READ THE .IPYNB FILE, NOT THIS FILE

Project 2: Supervised Learning

1. Classification vs Regression

This is a classification problem. The output takes on class labels (classification), not continuous values (regression).

2. Exploring the Data

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%
Training set: 300 samples
Test set: 95 samples

4. Models

Support Vector Machines

Complexity

Computational:

Training: O(max(n,d) min(n,d)^2), where n is the number of points, and d is the number of dimensions [1].

Testing: depends on the kernel used.

Space: O(n^2) [2].

General Applications

- Data with large numbers of features
- Text and hypertext categorization
- Image classification

Strengths

- Effective when there are large numbers of features
- Decision function is memory-efficient

Weaknesses

- Does not directly give probability estimates - they have to be computed seperately & expensively
- If # features >> # samples, SVNs will perform poorly
- More applicable to binary problems

Why choose this model?

There are quite a few features in our data. I'm interested to see how it compares to other models. It is also a binary classification model - which is relevant to our data (multiclass SVM models also exist).

[1] Chapelle, Olivier. "Training a support vector machine in the primal." Neural Computation 19.5 (2007): 1155-1178.

[2] http://www.jmlr.org/papers/volume6/tsang05a/tsang05a.pdf

```
=========
===SVC===
=========
----------------------------------------
Training set size: 300
Training SVC...
Done!
Training time (secs): 0.011
Predicting labels using SVC...
Done!
Prediction time (secs): 0.007
F1 score for training set: 0.877272727273
Predicting labels using SVC...
Done!
```

```
Prediction time (secs): 0.002
F1 score for test set: 0.851612903226
------------------------------------------
Training set size: 100
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001
F1 score for training set: 0.87898089172
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001
F1 score for test set: 0.853658536585
------------------------------------------
Training set size: 200
Training SVC...
Done!
Training time (secs): 0.004
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003
F1 score for training set: 0.894915254237
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.826666666667
------------------------------------------
Training set size: 300
Training SVC...
Done!
Training time (secs): 0.011
Predicting labels using SVC...
Done!
Prediction time (secs): 0.006
```

```
F1 score for training set: 0.877272727273
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.851612903226
```

Forest of Randomized Trees

Complexity

Computational:

Training: O(tmn*log n) where t is the number of decision trees, m is the number of features, and n is the number of data points.

Testing: O(t*log(n))

Space: O(t*2^m)

General Applications

- Very flexible across predictive applications
- Not appropriate for descriptive applications

Strengths

- Fast to train
- Can make speed/performance tradeoff by tuning parameters
- Flexible across many/few features, many/few data points

Weaknesses

- Slow to make predictions once trained
- Results of learning are not descriptive/understandable

Why choose this model?

It's flexible and easy to tune speed/performance with the parameters available. Decision trees are a sensible way of seperating data into two sets.

Notes

Changing the number of estimators in the classifier by an order or magnitude (1, 10, 100) appeared to change the F1 score by ~3%.

```
=========
===RFC===
=========
------------------------------------------
Training set size: 300
Training RandomForestClassifier...
Done!
Training time (secs): 0.029
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.001
F1 score for training set: 0.882217090069
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.001
F1 score for test set: 0.847682119205
------------------------------------------
Training set size: 100
Training RandomForestClassifier...
Done!
Training time (secs): 0.020
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.001
F1 score for training set: 0.926174496644
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.001
F1 score for test set: 0.84076433121
------------------------------------------
Training set size: 200
Training RandomForestClassifier...
Done!
Training time (secs): 0.023
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.002
```

```
F1 score for training set: 0.91095890411
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.813793103448
------------------------------------------
Training set size: 300
Training RandomForestClassifier...
Done!
Training time (secs): 0.041
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.002
F1 score for training set: 0.878378378378
Predicting labels using RandomForestClassifier...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.838709677419
```

K Nearest Neighbours

Complexity

Computational:

Training (when pre-processing k): O(n*m) where n is the number of data points and m is the number of features

Testing: O(n*m)

Space: O(n*m)

General Applications

- Economic forecasting [1]
- Breast cancer diagnoses [2]

Strengths

- Simple and easy to implement
- Nearest Neighbours is used for clustering

Weaknesses

- Does not 'learn' anything from training data - just uses the entire set of training data for classification
- Slow for large samples
- Changing K can change class labels
- Not robust to noisy data
- Does not give probabilistic output

Why choose this model?

