’ column. Here I’ll list the feature columns to get an idea of what’s there.

| Variable | Description | Data Values |
|---|---|---|
| Dalc | workday alcohol consumption | 1, 2, 3, 4, 5 |
| Fedu | father's education | 0, 1, 2, 3, 4 |
| Fjob | father's job | at_home, health, other, services, teacher |
| Medu | mother's education | 0, 1, 2, 3, 4 |
| Mjob | mother's job | at_home, health, other, services, teacher |
| Pstatus | parent's cohabitation status | A, T |
| Walc | weekend alcohol consumption | 1, 2, 3, 4, 5 |
| absences | number of school absences | 0 ... 75 |
| activities | extra-curricular activities | no, yes |
| address | student's home address type | R, U |
| age | student's age | 15, 16, 17, 18, 19, 20, 21, 22 |
| failures | number of past class failures | 0, 1, 2, 3 |
| famrel | quality of family relationships | 1, 2, 3, 4, 5 |
| famsize | family size | GT3, LE3 |
| famsup | family educational support | no, yes |
| freetime | free time after school | 1, 2, 3, 4, 5 |
| goout | going out with friends | 1, 2, 3, 4, 5 |
| guardian | student's guardian | father, mother, other |
| health | current health status | 1, 2, 3, 4, 5 |
| higher | wants to take higher education | no, yes |
| internet | Internet access at home | no, yes |
| nursery | attended nursery school | no, yes |
| paid | extra paid classes within the course subject (Math or Portuguese) | no, yes |
| reason | reason to choose this school | course, home, other, reputation |
| romantic | within a romantic relationship | no, yes |
| school | student's school | GP, MS |
| schoolsup | extra educational support | no, yes |
| sex | student's sex | F, M |
| studytime | weekly study time | 1, 2, 3, 4 |
| traveltime | home to school travel time | 1, 2, 3, 4 |

### 3.2 Preprocess feature columns

Some Machine Learning algorithms (e.g. Logistic Regression) require numeric data so the columns with string-data need to be transformed. The columns in this data-set that had 'yes' or 'no' values had the values converted to 1 and 0 respectively. Those columns that had other kinds of categorical data were transformed into dummy-variable columns.

In addition, the target data was also changed so that instead of 'yes' and 'no' values it contained only '1' and '0' values.

- Original Feature Columns: 30
- With Dummies: 48

With dummy variables there are now 18 more columns in the feature data.

### 3.3 Split data into training and test sets

Next the data was shuffled and then split into training and testing sets.

| Set | Count |
|---|---|
| Training Instances | 300 |
| Test Instances | 95 |

# 4 Training and Evaluating Models

## 4.1 Logistic Regression

The first model I chose was Logistic Regression. Logistic regression is a linear classification model that is useful when the target variable is categorical and the feature variables are numeric (Peng C., et al, 2002). If the features are categorical then they have to be converted to discrete numeric values prior to use. Logistic Regression has the advantage of being computationally cheap, reasonable to implement, and is interpretable but has the disadvantage that it is prone to underfitting (Harrington, 2012).

I chose this model for three primary reasons:

- Its coefficients are interpretable so besides predicting whether a student is at risk, factors that are most influential in determining students who are at risk can be identified
- It suports ridge regression, including lasso regression which might help reduce the number of variables in the data set, both to aid interpretation and improve performance
- It is well understood/well studied

Since it is a linear model it performs best when the data is linearly separable, which makes it a complement to Random Forests, the next model I chose.

## 4.2 Random Forests

Random Forests are ensemble learners that combine predictions from multiple decision trees.

Decision Trees have several advantages including:

- they are interpretable
- they can sometimes fit complex data more easily than linear models
- they don't require dummy variables.

They are, however, generally not as accurate (James G. et al., 2013). Random Forests overcome the failings of the individual Decision Trees using two methods:

- training each tree on a separate data set that is created by re-sampling with replacement from the original training set ( *bagging*) which improves the accuracy of the forest over that of an individual tree.
- each tree in the forest uses a random sub-set of the features when deciding on a split so that there is sufficient diversity in the forest to improve reliability

The predicted output for the features is created by taking an average of the predictions given by the trees.

Random forests, like decision trees, can be used for either classification or regression, but the trade-off for their improved accuracy is that the ensemble is not as directly interpretable as the individual decision trees are. Decision Trees, and thus Random Forests, don't assume linear separability and can perform better than linear models when the data isn't linearly separable, but may not do as well in the cases where the linear model is appropriate (James G. et al., 2013), thus, given the number of variables I thought that it might be more effective on this data set in the event that Logistic Regression cannot model the data well.

## 4.3 K-Nearest Neighbors

My final predictor used K-Nearest Neighbors (KNN) classification. It is a fairly straight-forward method that doesn't build a model in the sense that the other two methods do. Instead KNN stores the training data and when asked to make a prediction finds the *k* number of points in the training data that are 'closest' to the input and calculates then probability of a classification (say *y=1*) as the fraction of the k-neighbors that are of that class. Thus if k=4 and three of the chosen neighbors are classified as *1* then the predicted class will be *1*, because the majority of the neighbors were 1.

Because *KNN* doesn't build a model the way the other two classifiers do, it has the quickest training time but trades this for the longest prediction times.

