Student Intervention

1. Classification vs Regression

This is a classification problem, as we are aiming to predict what is the likelihood of the student to fail so that we can intervene at correct time so as to help him out. Moreover, this is a binary classification problem with two classes fails / pass.

2. Exploring the Data

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**

Use the code block provided in the template to compute these values.

3. Preparing the Data

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

- Identify feature and target columns
- Preprocess feature columns
- Split data into training and test sets

Starter code snippets for these steps have been provided in the template.

The code has been written and added in the ipython notebook

4. Training and Evaluating Models

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

Naive Bayes

This is very quick to train and predicting because it is the simplest model which can be used in order to predict thing very fast as its complexity is O(cd + nd) where c is the number of classes and the n is the number of data points and d is the number of features. It is best used in text classification problems like spam email detection where we need predictions very fast, face recognition etc. But being very simplistic it assumes that all the features are independent of each other thus the prediction results is not very accurate. But it even it does not overfit the data. F1- Score increases with increase in number of training data points.

Advantage:

1. Simple and Fast
2. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data

Disadvantage:

1. It has strong feature independence assumptions
2. If you have no occurrences of a class label and a certain attribute value together then the frequency-based probability estimate will be zero. Given Naive-Bayes' conditional independence assumption, when all the probabilities are multiplied you will get zero and this will affect the posterior probability estimate.
3. If used in classification problem with a small data set, precision and recall will be very low.

Training Data	100	200	300
Training Time (sec)	0.001	0.001	0.001
Prediction Time (Test)(sec)	0.001	0.001	0.001
F1 - Score (Training)	0.494117647059	0.810606060606	0.785542168675
F1 - Score (Testing)	0.408602150538	0.75	0.791044776119

Random Forest Classifier

With fairly small number of data points random forest can easily be used irrespective of its high time complexity. This is best suited for the problems where features are very likely to be dependent on each other or can be explained through trees thus using an ensemble of tree is a good approach. As it is evident from the following table that the training is very overfitted with less number of data points thus, as it assumes that all the features are mutually exclusive from each other which is not the case all the time, thus resulting in high F1- Score with less training data where it is more likely to overfit the data because of mutually exclusive assumption.

Advantage:

1. The Random Forests algorithm is a good algorithm to use for complex classification tasks. The main advantage of a Random Forests is that the model created can easily be interrupted.
2. No need of tunning of number of hymerparameters as in SVMs.
3. Can easily overfit

Distadvantage:

The main limitation of the Random Forests algorithm is that a large number of trees may make the algorithm slow for real-time prediction i.e. online predictions is really slow for realtime application.

Training Data	100	200	300
Training Time (sec)	0.045	0.022	0.037
Prediction Time (Test)	0.001	0.001	0.001
F1 - Score (Training)	1.0	0.992307692308	0.992405063291
F1 - Score (Testing)	0.629921259843	0.755905511811	0.832116788321

Support Vector Machines

Though this is the most computationally expensive algorithm as it has cubic time complexity, but because of its good classification results where we separable problems can easily be classified with high accuracy such as bioinformatics classification and image classification. As it comes with a kernel trick which can be used to convert non linear problem to linearly separable problem and thus finding the optimal hyperplane between the classes, but it seems to overfit if too many features were given, as in our case we have just 30 features therefore it performed really well. With the increase in the number of records the F1- Score for training and testing and training and testing time is increasing because the time complexity is a function of O(n^3) for training.

Advantage:

1. High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space.
2. Especially popular in text classification problems where very high-dimensional spaces are the norm.
3. Have good generalization.

Disadvantage:

1. Memory-intensive, hard to interpret, and kind of annoying to run and tune
2. Perhaps the biggest limitation of the support vector approach lies in choice of the kernel and selection of the kernel function parameters.

Training Data	100	200	300
Training Time	0.064	0.003	0.010
Prediction Time (Test)	0.001	0.001	0.002
F1 - Score (Training)	0.882352941176	0.881355932203	0.845986984816
F1 - Score (Testing)	0.774647887324	0.783783783784	0.833333333333

5. Choosing the Best Model

The best model of choice is Support Vector Machine. The other algorithms trailed performed poorly with increasing training set sizes compared to SVM.

1. The Gaussian Naive Bayes had a very fast and simple training and predicting time, but yielded a lower F1 score compared to SVM and even they do not have a good tuning potential.
2. The Random Forest Classifier had a much longer training time than SVM as a function of the number of trees and also has a lower F1 score.
3. SVM has a costly training time where the training time essentially doubles with each additional hundred dataset, it has a high performance that can handle large datasets well that are also unbalanced. SVMs can employ the use of kernels to fit the data in a higher dimensional space too using **kernel tricks**.

Support Vector Machines are based on the concept of decision lines that define decision boundaries. i.e. it involves finding the hyperplane (line in 2D, plane in 3D and hyperplane in higher dimensions. These decision boundaries as called as **HyperPlanes**. A decision line is one that separates between different sets of objects. In other words, given labeled training data as is in this supervised learning case, the algorithm outputs a clear divide that categorizes new examples. SVM chooses the best decision line or divide where the distance between that line and the nearest observations of differing classes are the largest. This is formally known as achieving the largest **margin**.

SVM cannot just classify the linear data but can also deal with highly dimensional data where using the **kernel trick** the non-linearly (3D) separable problem space is converted to 2D linearly separable problem.

From the following picture this can be easily depicted that the optimal line that can easily classify the two class labels is the one passing thorugh the center i.e **wx – b=0.**

The data points that kind of "support" this hyperplane on either sides are called the "**support vectors**".

