# Building a Student Intervention System: An Udacity Nanodegree ML Project

Omoju Miller

May 16, 2016

## Introduction

Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?

- This task sounds like a problem that would be best suited for a classification algorithm. The inherent task is to develop learners that can ”predicting a category.” If we look at the problem from another perspective, we can consider the student data available as a “labeled” dataset. We have features that we can use to determine who has succeeded in the class versus who has not. For that insight, we could use ‘`passed`’ column as our class label. Therefore, this is a binary classification problem for predicting discrete labels that a student might belong to.

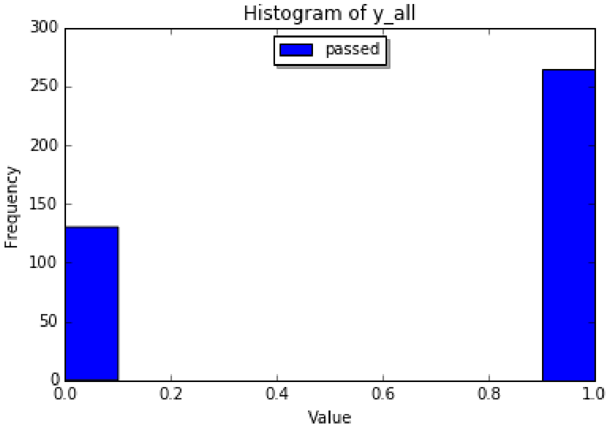


Figure 1: **Values of ‘passed’ column** *The histograms shows an unbalanced target dataset with approximately 125 values of* {*0 - did not pass*} *and over 250 values of* {*1 - passed*}.

# Training and Evaluating Models

For the student intervention challenge, three supervised learning algorithms have been selected as appropriate learners for the task. The algorithms are as follows:

1. Decision Tree Classifier
2. Random Forest Classifier
3. Support Vector Machines

## Decision Tree Classifier

- What is the theoretical $O(n)$ time & space complexity in terms of input size?
  The theoretical time & space complexity of decision trees classifiers as implemented in sci-kit learn package is:

  - Best Case: $\Theta(pN \log^2 N)$
  - Worst Case: $O(pN^2 \log N)$
  - Average Case: $\Theta(pN \log^2 N)$

  Where $N$ denotes the number of samples, and $p$ the number of input variables. [1]

- What are the general applications of this model?
  The decision tree algorithm is usually applied to classification and regression problems. They are very popular in operations research, especially for building decision support systems.
  What are its strengths and weaknesses?

  - Strengths of decision trees:

    - Very intuitive. You can look at the results and understand it.
    - Requires little data preparation.
    - The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree. In average case the cost of training $O(pN \log^2 N)$
    - Able to handle both numerical and categorical data.
    - Very robust. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

  - Weaknesses of decision trees:

    - Decision-tree learners can create over-complex trees that do not generalize the data well. They are prone to over-fitting especially in the case of data with lots of features.
    - Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

[1] Complexity analysis gotten from Louppe, Gilles PhD dissertation *Understanding Random Forests: From Theory to Practice*, 2014.

    - ∗ The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.
    - ∗ There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
    - ∗ Decision tree learners create biased trees if some classes dominate.

- Given what you know about the data so far, why did you choose this model to apply?
  The *major* reason why the decision tree classifier was selected was its interpretability. For this problem domain, it isn't just satisfactory to identify students that need intervention, what learning researchers ultimately want is to gain *insights* into the nature of learning, and the social factors that lead to certain outcomes for at-risk students. A decision tree learner, with its ability to graphically plot out the tree becomes a research tool in the hands of learning scientists. Consequently, this can help the school board of supervisors build better solutions for those students which well executed could potentially reduce the costs associated with remediating failed students.