Unlike Logistic Regression, KNN doesn't require linear separability and unlike some other methods also makes no assumption about the distribution of the data (it is *non-parametric*). This makes KNN more flexible, but how accurate it is depends on the choice of *k*. If *k* is too small it will tend to overfit the training data and if *k* is too large, it will become too rigid. Besides the difficulty in choosing *k*, because it is non-parametric it's not possible to inspect the model to decide which features are important and it needs more data to be accurate (James G., et al., 2013).

I thought that KNN might be a good non-parametric alternative to Logistic Regression since the data comes from students attending two schools in Portugal which might make the instances more similar than dissimilar, the 'nearest neighbor' method was conceptually appropriate and it is different enough in approach that it might improve on the other two methods should the separation of the classes be unusually difficult.

## 4.4 Performance Comparisons

To compare the models each was fit using their default parameters on the same sub-sets of the data. The sub-sets were made of the first 100, 200, and 300 observations of the data. Times are in seconds and the 'best' scores are based on their F1 scores, the weighted average of their prediction and recall scores.

Since the running times are based on my machine's performance as much as that of the classifiers' all times are the minimum value from 100 repetitions (following the advice in the python timeit documentation). The f1-scores are the median values for the 100 repetitions.

### Logistic Regression

| Size | Time (train) | Time (predict) | Train F1 | Test F1 |
|---|---|---|---|---|
| 100 | 0.0011 | 0.0001 | 0.8702 | 0.6720 |
| 200 | 0.0025 | 0.0001 | 0.8333 | 0.7910 |
| 300 | 0.0029 | 0.0001 | 0.8075 | 0.8120 |

**Best score and size of data-set that gave the best test score:**

- best score: 0.81
- best size: 300

### Random Forest Classifier

| Size | Time (train) | Time (predict) | Train F1 | Test F1 |
|---|---|---|---|---|
| 100 | 0.0203 | 0.0011 | 0.9921 | 0.7049 |
| 200 | 0.0217 | 0.0012 | 0.9923 | 0.7302 |
| 300 | 0.0224 | 0.0012 | 0.9949 | 0.7937 |

**Best score and size of data-set that gave the best test score:**

- best score: 0.79
- best size: 300

### K-Nearest Neighbors

| Size | Time (train) | Time (predict) | Train F1 | Test F1 |
|---|---|---|---|---|
| 100 | 0.0004 | 0.0013 | 0.8244 | 0.7519 |
| 200 | 0.0005 | 0.0019 | 0.8099 | 0.7536 |
| 300 | 0.0007 | 0.0027 | 0.8491 | 0.7945 |

**Best score and size of data-set that gave the best test score:**

- best score: 0.79
- best size: 300

### Summation

| Classifier | Score (f1 test) | Training-Size | Training-Time | Prediction-Time |
|---|---|---|---|---|
| LogisticRegression | 0.81 | 300 | 0.0029 | 0.0001 |
| KNeighborsClassifier | 0.79 | 300 | 0.0007 | 0.0027 |
| RandomForestClassifier | 0.79 | 300 | 0.0224 | 0.0012 |

It looks like all three did about equally well on the test-sets.

As expected, KNN had the shortest training time and the longest prediction time. Since the values are so small, I'll look at the ratios of the times next instead of the absolute times.

#### Training Times

| Classifiers | Ratio |
|---|---|
| LogisticRegression/KNeighborsClassifier | 4.44 |
| RandomForestClassifier/KNeighborsClassifier | 34.35 |
| RandomForestClassifier/LogisticRegression | 7.74 |

The Random Forest classifier was 5-10 times slower than the Logistic Regression classifier, which was itself about 5 times slower than the KNN classifier when training the models.

#### Prediction Times

| Classifiers | Ratio |
|---|---|
| KNeighborsClassifier/LogisticRegression | 19.74 |
| KNeighborsClassifier/RandomForestClassifier | 2.21 |
| RandomForestClassifier/LogisticRegression | 8.92 |

It looks like the Logistic Regression classifier was significantly faster than either the Random Forest classifier or the K-Nearest Neighbors classifier - about 20 times faster than KNN and 5-10 times faster than the Random Forest classifier.

### F1 Prediction Scores (Test Set)

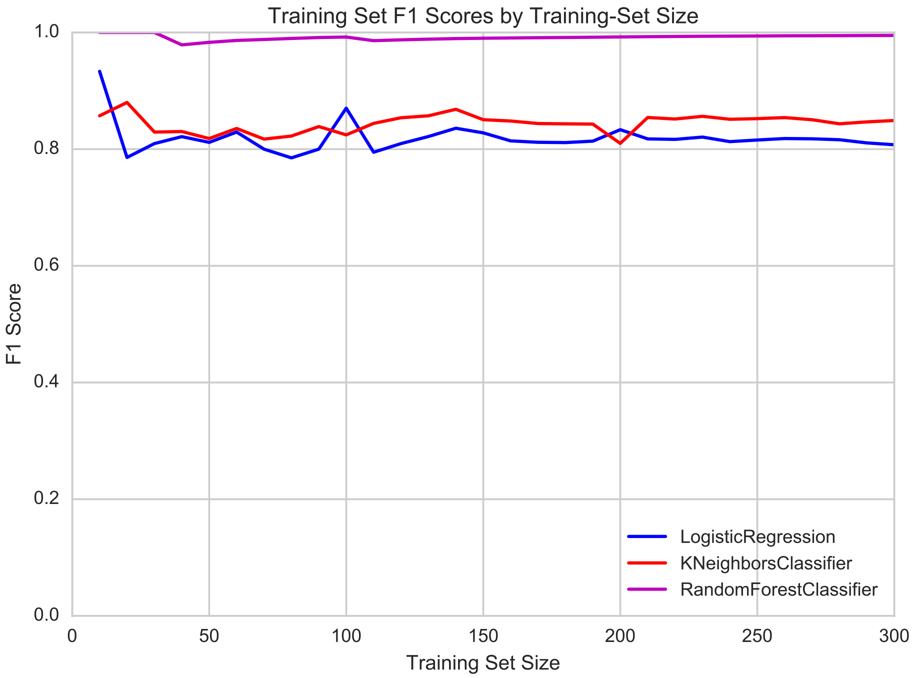
| Classifiers | Ratio |
|---|---|
| LogisticRegression/KNeighborsClassifier | 1.02 |
| KNeighborsClassifier/RandomForestClassifier | 1 |
| LogisticRegression/RandomForestClassifier | 1.02 |