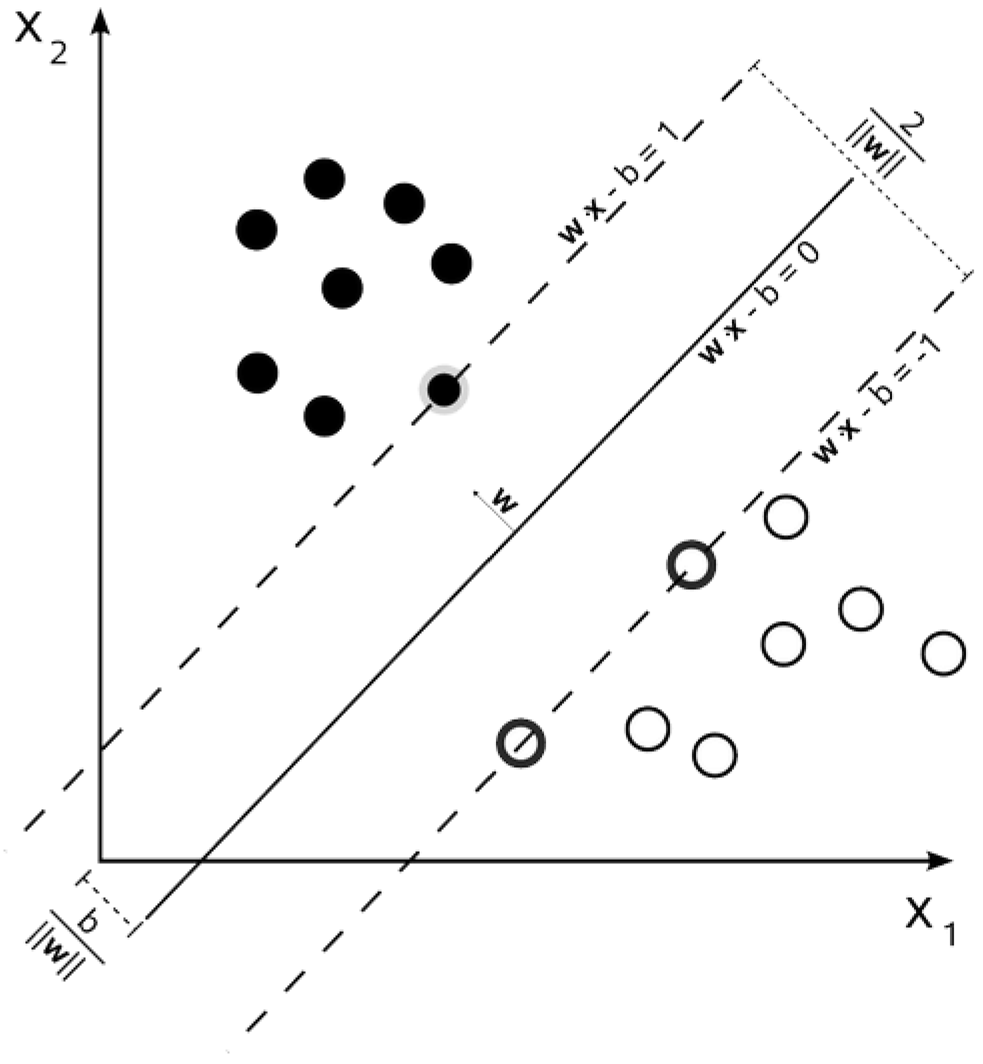


Figure 1. Finding the optimal margin using the support vectors

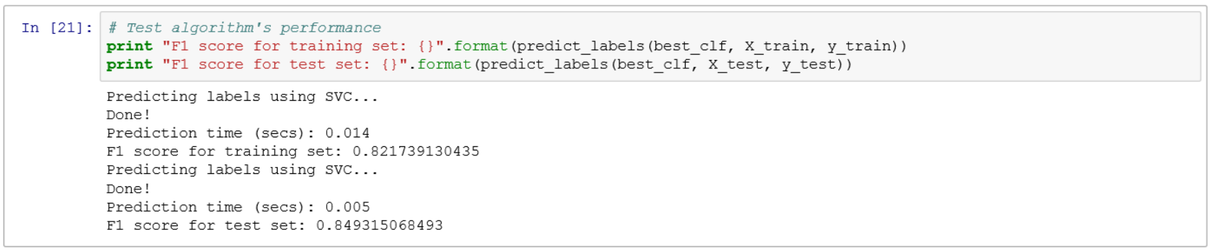
Given the ambitious goal of reaching a 95% graduation rate, this processing time is a small price to pay.

In our case as we can see there is not much difference in time complexity as we have just 30 features and 300 training data points thus, we can use SVC for the particular problem. As the F1- Score training and testing quite impressive and quite evidently increasing too which is better than random forest (testing) though random forest has good training F1- Score which indicates overfitting thus lead poor F1 Score (testing), whereas even Naive Bayes even do not have such a good F1 - Score as compared to SVC. Therefore, SVC is the best choice.

As SVC have 2 hyperparameters that can be tuned as well i.e. gamma and c for which because of ipython kernel limitations kernels cannot be tuned.

GridCV Search and Stratified ShuffleSplit has been used because the data is small and unbalanced.

What is the model's final F1 score? : `0.821739130435`



```
In [21]: # Test algorithm's performance
         print "F1 score for training set: {}".format(predict_labels(best_clf, X_train, y_train))
         print "F1 score for test set: {}".format(predict_labels(best_clf, X_test, y_test))
```

```
Predicting labels using SVC...
Done!
Prediction time (secs): 0.014
F1 score for training set: 0.821739130435
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005
F1 score for test set: 0.849315068493
```

References:

1. http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/
2. http://www.svms.org/disadvantages.html
3. http://scikit-learn.org/stable/modules/svm.html
4. https://www.quora.com/What-does-support-vector-machine-SVM-mean-in-laymans-terms