## Random Forest Classifier

- What is the theoretical $O(n)$ time & space complexity in terms of input size?
  The theoretical time & space complexity for building a complete unpruned decision tree is:
  - Best Case: $\Theta(MK\widetilde{N}\log^2\widetilde{N})$
  - Worst Case: $O(MK\widetilde{N}^2\log\widetilde{N})$
  - Average Case: $\Theta(MK\widetilde{N}\log^2\widetilde{N})$

  Where $M$ denotes number of randomized trees, $N$ the number of samples, and $K$ the number of variables randomly drawn at each node. $\widetilde{N} = 0.632N$, due to the fact that bootstrap samples draw, on average, $63.2\%$ of unique samples. [2]

- What are the general applications of this model?
  Ensemble learners are used in supervised learning. They have multiple applications. They have been used in bioinformatics for example to classify micro-array data, they have been used in engineering to solve aircraft engine fault diagnosis, they are used in education data mining to build student intervention systems and so forth.

- What are its strengths and weaknesses?
  - Strenghts of SVMs:
    - ∗ Considered one of the best off-the-shelf learning algorithm, requires almost no tuning.
    - ∗ Fast to train because algorithm lends itself well to parallelization
    - ∗ Flexible, can be used with large number of attributes, small or large datasets
    - ∗ Good control of bias and variance because of the averaging and randomization which leads to better performance.

[2] Complexity analysis gotten from Louppe, Gilles PhD dissertation *Understanding Random Forests: From Theory to Practice*, 2014.

  - Weaknesses of decision trees:
    * Loss of interpretability as compared to decision trees give.

- Given what you know about the data so far, why did you choose this model to apply?
  From a cursory

  This learners was chosen as a means of reducing the effects of overfitting of decision trees.

## Support Vector Machine (SVMs)

- What is the theoretical $O(n)$ time & space complexity in terms of input size?
  The theoretical time & space complexity of SVMs is:

  - Best Case: $\Theta(pN^2)$
  - Worst Case: $O(pN^3)$
  - Average Case: $\Theta(pN^2)$

  Where $N$ denotes the number of samples, and $p$ the number of input variables. It can be a costly algorithm since the compute and storage requirements increase rapidly with the number of training vectors. Even though the algorithm spends more time in the training, however, it achieves better $F_1$ score for prediction – thus it is more accurate.

- What are the general applications of this model?
  SVMs are an algorithm used for classification, regression and outlier detection. They are popular in bioinformatics for protein classification. They are also frequently used in text and hypertext classification.
  What are its strengths and weaknesses?

  - Strenghts of SVMs:

    * Versatile: utilizes **kernel** transformation and several functional forms so that what was once non-linearly separable now becomes linearly separable.
    * Effective in high dimensional spaces.
    * Still effective in cases where number of dimensions is greater than the number of samples.
    * Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

  - Weaknesses of decision trees:

    * If the number of features is much greater than the number of samples, the method is likely to give poor performances.
    * SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
    * SVM is a binary classifier. To do a multi-class classification, only pair-wise classifications can be used (one class against all others, for all classes).

- Given what you know about the data so far, why did you choose this model to apply?

One of the more interesting aspect of this dataset is that it is unbalanced. As previously stated, and can be seen from figure 1 there are more examples of student success than failure. As such, the optimal learner for this problem is one that can still generate reasonable classification given the unbalanced dataset. SVMs are very very robust classifiers and *more importantly*, they have a method of *biasing* the soft-margin constant, $C$, to correct for class imbalances. The solution is to assign a different soft-margin constant to each class.

# Choosing the Best Model

## Required:

- Please thoroughly compare the F1 scores and computation costs of each model with each other model. Well done using accuracy to choose SVMs, however. Please make your comparison more thorough, and include Decision Trees in your comparison.