The three models seem to have been comparable once the training data reached 300 instances.

## 4.5 F1 Scores

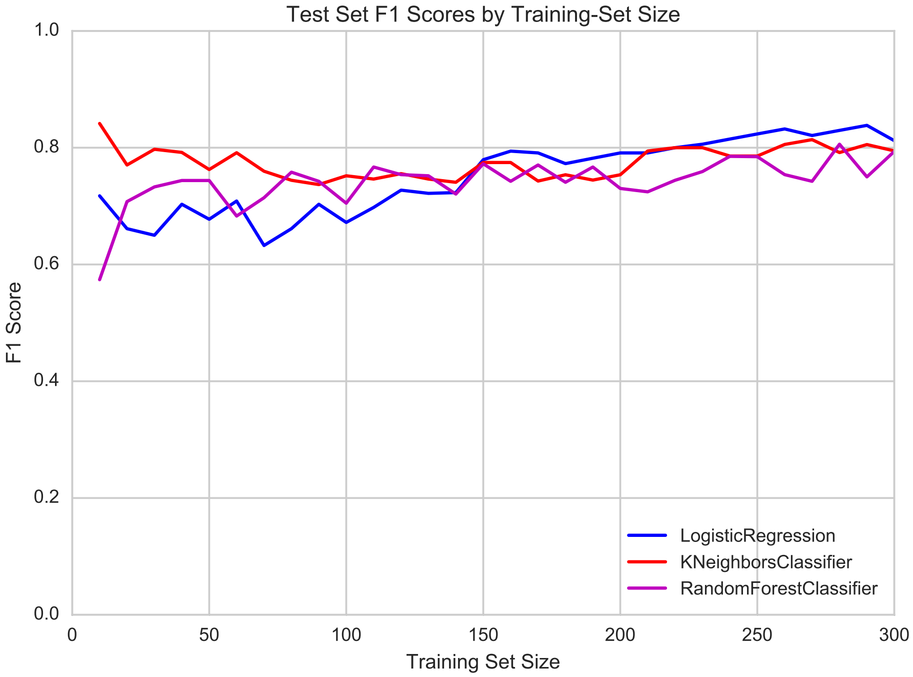
Although I printed the tables for the F1 scores I will plot them here to take a closer look at them. The training-set sizes for the plots ranged from 10 to 300, increasing in steps of 10.

### Training Set



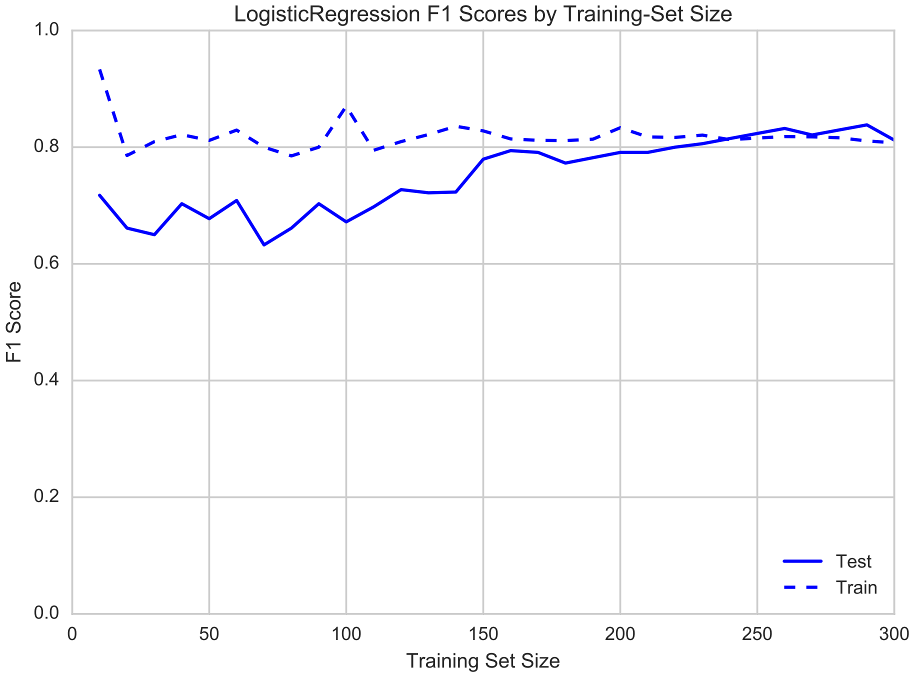
The Random Forest did well on the training set from the start, while the K-nearest Neighbor classifier and the Logistic Regression classifier were erratic until just over 100 training instances. Neither K-Nearest Neighbors nor Logistic Regression was able to do as well on the training set as Random Forests did, suggesting that they are underfitting the data. Interestingly, with this particular split, training set sizes of 100 and 200 (which were used in the tables) appear to have been anomalous with the KNN and Logistic Regression classifiers flipping their relative f1 scores.

