Machine Learning Nanodegree Project 2

1. Classification vs Regression

Q: Which type of supervised machine learning problem is this, classification or regression? Why?

The goal of this exercise is to determine if a student will pass or fail. In other words, this problem is binary, we are not "grading" students on a scale, we just want to know if they passed or not, and for that reason this problem is a classification problem.

Alternatively, if the problem were to predict the final grade of a student on a numeric scale, such as 1-100, the problem *could* still be classification, but would most likely be better suited to a regression problem.

A: Classification

2. Exploring the Data

Q: Can you find out the following facts about the dataset?

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

3. Preparing the Data

Q: Execute the following steps to prepare the data for modeling, training and testing:

1. Identify feature and target columns
2. Preprocess feature columns
3. Split data into training and test sets

4. Training and Evaluating Models

Q: Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem. For each model:

1. What are the general applications of this model? What are its strengths and weaknesses?
2. Given what you know about the data so far, why did you choose this model to apply?

3. Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.
4. Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

A:
I have chosen K-Nearest Neighbors, Support Vector Machines, and Naïve Bayes as my models. Each will be discussed in their own sections below.

K-Nearest Neighbors

The K Nearest Neighbor classifier method was chosen as a baseline method based on its simplicity.

Some benefits of the K-nn method for this problem:

- This method has no learning process, therefore computational cost during the learning phase is zero.
- An easy to understand method that uses very simple operations to achieve its approximation.

Some issues with K-nn in this case:

- The selection of distance function is difficult in such a high dimensional space
- K-nn will generally return better results with larger datasets, with this data set being so small, the accuracy of the prediction is questionable

The underlying theory was to test whether some sub group of common features created a cohort more likely to pass than those students who did not belong in that cohort. With that hypothesis in mind, the model was tuned for its initial run to a high number of neighbors (20) and to weight neighbors by distance. This would show whether the theory of having cohorts with common features explained pass rates and would provide a prediction time and score baseline. For example, if having a good attendance record and internet access were strong indicators of a passing score, K-nn would have identified that pattern very well at a fairly low computational cost.

Table 1: Model 1 utilizing K Nearest Neighbors

KNN	**Training Set Size**		
	100	**200**	**300**
Training Time (secs)	0.001	0.001	0.002
Prediction Time (secs)	0.002	0.003	0.004
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.7792	0.8	0.7895

Naïve Bayes

Benefits of Naïve Bayes:

- Naïve Bayes is an extremely fast algorithm with very low computational cost especially in very high feature spaces.

Issues with Naïve Bayes:

- The high dimensionality of this data set could be an issue with Naïve Bayes. If it happens that student success is better indicated by the interrelationships between a high number of features, the independent variable nature of Naïve Bayes will completely ignore those relationships.

Naïve Bayes (NB) was the second model chosen for the exact opposite reason as K-Nearest Neighbors. Naïve Bayes assumes independence between features (thus Naïve). If the feature set starts to present with a lot of either-or decisions, such as a strong attendance record and internet access or highly educated parents, but not all three, Naïve Bayes would perform very well at predicting a pass/fail outcome. Additional benefits of NB are the low computational cost as the calculation is actually fairly elementary and the ability of the algorithm to create a model with very little training data.

Table 2 Model Utilizing Naïve Bayes

Gaussian NB	**Training Set Size**		
	100	**200**	**300**
Training Time (secs)	0.002	0.002	0.002
Prediction Time (secs)	0.001	0.001	0.002
F1 score for training set	0.6729	0.8030	0.7871
F1 score for test set	0.5106	0.736	0.7538

Support Vector Machines

Benefits of SVMs:

- Works very well with smaller data sets, such as the one used in this case.
- Works well in complex domains with a lot of features, such as in this case

Issues with SVMs:

- If the data is noisy or overlaps, the result will not be very accurate.
- When data sets become large, computational cost rises significantly.

The final method chosen, and the one that I initially assumed would perform the best, was Support Vector Machines (SVMs). SVMs are especially well suited for high dimensional data sets and are very "tunable" for performance. The initial test run was operated using the sci-kit learn default settings for SVC, which uses the provided rbf kernel. This method returned results similar to K-nearest neighbors, however, I believe with tuning it could achieve F1 scores significantly higher than the K-nn model with similar prediction times.

