Project 2: Supervised Learning

Building a Student Intervention System

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?

```
*This is a classification problem as we are modeling onto a discrete re
sult space [pass|fail]
```

2. Exploring the Data

Let's go ahead and read in the student dataset first.

*To execute a code cell, click inside it and press **Shift+Enter**.*

```
Student data read successfully!
```

Now, can you find out the following facts about the dataset?

- Total number of students
- Number of students who passed
- Number of students who failed
- Graduation rate of the class (%)
- Number of features

*Use the code block below to compute these values. Instructions/steps are marked using **TODO**s.*

```
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%
```

3. Preparing the Data

In this section, we will prepare the data for modeling, training and testing.

Identify feature and target columns

It is often the case that the data you obtain contains non-numeric features. This can be a problem, as most machine learning algorithms expect numeric data to perform computations with.

Let's first separate our data into feature and target columns, and see if any features are non-numeric.
Note: For this dataset, the last column (`'passed'`) is the target or label we are trying to predict.

```
Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu',
'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'study
time', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'n
ursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime',
'goout', 'Dalc', 'Walc', 'health', 'absences']
Target column: passed

Feature values:-
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...
1	GP	F	17	U	GT3	T	1	1	at_home	other	...
2	GP	F	15	U	LE3	T	1	1	at_home	other	...
3	GP	F	15	U	GT3	T	4	2	health	services	...
4	GP	F	16	U	GT3	T	3	3	other	other	...

5 rows × 30 columns

Preprocess feature columns

As you can see, there are several non-numeric columns that need to be converted! Many of them are simply `yes`/`no`, e.g. `internet`. These can be reasonably converted into `1`/`0` (binary) values.

Other columns, like `Mjob` and `Fjob`, have more than two values, and are known as *categorical variables*. The recommended way to handle such a column is to create as many columns as possible values (e.g. `Fjob_teacher`, `Fjob_other`, `Fjob_services`, etc.), and assign a `1` to one of them and `0` to all others.

These generated columns are sometimes called *dummy variables*, and we will use the `pandas.get_dummies()` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html?highlight=get_dummies#pandas.get_dummies) function to perform this transformation.

```
Processed feature columns (48):-
['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R',
'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstatus_
T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other',
'Mjob_services', 'Mjob_teacher', 'Fjob_at_home', 'Fjob_health', 'F
job_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'rea
son_home', 'reason_other', 'reason_reputation', 'guardian_father',
'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'f
ailures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout',
'Dalc', 'Walc', 'health', 'absences']
```

Split data into training and test sets

So far, we have converted all *categorical* features into numeric values. In this next step, we split the data (both features and corresponding labels) into training and test sets.

```
Training set: 300 samples
Test set: 95 samples
```

4. Training and Evaluating Models

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

- What are the general applications of this model? What are its strengths and weaknesses?
- Given what you know about the data so far, why did you choose this model to apply?
- Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

Produce a table showing training time, prediction time, F_1 score on training set and F_1 score on test set, for each training set size.

Note: You need to produce 3 such tables - one for each model.

Support Vector Machine:

```
* Used in: Image claficiation, text and hypertext classification, class
ification of proteins in medicine, handwriting recognition.

* SVMs are an ideal candidate and they are not prone to overfit on smal
l datasets as they will always try to maximize the margin.

* If many features are used, we might run into overfitting issues with
SVMs unless the number of samples is significantly higher than the numb
er of features. In this case we have many more samples than features so
this should not be a problem.

* Can only work on linearly separable data, but with an appropriate ker
nel this limitation can be easily solved.
```

Naive Gaussian:

```
* Used in: text classification like spam filters, automatic medical dia
gnosis.

* Is resistant to overfitting on small datasets.

* Is very simple and fast to train.

* On the down side, if the features are not as independent as they see
m, the classifier will run into problems as Naive Bayes makes strong in
dependence assumptions.

* This dataset is small and contains a lot of features, making Naive Ba
yes' independence from dimenasionality very useful.
```

DecisionTreeClassifier:

```
* Used in: agriculture, astronomy, choosing impaltable devices in biome
dical engeneering, 3D object recognition.

* Are very easy to visualize as a tree structure is very easy to unders
tand.

* Simple algorithms mean it is realtively fast to train.

* On the down side, with limited data they are prone to overfit and com
mit too much to the training data.

* The features seem to be independent, making decision trees a good can
didate.
```

RandomForestClassifier:

```
* Random forests have many of the same advantages of Decision Trees out
lined above.

* Random forests try to solve the issue of overfititng by using multipl
e trees at training time. But they can still suffer from overfitting.

