Project 2 Writeup: Supervised Learning

1. Classification vs Regression

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

A: We're trying to predict a discrete outcome: whether a student will graduate or not, instead of a continuous outcome (like GPA) so it makes sense to use a **classification**-based supervised machine learning algorithm instead of a regression-based one.

2. Exploring the Data

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

A:

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%

3. Preparing the Data

See iPython Notebook

4. Training and Evaluating Models

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

A:

1. Decision Trees
2. Support Vector Machines
3. Gaussian Naïve Bayes

Q: What are the general applications of this model? What are its strengths and weaknesses?

A:

Decision Trees

- Time Complexity: $O(f*n^2\log(n))$ where f is the # of features and n is the sample size.
- Space Complexity: $O(f*n^2\log(n))$
- General Application: Decision Trees make a prediction about a variable's value given particular values for a set of features by following a series of "if-then-else" rules about the data.
- Strengths:
  - The series of rules used to classify the data can be visualized as a "tree" which makes it possible for a non-technical audience to follow the logic behind the model.
  - Can be validated using statistical tests.
  - Quick predictions that run in O(log(n)) time, where n points were used to train the tree.
- Weaknesses:
  - Unstable because variations in data can cause a completely different tree to be generated (which also leads to overfitting.)
  - Problem is NP complete if an optimal tree is desired.
  - Some problems like XOR are not easily expressed by a tree and may require exponential computing time to arrive at a reasonable answer.
- (Source: http://scikit-learn.org/stable/modules/tree.html)

Support Vector Machines (SVMs)

- Time Complexity: $O(f*n^3)$, where f is the # of features and n is the sample size.
- Space Complexity: $O(f*n^2)$
- General Application: SVMs (when used as SVCs for classification) are trained by representing a set of points in m-dimensional space (where m is the # of features) and projecting those points into (m+1)-dimensional space to find hyperplanes that separate the data into different categories.
- Strengths:
  - Relatively effective when the # of samples is limited.
  - Can use custom kernel functions that are better tailored to the data to project those points into higher dimensional space.
- Weaknesses:
  - Computationally inefficient (cubic time)
  - Prone to overfitting
- (Source: http://scikit-learn.org/stable/modules/svm.html)

Gaussian Naïve Bayes

- Training complexity: O(f*n), where f is the # of features and n is the sample size.
- Space complexity: O(f*n)
- General Application: Naïve Bayes assumes all of the features are independent of each other and the likelihood of each feature is assumed to be Gaussian (other variants may assume the features follow a Multinomial or Bernoulli distribution.) For every outcome observed in the training set, it calculates a probability for each possible value for each feature while training. And it then uses this info to predict the most likely outcome for a given set of features and their values.
- Strengths:
    - Extremely fast
    - Don't need a lot of data to train
- Weaknesses:
    - Assumes all features are independent; can be inaccurate when 2 or more features are highly dependent.
    - Doesn't have very many parameters to tune which means you'd have to try another model if you're not getting good accuracy.
- (Source: http://scikit-learn.org/stable/modules/naive_bayes.html)

Benchmarks:

Decision Trees	**Training Size 100**	**Training Size 200**	**Training Size 300**
Training time (s)	0.002	0.002	0.003
Prediction time (s)	0.000	0.001	0.001
F1 Score for training	1.0	1.0	1.0
F1 Score for test set	0.737704918033	0.710743801653	0.721804511278

SVM (SVC)	**Training Size 100**	**Training Size 200**	**Training Size 300**
Training time (s)	0.004	0.012	0.143
Prediction time (s)	0.001	0.002	0.06
F1 Score for training	0.96644295302	0.908496732026	0.905077262693
F1 Score for test set	0.794701986755	0.805369127517	0.791946308725

Gaussian NB	**Training Size 100**	**Training Size 200**	**Training Size 300**
Training time (s)	0.001	0.001	0.017
Prediction time (s)	0.001	0.001	0.002
F1 Score for training	0.861111111111	0.808510638298	0.778325123153
F1 Score for test set	0.601769911504	0.755905511811	0.741935483871

5. Choosing the Best Model

Q: Which model?

A:

(All 3 models managed to train in negligible time using the sample data set of 395 students but I'll assume we'll be training on a much larger data set when fully implementing it.) Out of the above 3 models, the one I picked is the **Decision Tree Classifier** based on the tradeoffs between performance (in training and prediction time) and accuracy (as measured by the F1 score which accounts for both Precision and Recall.)   I highly recommend making the investment in a Decision Tree Model now so you can target the right students for an intervention; this will also save the schools money and time in the long run by accurately identifying and not wasting time and money on interventions with students who are likely to graduate high school anyways.

Support Vector machines were not chosen despite their superior accuracy due to their slow performance. Even when benchmarking on a sample of 395 students, I found them to be the slowest although most accurate. The training and prediction times rise very rapidly from 100 to 300 samples. On the other hand, Naïve Bayes was not selected because the model doesn't have many parameters to tune in order to achieve a better prediction even though it also had a slightly better accuracy out of the box compared to Decision Tree Classifiers on a sample size of 200 and 300. Another problem with Naïve Bayes is that it assumes all features are independent of each other but this isn't necessarily true in real life (for example, a student who drinks a lot will not likely be in good health but Naïve Bayes assumes those two are independent.)

Q: About Decision Tree Classifiers

A:

Decision Trees classify data by asking a series of questions about the data that can all be unambiguously answered. This series of questions can be modeled like a flowchart that resembles a tree where you begin at the top and branch off in different directions further down depending on your answers to the questions (hence the name Decision **Tree**.) I'll walk you through an example of how a decision tree might work using the student data, though this isn't necessarily the most optimal tree out there.

The first question we might ask is if a student studies more than 5 hours a week. Next, we might ask whether the student had 15 or more absences. Answering these 2 questions effectively creates a 2 level Decision Tree that divides all students into 4 different categories. We could then decide that all students that study more than 5 hours and have less than 15 absences will graduate while those that study less than 5 hours and have 15+ will not graduate without intervention. The remaining 2 categories of students that have either 15+ absences or study less than 5 hours a week but not both simultaneously will be subjected to further questions along the decision tree such as their travel time to school, alcohol use, internet access at home, extracurricular activities.

Q: 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.

A: I picked the following parameters to tune for Decision Tree Classifier and their legal values:

- Max_depth: 1,2,3,4,5
- Min_samples_split: 2,3,4,5,6,7
- Max_features: ‘sqrt’, ‘log2’

Since Grid Search picked different optimal values every time, I’ve compiled a table below to find any patterns in the potential parameters before deciding on an optimal one:

Run	**Max Depth**	**Min Sample Split**	**Max Features**
1	3	7	log2
2	1	2	log2
3	1	2	log2
4	1	3	log2
5	1	7	log2
6	1	4	log2
7	1	4	log2
8	1	3	log2
9	1	5	log2
10	1	3	sqrt
Mean	1.2	4	N/A
Mode	1	3	log2

So the final parameters I’ve picked based on this table are:

Max_depth = 1, min_samples_split = 3, max_features=’log2’

Running DecisionTreeClassifier with these optimized parameters, we get:

Training set size: 300

```
Training DecisionTreeClassifier...

Done!

Training time (secs): 0.000

Predicting labels using DecisionTreeClassifier...

Done!

Prediction time (secs): 0.000

F1 score for training set: 0.811881188119

Predicting labels using DecisionTreeClassifier...

Done!

Prediction time (secs): 0.001

F1 score for test set: 0.774193548387
```

We improved the final F1 score by over 0.05 by tuning its parameters using GridSearchCV!