**Question** Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?

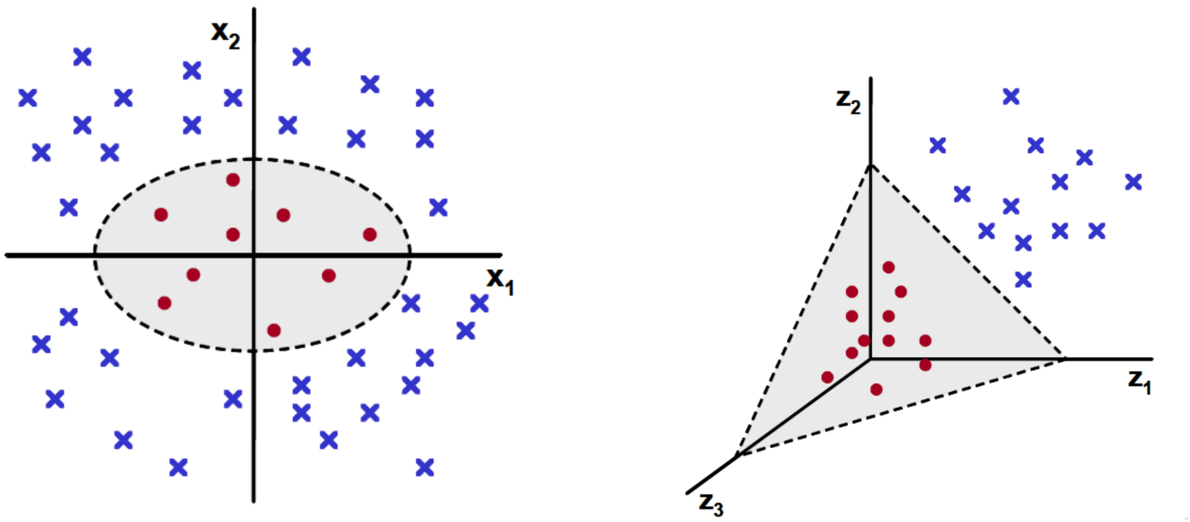
**Answer** For the problem of identifying students that need intervention, I would advice the board of supervisors to go with a **Support Vector Machine**. First, it is a relatively fast algorithm to train and predict. Second, as we can see from figure 1, the dataset is quite unbalanced (number of passed students $>>$ number of failed students) a relatively small. The SVM algorithm is able to handle this better than the other two models.

While its marginally slower in prediction than Random Forest classifier with a prediction time of 0.001 versus 0.006 as can be seen from tables 3 and 4. The difference of negligible and SVM makes up for what it loses in speed with improvement in accuracy.

**Question** In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).

**Answer** A Support Vector Machine is a class of discriminatory algorithms. The goal is to try and correctly classify a dataset into separate classes. Subject to that constraint, an SVM picks a decision boundary that separates the classes by maximizing the distance to the nearest points in either classes. These points are a subset of the dataset and are called support vectors.

The most interesting thing about SVMs is what is know as the *kernel* trick. This procedure make linear models work in nonlinear settings by mapping the data into higher dimensions where we



(a) Features in two dimensional space.

(b) Features projected in three dimensional space.

Figure 2: **Kernel trick.** *Figure (a)* $\mathbf{x} = \{x_1, x_2\}$. *Figure (b)* $\mathbf{z}$ *by* $\mathbf{x} = \{x_1, x_2\} \longrightarrow \mathbf{z} = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$. *By mapping the data from two dimensions to three dimension, the data now becomes linearly separable in the new representation.*

can see the linear behavior. For example, figure 2a shows a data in two-dimensional space where it is evident we will not be able to find a decision boundary that separates the data into two classes. Using a kernel, the features are projected into a three dimension space as is shown in figure 2b. In this space, we can find a hyperplane that will linearly separate the data into two classes. This aspect of SVMs is what makes them really robust.

An SVM does prediction on the data by classifying the test dataset based on the decision boundary it "fitted" during its training phase. There are several evaluation metrics that let you determine the accuracy of the algorithm.

**Question** Fine-tune the model. What is the model's final $F_1$ score?

**Answer** As it is often done in practical machine learning, we try several kernels first, often the linear kernel, then the RBF kernel. Using the linear kernel, the $C$ hyper-parameter was tuned with the following values: $\{0.03125, 0.0625, 0.125, 0.25, 0.5, 1.0, 2.0\}$. Figure 3 shows the results of trying various $C$ with gridsearch.