### Test Set

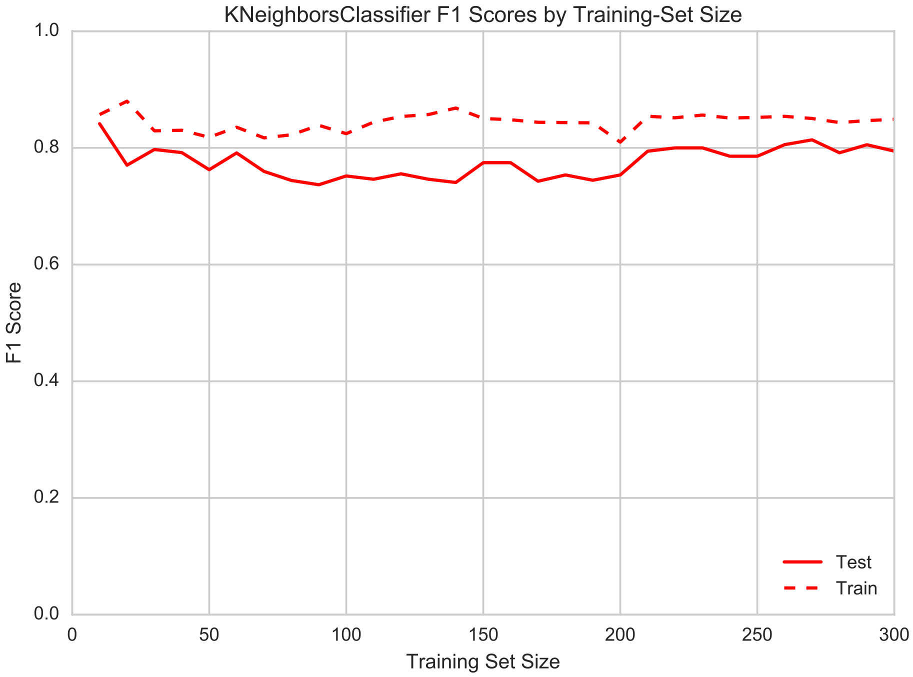


All three classifiers did comparably once the training set was increased to 300 instances, but the Random Forest Classifier shows larger fluctuations in the F1 score than the other classifiers, while the Logistic Regression classifier seemed to be the most stable, and performed the best for most of the instance-counts.

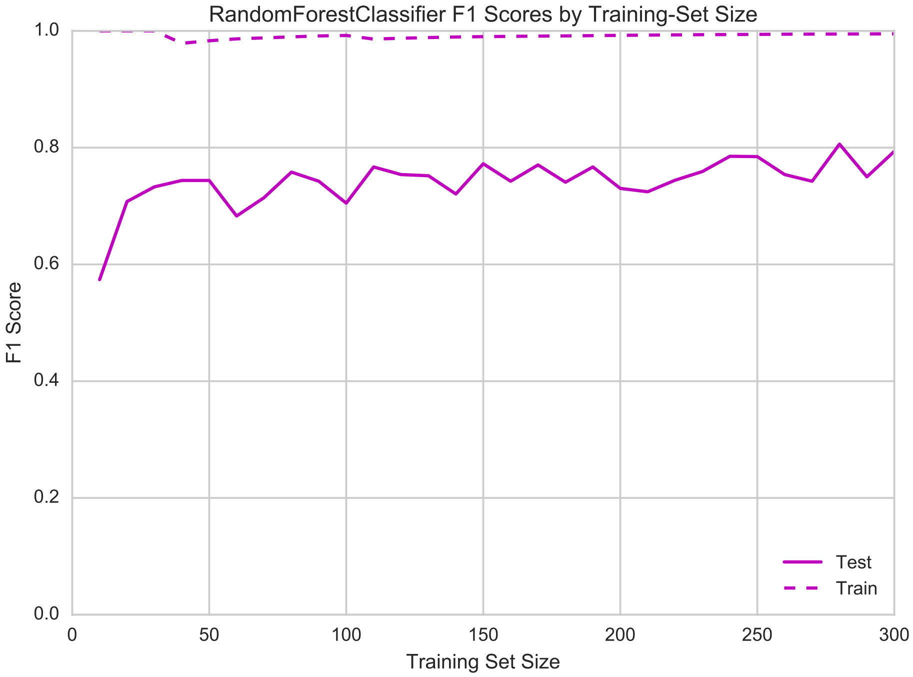
## 4.6 Comparing Test vs Train Scores by Model



The training and testing sets for the Logistic Regression seem to be converging around 0.8, suggesting the model is underfitting and may not be complex enough for the data. Oddly, the test score is better than the training score after about 250 training instances. Looking at the table above, the differences are fractional and might just be that the larger training set has proportionally more difficult instances than the test-set.