* They take a long time to train as they have to go through multiple tr
ees, which is bad in a situtation with limited resources.
```

```
{'RandomForestClassifier': {'training_time': 0.02227687835693359
4}, 'GaussianNB': {'training_time': 0.001689910888671875}, 'SVC':
{'training_time': 0.010936975479125977}, 'DecisionTreeClassifier':
{'training_time': 0.004102945327758789}}
```

	training_time	**train_f1_score**	**test_f1_score**
DecisionTreeClassifier	0.004103	1.000000	0.640000
GaussianNB	0.001690	0.803783	0.763359
RandomForestClassifier	0.022277	0.995098	0.708661
SVC	0.010937	0.876068	0.783784

		training_time	train_f1_score	test_f1_score
DecisionTreeClassifier	**100**	0.000729	1.000000	0.787879
	200	0.001393	1.000000	0.748092
	300	0.002016	1.000000	0.634146
GaussianNB	**100**	0.000776	0.325581	0.404762
	200	0.000774	0.828571	0.753846
	300	0.000818	0.803783	0.763359
RandomForestClassifier	**100**	0.021202	0.979310	0.800000
	200	0.022073	0.996390	0.794118
	300	0.022813	0.987835	0.753846
SVC	**100**	0.001297	0.904459	0.768212
	200	0.003199	0.890323	0.797297
	300	0.006913	0.876068	0.783784

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?

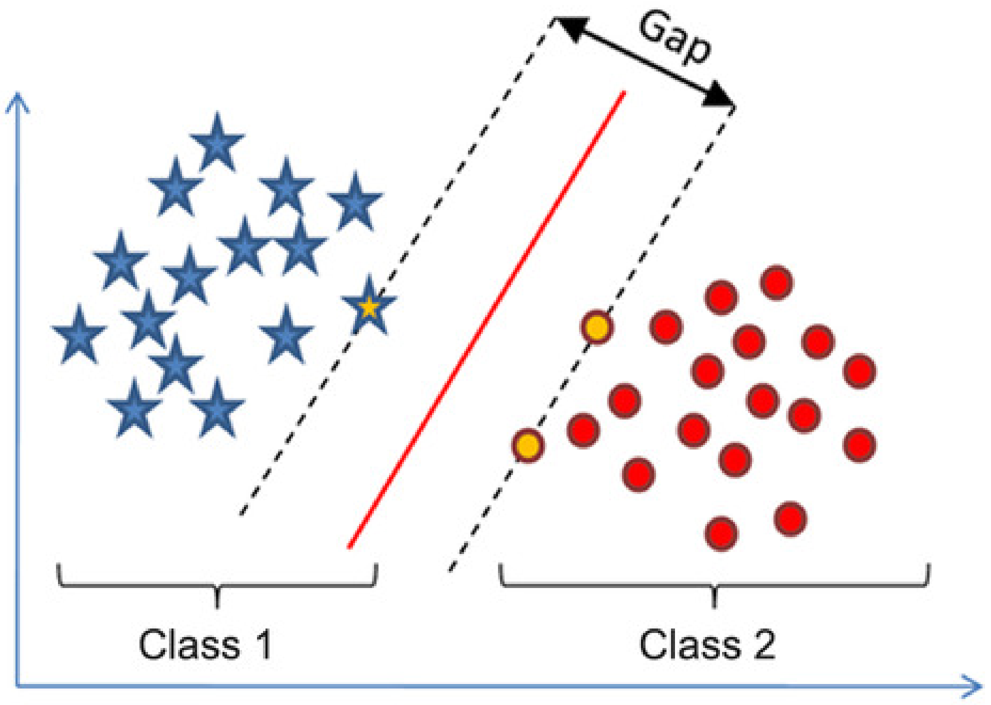
There are two candidate models after running the "un-tuned" comparasin. Both the untuned Naive Bayes and SVM are quick to train and achieve much higher F1 scores than the rest of the tested clasifiers. To choose between the Naive Byes and the SVM we will have so comprimse on a tradeoff. The Naive Bayes trains increibly quickly, taking only 0.000798s on the largest training set. This is almost an order of magnitude less than the SVM which takes 0.006496s.

But we not only care about speed, the F1 score tell us how well the model is performing and here the clear winner is the SVM scoring better than Naive Bayes even on the smallest trainign set (0.404762 vs 0.768212). If fact, the SVM perfomrs better with the smalles training set than the Naive Bayes with the largest trainig set (0.763359 vs 0.768212). Additionaly, Naive Bayes allows for less cutomization and tuning than the SVM.

