P2: Building an Intervention System
By Himanshu Dongre

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?

Ans: Here we want to identify students who need early intervention by learning on student data which provides previous student activities and whether they passed or failed. Thus we can identify weak students by dividing them based on their activities into the likelihood of whether they will pass or fail. Since we are trying to classify students into 2 discrete classes (pass or fail), it seems to be a classification problem.

2. Exploring the Data

Can you find out the following facts about the dataset?

Ans:

- **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**

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

Ans: Done in python notebook file

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?

Ans: I chose 4 models to model the given student performance data for this classification problem.

1. Decision Trees: Used for classification and regression when number of features are less. However, it works better in classification setting.

Strengths:

-Natural tendency to divide data into parts

-Easy to use

-Graphically intuitive

Weaknesses:

-Prone to over fitting with lots of features

2.Ensemble Random Forest Classifier: Used to add complexity to existing models to reduce over fitting and increase predictive power of individual classifiers. Used in classification and regression

Strengths:

-Easy to introduce more complexity to existing classifiers

-Can also be used to combine more than one classifiers

Weaknesses:

-Adds more complexity hence more computation cost

-Requires more storage

3.Naive Bayes: Again used in both classification and regression

Strengths:

-Easy to implement and inference is cheap

-Does well with big feature spaces

-Does better than most algorithms where the noise in the dataset is considerably high.

Weaknesses:

-Problems containing multiple phrases with different semantics cannot be interpreted well with Naïve Bayes

4.Support Vector Machines-Can be used in both classification and regression problems.

Strengths:

-Works well in complicated problems where there is no clear linear separation in the dataset.SVM can use their kernel trick to lift the data in higher dimensions to deal with non linearly separable data.

Weaknesses:

-Does not work well with large datasets(slow)

-Does not work well when dataset contains lots of noise

- Given what you know about the data so far, why did you choose this model to apply?

  Ans:

  Decision Trees: Decision Trees while generally used for both classification and regression problems, tend to do pretty well in such classification problems as their natural tendency is to divide data into parts by asking questions. Less complex so should be very time efficient.

  Ensemble Random Forest Classifier: I added Ensemble Random forest classifier as I liked the idea of how decision trees worked with the data, however were too simple. Random forest classifier gave a nice way to add complexity to simple decision trees to handle over fitting and increase its predictive power.

  Naïve Bayes: Since we are looking for algorithms with time and space constraints therefore I added Naïve Bayes to the list. Its a simple algorithm and cheap with time and space complexity.

  Support Vector Machines: Since the dataset here is very small, I thought SVM is a good candidate for this problem. As it introduces some complexity to the model to catch small subtle changes in attributes which can help to classify data.

- Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

  Ans: Done in python notebook

- Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

  Ans:

Decision Trees	**Training set size**		
	100	**200**	**300**
Training time (secs)	**0.001**	**0.001**	**0.005**
Prediction time (secs)	**0**	**0**	**0**
F1 score for training set	**1**	**1**	**1**
F1 score for test set	**0.710743802**	**0.772727273**	**0.74796748**

Ensemble Random Forest Classifier	Training set size		
	100	200	300
Training time (secs)	0.02	0.018	0.02
Prediction time (secs)	0.001	0.001	0.001
F1 score for training set	0.986666667	0.996491228	0.995145631
F1 score for test set	0.77037037	0.753623188	0.753846154

Naïve Bayes	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.002
Prediction time (secs)	0	0.001	0
F1 score for training set	0.598130841	0.833333333	0.808823529
F1 score for test set	0.426966292	0.768115942	0.75

Support Vector Machines	Training set size		
	100	200	300
Training time (secs)	0.001	0.004	0.009
Prediction time (secs)	0.001	0.001	0.002
F1 score for training set	0.875739645	0.881987578	0.869198312
F1 score for test set	0.766233766	0.765100671	0.75862069

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

5. Choosing the Best Model

Based on the experiments you performed earlier, in 2-3 paragraphs explain to the board of supervisors what single model you choose as the best model. Which model has the best test F1 score and time efficiency? Which model is generally the most appropriate based on the available data, limited resources, cost, and performance? Please directly compare and contrast the numerical values recorded to make your case.