The K-Nearest Neighbors classifier seems to perform comparably to the Logistic Regression classifier, although the two curves haven't converged yet, suggesting that it might be improved with more data, although it will still underfit the data.



The Random Forest classifier doesn't do better with the test data than the other two classifiers but is much better with the training data, suggesting that it is currently overfitting, and might be improved with more data.

## 5 Choosing the Best Model

Based on the previous experiments I chose *Logistic Regression* as the classifier to use. Given the data available, all three models have comparable F1 scores (on the test data) but the Logistic Regression classifier is the fastest for both training and prediction when compared to *K-Nearest Neighbor* and *Random Forests*. In addition, the Logistic Regression classifier offers readily interpretable coefficients and L1 regression to sparsify the data, allowing us to see the most important of the variables when deciding who will pass their final exam.

Logistic Regression works by using optimization methods to find the best parameters for a function that calculates the probability that a student's attributes (e.g. *age*, *school*, etc.) predicts that he or she will pass the final exam. When the Logistic Regression model is created, the parameters are adjusted to maximize the *f1 score*, a weighted average of the model's *precision* and *recall*.

Once the model is created, it is used to predict whether a student will pass the exam by multiplying the student's attributes by the model's parameters and these products are summed and added to the intercept. The resulting sum is then converted to a probability - if the probability is greater than $\frac{1}{2}$ the prediction will be that the student passed (*passed='yes'*) otherwise it will be that the student did not pass (*passed='no'*).

### 5.1 Fitting The Model

The Logistic Regression model was fit using sklearn's *GridSearchCV* with 10 folds for the cross-validation. The parameters tested were the penalty-type (*l1* or *l2*), *C* - the inverse of the regularization strength (the smaller the number the larger the penalty), and weights associated with each class.

The *penalties* refer to regularization penalties that shrink or eliminate variable-coefficients in the model. The *l1* penalty refers to *lasso regularization* which causes some of the feature-variable coefficients to go to `0` if they don't contribute as much to the model as the other, more significant variables. The *l2* penalty refers to *ridge regression* which shrinks the coefficients but never pushes them all the way to `0`. Since *lasso regularization* simplifies the model it might seem that it would be preferable, but it will only outperform *ridge regression* if there are differences in how much the variables contribute. If all the variables contribute equally to the model, then *ridge regression* will outperform the *lasso* (James G. et al., 2013).

*C* is the inverse of the regularization strength. The stronger the regularization (and thus the smaller *C* is), the smaller the coefficients in the model will be and for *lasso* the fewer the coefficients it will have. Shrinking the coefficients too much or eliminating too many variables can weaken the model so how much regularization is needed is determined here using cross-validation.

The weights used for the classes were either 1, 'balanced' $\left(\frac{\textit{number of samples}}{\textit{number of classes}\times<\textit{number of 0's in y, number of 1's in y}>}\right)$, or 0.67 for passed and 0.33 for didn't pass.

The following is the outcome of fitting the Logistic Regression classifier using Grid Search with the parameters given.

### Parameters

| Parameter | Value |
|---|---|
| penalty | l1 |
| C | 0.09 |
| class_weight | |

These are the parameters for the model that had the highest F1 score. The best scorer used *l1* (lasso regularization) suggesting that the features contribute unequally to the outcome. The amount of regularization used was fairly high (*C* is the inverse of the amount of regularization, the smaller it is the more regularization there is) so the model should be fairly sparse compared relative to the number of variables in the data set. Although it looks like there was no `class_weight` set, this is because `None` was chosen, so the class weights were assumed to be 1.

### Coefficients

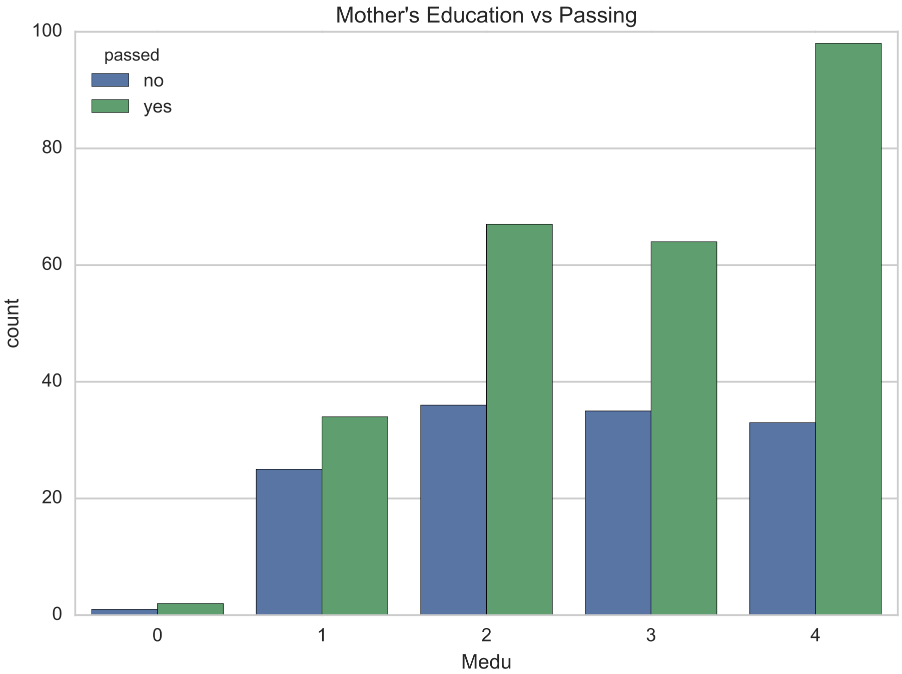
| Variable | Description | Coefficient | Odds |
|---|---|---|---|
| Medu | mother's education | 0.14 | 1.16 |
| age | student's age | 0.03 | 1.03 |
| famrel | quality of family relationships | 0.03 | 1.03 |
| Fedu | father's education | 0.01 | 1.01 |
| absences | number of school absences | -0.02 | 0.98 |
| goout | going out with friends | -0.07 | 0.93 |
| failures | number of past class failures | -0.46 | 0.63 |