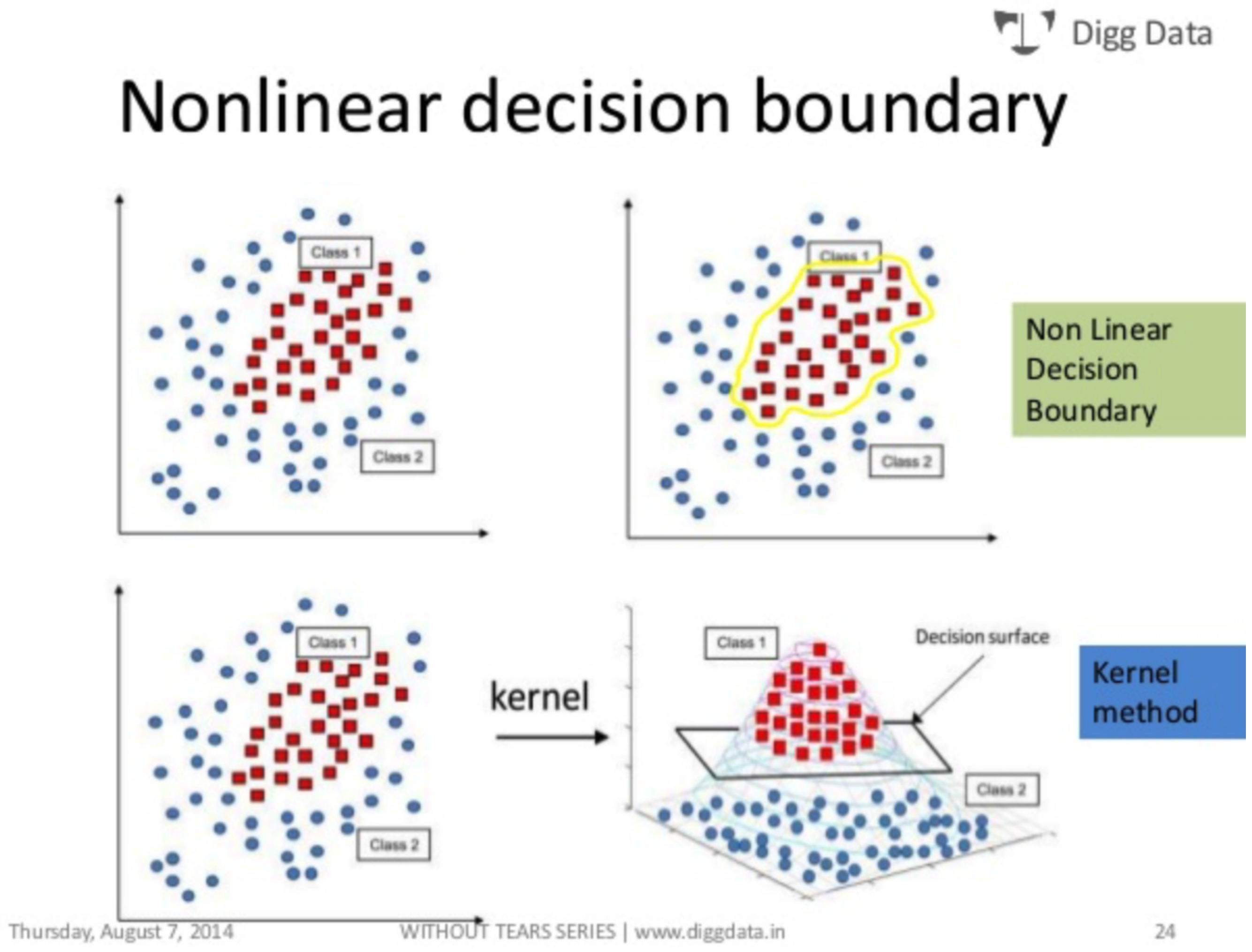
If resources are very very constrained, then Naive Bayes becomes a candidate, but everything points to SVM being a better investment, even if the cost is a bit higher, the goal of the program is to reach a 95% graduation rate, it's a high goal compraed to the curent 67.09% so better accuracy will be very helpful in identifiying the correct students. With all this in mind the clear winner is the SVM.

How do Support Vector Machines work?

We are trying to classify students into pass and fail. In its simplest form, a SVM will find the best line to separate and create the "classes" of data. In our case we can imagine separating between students who pass and students who fail.



The example in the fisrt diagram is a very simplified form of the problem where the data is easy to separate, but SVMs have "tricks" that allow us to use them even when the data appears to be "unseparable". By transforming the data using a kernel (without modifiying its meaning), SVMs allow us to separate data that seems impossible to separate linearly.



```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=1, gamma='auto', kernel='po
ly',
  max_iter=-1, probability=False, random_state=None, shrinking=Tru
e,
  tol=0.001, verbose=False)
Train F1 Score: 0.828865979381
Test F1 Score: 0.794520547945
All F1 Score: 0.820919175911
```

References

- http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/ (http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/)
- http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html (http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)
- http://diggdata.in/post/94066544971/support-vector-machine-without-tears (http://diggdata.in/post/94066544971/support-vector-machine-without-tears)
- https://en.wikipedia.org/wiki/Support_vector_machine (https://en.wikipedia.org/wiki/Support_vector_machine)
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier (https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
- http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.html (http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.html)