I'm curious how a type of model more often used for clustering would perform at prediction.

[1] http://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf

[2] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2243774/

```
=========
===KNN===
=========
------------------------------------------
Training set size: 300
Training KNeighborsClassifier...
Done!
Training time (secs): 0.002
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.013
F1 score for training set: 0.849765258216
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.040
F1 score for test set: 0.813793103448
------------------------------------------
Training set size: 100
Training KNeighborsClassifier...
Done!
Training time (secs): 0.002
```

```
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.002
F1 score for training set: 0.838709677419
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.802631578947
------------------------------------------
Training set size: 200
Training KNeighborsClassifier...
Done!
Training time (secs): 0.001
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.005
F1 score for training set: 0.863945578231
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.8
------------------------------------------
Training set size: 300
Training KNeighborsClassifier...
Done!
Training time (secs): 0.001
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.010
F1 score for training set: 0.849765258216
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.003
F1 score for test set: 0.813793103448
```

5. Choosing the Best Model

- 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?
- 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).
- 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 F_1 score?

Best Model

K Nearest Neighbours is the model I would recommend. Predictive performance is the deciding factor of model choice, as long as the time and space complexities don't make the model unreasonable to use. Here is a summary of the model performances on the whole dataset:

K Nearest Neighbours

- Training time: 0.001 sec
- Prediction time: 0.007 sec
- Training F1 Score: ~0.87
- Testing F1 Score: ~0.80

Forest of Randomized Decision Trees

- Training time: 0.011 sec
- Prediction time: 0.010 sec
- Training F1 Score: ~0.89
- Testing F1 Score: ~0.77 - 0.80

SVC

- Training time: 0.001 sec
- Prediction time: 0.007 sec
- Training F1 Score: ~0.87
- Testing F1 Score: ~0.78 - 0.8

K Nearest Neighbours performs best at Testing F1 score. Not only is it always higher than the other methods, but it is consistent given the same training/test split- unlike the others which vary in performance every execution on the same training/test data split. KNN also has closer Training and Testing F1 scores - which indicates that it is not overfitting as much as the other models.

KNN has significant disadvantages which don't appear to prevent from it being the best choice. The prediction time will scale quickly with data points. In this case, there were 300 training points, ~100 testing points, and x features. Prediction time on my computer was 0.007 sec = O(x *300)* 100 which means each prediction will take about 0.00000023 * # training points. If there are 10000 students that took a particular course in the past, and you have to make predictions on that course for 500 new students, it will take ~1.17 sec to perform all the predictions. This appears to be tolerable. If this dataset was a few orders of magnitude larger, KNN would be unreasonable.

Finally, we don't seem to need to understand how the model is making predictions (we don't care about descriptiveness), so KNN does not fail us in that respect.

How the Model Works

During training, KNN simply stores each data point (the student and all their properties) and the class it belongs to (whether the student passed or failed). When predicting what class a new data point should belong to (whether or not a new student will pass), KNN finds the top k most similar data points (where k is some number we chose beforehand). The class we predict for the new data point is the most commonly found class in the k data points we just found. The way we define how 'similar' any two data points are can be customized, but there are some commonly-used ways to choose from.

Final Score

~0.80

Surprisingly, GridSearchCV (consistently) picked a set of parameters that performed marginally worse than the default parameters. GridSearchCV must have performed a different split on the training data to test which one performs best (so the training data used to train wasn't the exact same for the grid search vs trial with default

```
------------------------------------------
Training set size: 300
Training KNeighborsClassifier...
Done!
Training time (secs): 0.001
```

```
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.007
F1 score for training set: 0.803493449782
Predicting labels using KNeighborsClassifier...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.828947368421
KNeighborsClassifier(algorithm='kd_tree', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=30, p=2,
           weights='uniform')
0.801677746122
```