These are the variables that remained in the best model after the regularization was applied, sorted by their coefficient-values. The coefficients are log-odds so calculating $e^{coefficient}$ gives you the increase in odds that the student will graduate (Peng C. et al., 2002). If the odds are greater than one, then increasing the attribute associated with the coefficient will increase the probability that the student passed and if the odds are less than one, then increasing the attribute associated with the coefficient will reduce the probability that the student passed.

### Positive Contributions

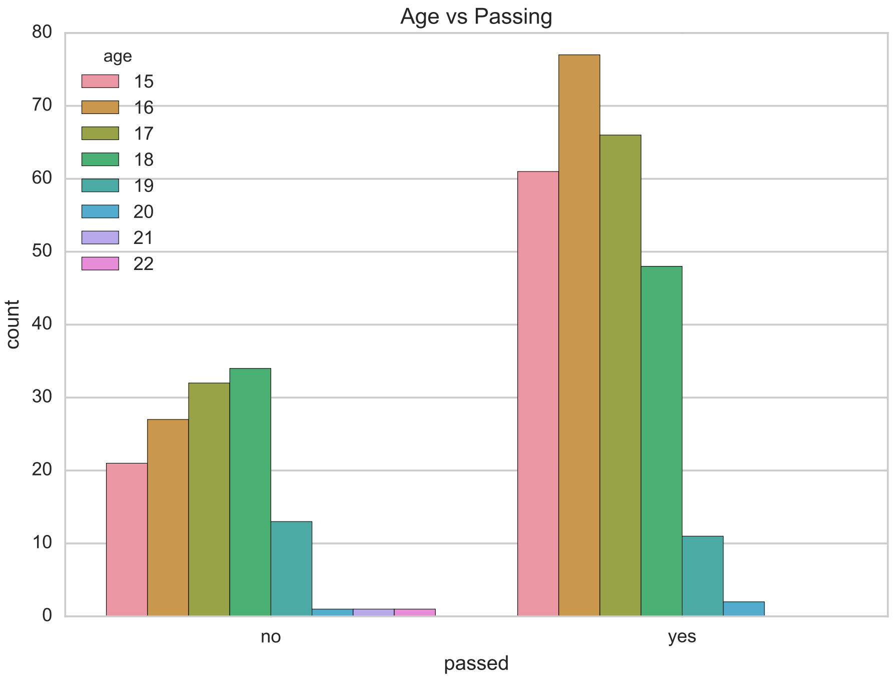
**Medu** The predictor with the greatest positive effect in the model was the amount of education the student's mother received. According to the read-me file there are 5 levels.

| Level | Meaning |
|---|---|
| 0 | None |
| 1 | 4th Grade |
| 2 | 5th to 9th Grade |
| 3 | Secondary Education |
| 4 | Higher Education |

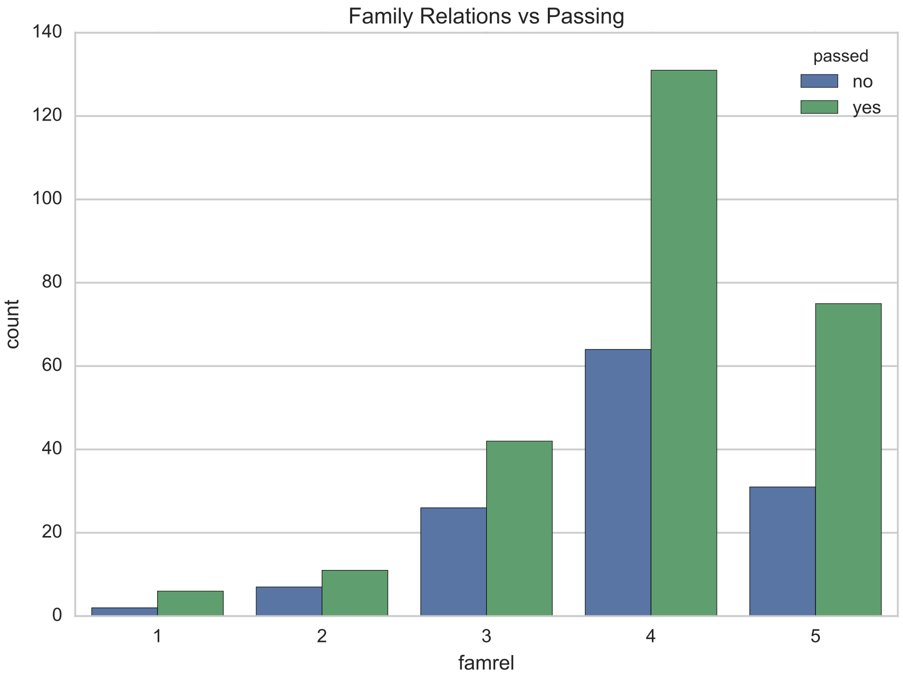


Looking at a plot of all the data, as the mother's education-level goes up the proportion of those that pass goes up relative to those that don't pass, with a large jump from Kindergarten - 4th grade to 5th - 9th grade levels .