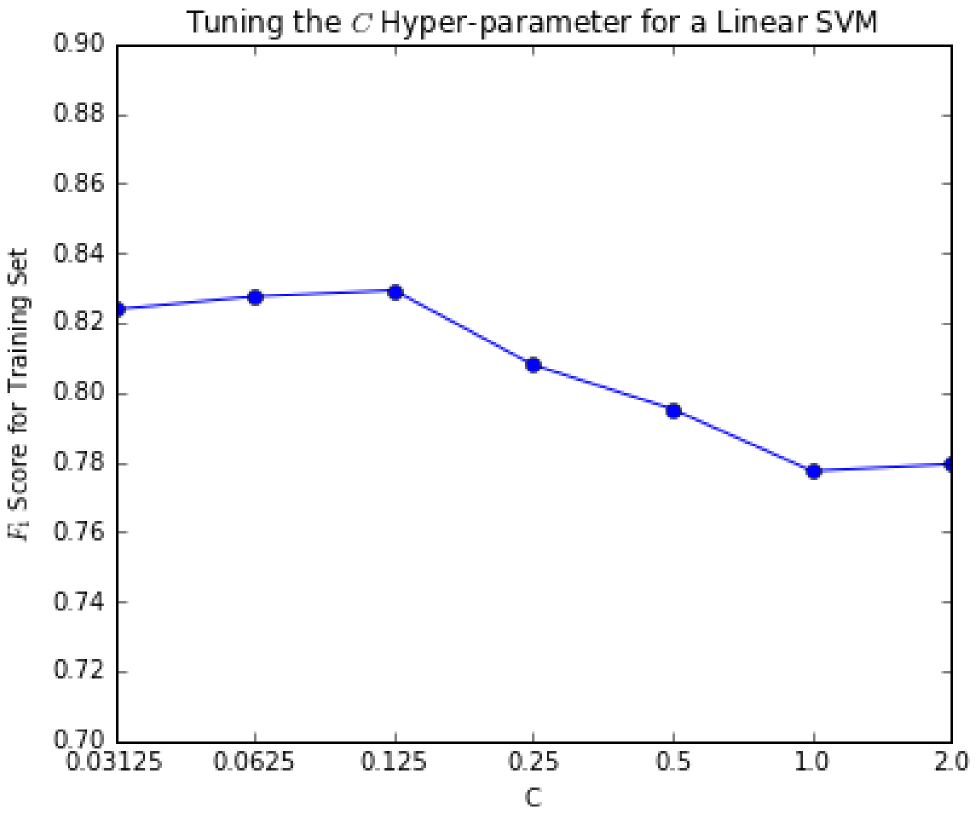


Figure 3: **Values of $F_1$ scores for different values of $C$.** *From this figure, we can see that we achieve the highest value of $F_1$ where $C$ = 0.125*

Table 1: Result of tuning with **GridSearchCV**

| | GridSearchCV |
|---|---|
| F1 score for training set | 0.848 |
| F1 score for test set | 0.800 |
| | Best Parameters |
| Kernel | Linear |
| $C$ | 0.125 |

# Tables

Table 2: Result of training with a **DecisionTreeClassifier**

| | Training set size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.001 | 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 | 0.610 | 0.750 | 0.633 |

Table 3: Result of training with a **Random Forest Classifier**

| | Training set size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.019 | 0.019 | 0.019 |
| Prediction time (secs) | 0.001 | 0.001 | 0.001 |
| F1 score for training set | 1.000 | 0.993 | 0.990 |
| F1 score for test set | 0.716 | 0.726 | 0.746 |

Table 4: Result of training with a **SVMs**

| | Training set size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.003 | 0.006 |
| Prediction time (secs) | 0.001 | 0.001 | 0.002 |
| F1 score for training set | 0.878 | 0.868 | 0.876 |
| F1 score for test set | 0.775 | 0.781 | 0.784 |