Ans:

Based on the experiments performed earlier it seems that Decision Trees, due to their simplicity were most time efficient. However, it must be noted that all models performed quite well in case of time efficiency and the time difference is so insignificant that it cannot be a factor to choose the best model for this problem. So the model in this case should solely be chosen based on the F1 scores that they produce.

In case of F1 score it can be seen that Decision trees did perfect job on classifying training data correctly and hence producing best F1 score possible for training set i.e. 1.0. However, it also means that Decision trees

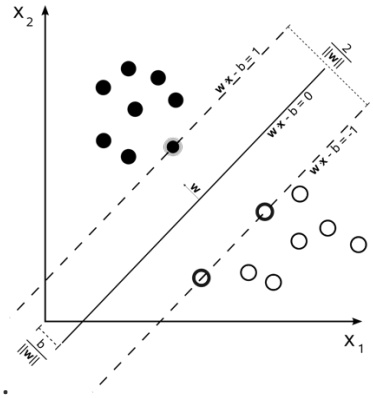
believed the training data too much as it was not able to produce same 1.0 F1 score for testing set. In terms of the testing F1 scores it seems that SVM produced the best F1 score (0.75862069) for the most training examples (300). But again the F1 scores of almost all the models are almost same for 300 training examples. Also it seems that Decision trees got the best F1 score for testing set at 200 training examples (0.772727273).

Considering all this one can say that Decision tree is the best model for this problem. However, I chose SVM as the best model for this problem. While decision trees performed well, there were significant fluctuations in testing F1 score with varying training data sizes. Same can be seen in other models that I used for this problem. However, I found that SVM's F1 score were more consistent and changed very little with varying training dataset sizes. Also the F1 scores produced by SVM for testing data were very near to what the decision trees could produce at its best. Therefore, I chose SVM's SVC as the best model to describe the data in this problem.

In 1-3 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 learn to make a prediction).

Ans:

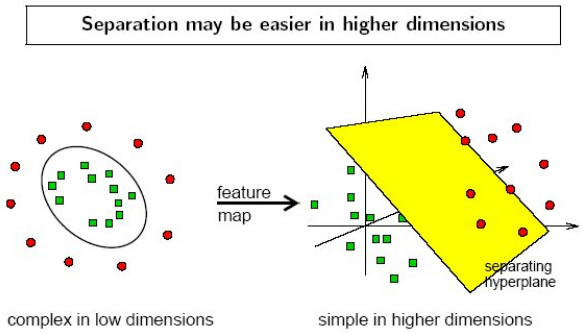
I chose SVM's Support vector classifier (SVC) as my final model. A basic linear SVM tries to classify dissimilar data by drawing a line between them. For ex: In the below dataset there are 2 kinds of data black and white.SVM draws a line (solid line in this case) between black and white data separating them into 2 parts.



.

While many lines can be drawn which separate the dataset, SVM chooses the line in the middle which maximizes margin i.e. it tries to maximize the distance between the line being drawn and data points which are nearest to the line (shown in dark borders in the above image). These data points which affect the line being drawn are called as support vectors which gives algorithm its name i.e. Support Vector Machines.

Sometimes a dataset might not be easily separable in a lower dimension (X-Y axis) with just a line. However, in such cases SVM uses a clever mathematical trick which allows it to project the data in higher dimensions (e.g.-X-Y-Z axis).



See how data that seems complex to classify in a lower dimension separates easily in a higher dimension. As we project the data from a 2D(X-Y) space to a 3D space(X-Y-Z) we can see that the data can be separated using a simple plane (line in 2D becomes plane in 3D). This mathematical trick is also known as a kernel.

This way SVM is able to separate complex datasets which are non linearly separable.

Fine-tune the model. Use grid search 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 F1 score?

Ans: Fine-tuned SVM with grid search by trying different Kernels, C and Gamma Settings. With these parameters, the best model which grid search could produce is with kernel= rbf, C=10 and gamma=0.001. Thus the final F1 score that the tuned model can produce is 0.786206896552, which is slightly better that what the default SVM could produce (0.75862069)

Sources:

Images taken from wikipedia.com and dtreg.com