**age** This is the age of the student. It wasn't immediately obvious why this would be a factor, assuming that the students are all in the same grade, but a plot of the data showed that the ages range from 15 to 22, with the oldest students not passing the final exam, and the 15-17 year olds being more proportionally represented among those who passed than those who didin't, possibly because the older students were held back and thus were lower-performers to begin with.

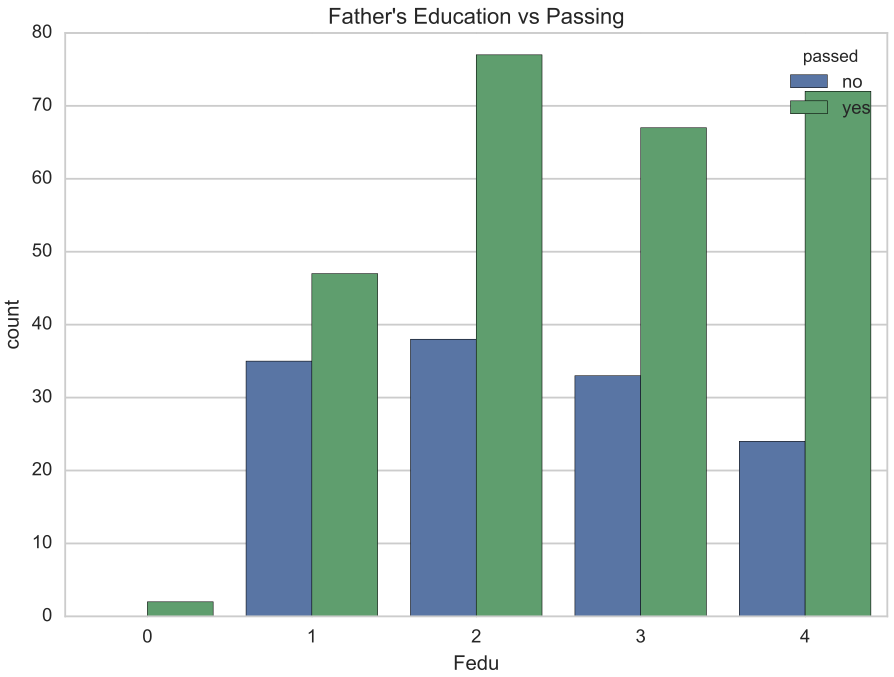


**famrel** According to the readme file `famrel` is a rating of the quality of family relations ranging form 1 (very bad) to 5 (excellent).



The plot seems to show that relations of above average (4 and 5) family-relations improved the likelihood of passing.

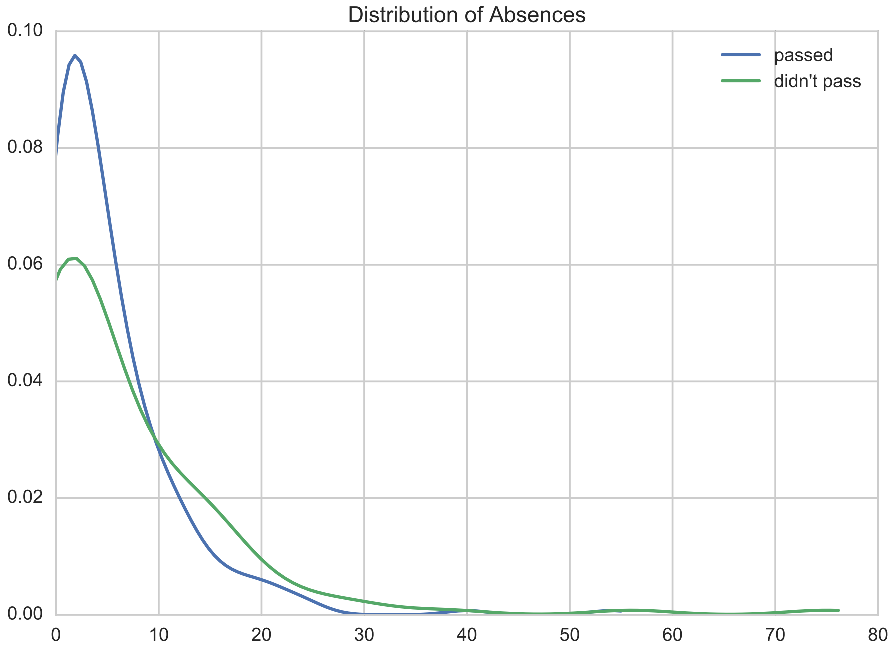
**Fedu** This is the student's father's education level. The levels are the same as *Medu*.



