Udacity Machine Learning Nanodegree
Project Two: Student Intervention System
Josh Casale
1/31/2015

1) Classification vs Regression

- Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?
    - In this problem, we are trying to differentiate students into a discrete number classes (those who need early intervention and those who do not). Therefore, this is a classification problem.

2) Exploring the Data

- Total number of students: 395
- Number of students who passed: 265
- Number of students who failed: 130
- Graduation rate: 67%
- Number of features (excluding the label/target column): 31

3) Preparing the Data

- Identify feature and target columns: ✓
- Preprocess feature columns: ✓
- Split data into randomly generated 75/25 training/test set: ✓

4) Training and Evaluating Models

MODEL 1: Decision Tree Classifier.

- This model can be applied to both classification and regression problems, but is typically used for classification. In the context of a classification problem, this model infers simple decision rules from a data set to create a decision boundary that separates the data into classes.
- **PROS:**
    - Does not require normalisation of the input data
    - Can handle multi-output problems
    - Cost of making a prediction using the tree is log(n) where n is the number of data points used to train the tree.
- **CONS:**
    - Prone to overfitting
    - The resulting tree is very sensitive to the input data (the output model is not very stable)

    - In practice, we must use heuristics to approximate an optimal tree, rather than generating an explicit solution (which can be an NP-complete problem).
    - Decision trees create piecewise-linear decision boundaries, which are not always optimal for describing a given dataset.
- I chose to apply this model because it is a simple-to-implement, yet effective model for classification, and because we have only 31 features and two classes in our dataset, it is cheap to compute.

	Training set size		
	100	200	300
Training time (secs)	.0011	.0018	.0021
Prediction time (secs)	.001	.001	.002
F1 score for training set	1.0	1.0	1.0
F1 score for test set	.6942	.7167	.6977

MODEL 2: Random Forest Classifier.

- Pros:
    - Logical extension of conceptually simple Decision Tree model
    - Applicable to both regression and classification problems
    - Easily parallelizable for large datasets
    - Gives estimates of weights for input variables
- Cons:
    - Can suffer from overfitting given noisy data. Rectifying this via pruning is a nontrivial problem.
    - High degree of model complexity make model tuning a non-intuitive process
- I chose to apply this model because while it is based on a decision-tree classification model, ensembling techniques make this model more sophisticated, and thus more powerful.

	Training set size		
	100	200	300
Training time (secs)	.022	.023	.026

Prediction time (secs)	.001	.001	.001
F1 score for training set	.984	.990	.996
F1 score for test set	.7407	.7907	.7442

MODEL 3: Logistic Regression Classifier:

- Pros:
  - Very simple model
  - Fast and cheap to compute, especially given high-dimensional datasets
  - Resistant to overfitting
- Cons:
  - Can be unstable when one feature in the dataset is a much stronger predictor than the others.
  - Less powerful under ideal conditions than Decision Trees or Random Forests
  - **REVISION:** Because Logistic regression creates a linear decision boundary to separate the data, it is not well-suited for classifying data that contains underlying nonlinear, or "unstable" patterns. Statistically speaking, an operation (in our case, classification) is "unstable" is a small change in a data-point may result in a large difference in the result of the operation. Simply put, if there exist patterns in the data where a small change changes the classification of the data, logistic regression will not perform well.[1]
- I chose to apply this model due to its simplicity and fundamental difference from the other models that I used, which are both based on decision trees. If this model's performs well on the task, there is no need to introduce the complexity of the other models.

	Training set size		

[1] pp 405-406: Nonlinear Estimation and Classification, edited by David D. Denison, Mark H. Hansen, Christopher C. Holmes, Bani Mallick, Bin Yu

	100	200	300
Training time (secs)	.0012	.0015	.0031
Prediction time (secs)	.026	.026	.026
F1 score for training set	.838	.850	.873
F1 score for test set	.7218	.7612	.7591

5) Choosing the Best Model

When choosing a model, there are a number of factors that should be taken into account. In our use-case, we have a dataset that is very small, but robust in terms of feature count, we have limited computational resources, and because we have a small development team, (just me!) we have limited developer resources as well. While more complex models can be very powerful under ideal circumstances, the amount of model complexity is directly correlated to the size of a training set needed to achieve good performance. Because we only have 300 samples to train on, this make model simplicity important. Complex models also typically require more computational resources to train; this is reflected in the fact that my Random Forest model took approximately 15-20x the training time of the Logistic Regression and Decision tree classifiers.

Of course, model simplicity is worthless if the model doesn't perform well. In my observations of out-of-the-box models, the Decision Tree model achieved a maximum test set F1 score of ~.72 on the and the Logistic Regression model achieved a maximum test set F1 score of ~.76. The more complex Random Forest model did outperform these models with a maximum test set F1 score of .79, but given the order-of-magnitude increase in training time required to do so, I do not believe that the added complexity of the model is cost-effective.

Taking all of the above into account, it is my recommendation that the Logistic regression model is most appropriate for modeling the available data.

In Layman's Terms for the Committee:

The model that I have proposed for predicting the success or failure of students based upon the available data is called a Logistic Regression Analysis. At the basic level, this model makes predictions by plugging numeric representations the available data into an equation that results in a probability between 0 and 1 that a student will pass; If that probability is greater than ½, we predict success, and if the probability is less than ½, we predict failure. In our dataset, we have a number of features that describe each student: these are our *independent variables*, and we have whether or not each student passed: this is our *dependent variable*. The equation that we use assigns a different weight to each feature of our input, sums them together, and then adds a corrective constant. The difficult part of this problem, of course, is determining what these constants should be, and this is where machine learning comes into play. We start with equal weights use part of the available data to 'train' the model, which means solving a system of equations with calculus to find the weights that give us the best results.

Conclusion:

In order to get the best possible performance out of my Logistic Regression model, I performed two types of hyperparameter optimization, Grid Search and Randomized Search, each based on a the same set of parameters, and compared the results. The performance of a machine learning algorithm is dependent on a number of parameters outside of the dataset, and the purpose of hyperparameter optimization is to search through a set of potential values for these parameters to find the best-performing combination. Grid Search performs an exhaustive search of every combination of the provided parameter choices. While this is very powerful, it is slow, and much of the performance gains can be had much more quickly using Randomized Search, which instead performs a preset number of pseudo-random combinations of the given parameters. The un-optimized version of my Logistic Regression model achieved an F1 score of . The usefulness of hyperparameter search is demonstrated through the results of performing a randomized search for my model; it took 1.447 seconds and achieved an F1 0.7945 score on the test set. A Grid search optimization took over 8 seconds and found the same optimal parameter set, and thus resulted in the same F1 score of .7945.