Table 3 Model Utilizing Support Vector Machines

SVMs	**Training Set Size**		
	100	**200**	**300**
Training Time (secs)	0.002	0.005	0.008
Prediction Time (secs)	0.002	0.004	0.005
F1 score for training set	0.8947	0.8926	0.8701
F1 score for test set	0.7919	0.8108	0.7947

5. Choosing the Best Model

Q: Based on the experiments you performed earlier, in 2-3 paragraphs explain to the board of supervisors what single model you choose as the best model.

- **Which model has the best test F1 score and time efficiency?**
- **Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?**

Please directly compare and contrast the numerical values recorded to make your case.

Based on the three models chosen, I believe that the Support Vector Machines model is the most appropriate for this use case. Naïve Bayes did not perform well in comparison with the other two models, providing an F1 score 5% below that of the SVM model and 4.5% below the K-nn model. The Naïve Bayes model did, however, perform much more quickly, by a factor of two, than the other two methods on this limited dataset. In my opinion, the sacrifice in performance is acceptable for the improved accuracy in making predictions, and we will therefore rule out the Naïve Bayes method.

Comparing K-nn and SVM models, they ran predictions in very similar times for all different training set sizes. SVM consistently outperformed the K-nn model by at least 0.6% before model tuning and I will therefore choose the more accurate model, Support Vector Machines

Q: In 1-3 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 learn to make a prediction).

Support Vector Machines operate by assigning data from a training set to one of two categories based on features of the data, in this case the assignment would be to either pass or fail. It then creates a boundary between the assigned data points, maximizing the boundary by creating the largest span between the data. The model will then predict new data by assigning it to a region established by the training data.

For an example of SVMs in a low dimensional space, if we assume that data is input into a 2-dimensional space the data could be randomly placed and not grouped in any significant way. This does not lend itself to a simple boundary between classes. However, using a method known as "The Kernel Trick" data can be projected onto a higher dimensional space and separated by a "hyperplane".

Figure 1, from www.eric-kim.net is an excellent graphical example of the Kernel Trick and how SVMs learn by projecting data into higher dimensional spaces and finding decision boundaries with the maximum margin, or distance between points from two classes.

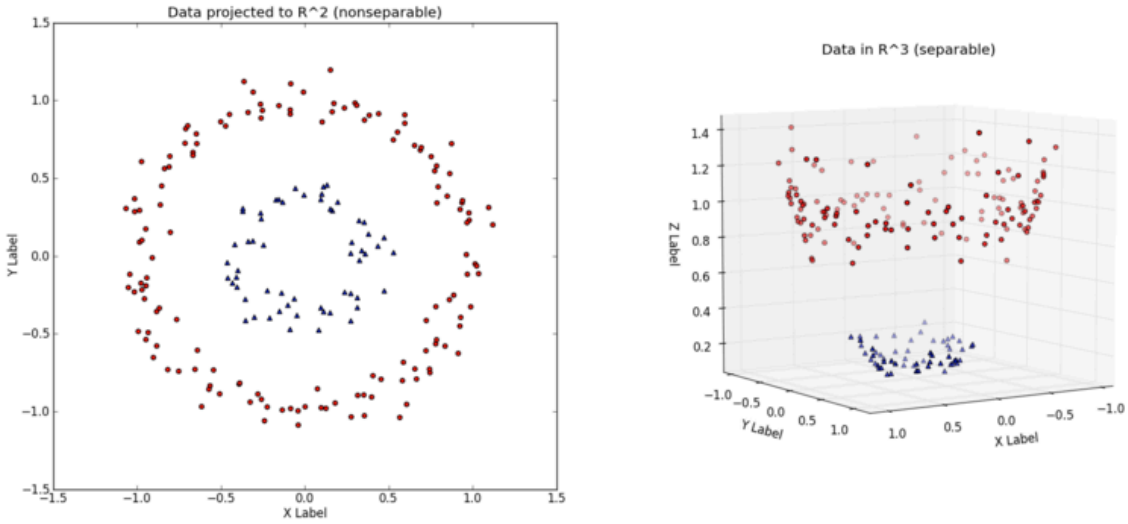

Figure 1 Example of dataset projected from a 2-dimensional to a 3-dimensional space[1]

Fine-tune the model. Use gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.
What is the model's final F1 score?

Using GridSearchCV in sklearn, I had the SVC model run with three different kernel functions and error penalty parameters, rbf, linear, and poly and 1, 10, and 100 respectively. The final results were actually that the initial run was the best, using the SVC defaults using the rbf kernel and an error penalty parameter of 1

Final F1 Score	**0.8134**

[1] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html , Accessed February 1, 2016