### Negative Contributions

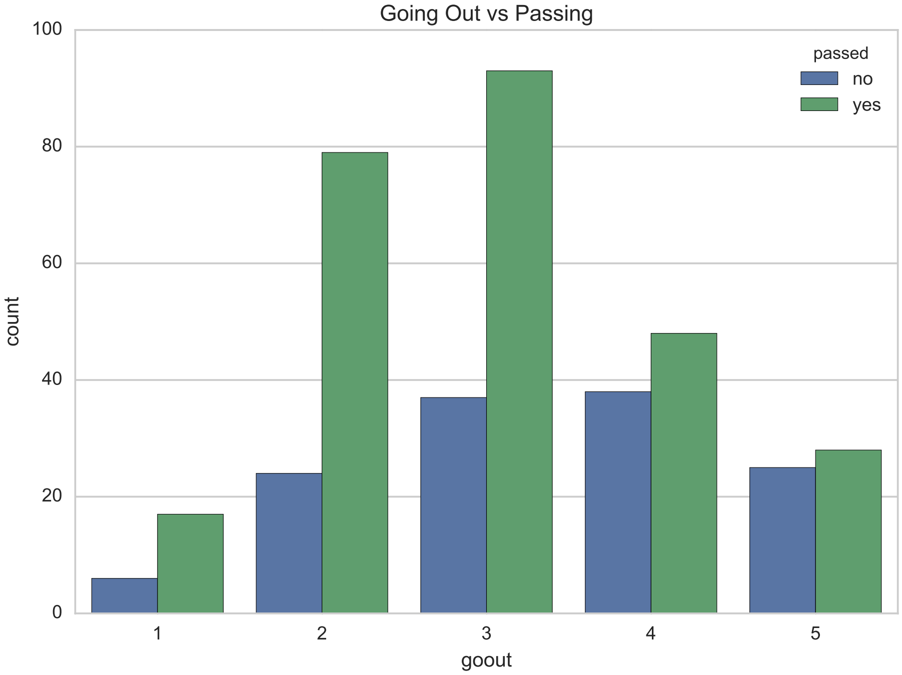
These are variables that decrease the odds of a student passing as their values increase.

**absences** This is a straightforward count of the number of absences there were.



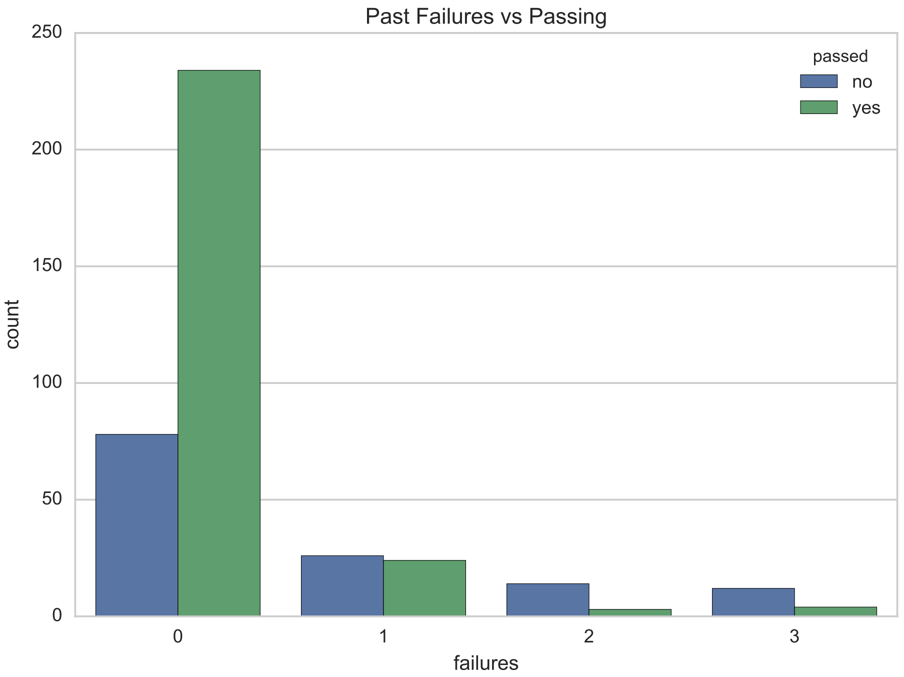
The differences are not large, but the distribution for those that didin't pass has a greater spread and at around 10 absences the distributions seem to cross over with the non-passing line raised above the passing line (indicating a greater proportion of non-passing students).

**goout** This is how often the student goes out with friends. There are 5 levels from 1 (very low) to 5 (very high).



It looks like going out more than the average has a negative effect on passing.

**failures** This is the most negative variable and represents the number of past class failures for a student. It ranges from 0 to 4 with 4 meaning 4 or more failures.



It looks like no student failed 4 or more classes before taking the final exam and there were more failing students than passing students once there were any failures.

### 5.2 F1 score (Test Set)

The best F1 score for the Logistic Regression classifier was 0.86, which is a slight improvement over the default Logistic Regression classifier used earlier which had an f1 of approximately 0.81 for the test set when trained with 300 training instances.

## 6 References

Alpaydin E. 2010. Introduction to Machine Learning. Second Edition. Cambridge, MA: The MIT Press.

Harrington P. 2012. Machine Learning in Action. Shelter Island, NY: Manning Publication Co.

James G., et al. An Introduction to Statistical Learning: with applications in R. New York: Springer.

Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. 2002. An introduction to logistic regression analysis and reporting. The Journal of Educational Research, 96(1), 3-14.

Witten I., Eibe F. 2005. Data Mining: practical machine learning tools and techniques. Second Edition. San Francisco, CA: Morgan Kaufman Publishers.