Student Intervention System

1. Classification vs Regression

This is a Classification problem given that we have a discrete output: does the student require early intervention or not.

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	67.00%

3. Preparing the data

This task can be found in the iPython notebook.

4. Training and Evaluating Models

The content of this section was extracted and adapted from the Machine Learning Nanodegree course, the *scikit-learn* documentation, and the following sources:

[1] Brandon Rohrer, "*How to choose algorithms for Microsoft Azure Machine Learning*". Link: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/

[2] Edwin Chen, "*Choosing a Machine Learning Classifier*". Link: http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/

- *DECISION TREES.*

Decision trees are one of the *"simplest"* models for classification (and regression) that we have learned so far and they are based on a set of simple decision rules which are created based on the training data. I choose this model basically because it will allow me to interpret the results and it will automatically perform some feature selection according to their relevance.

ADVANTAGES	DISADVANTAGES
They can be plotted, and they are easy to interpret. In our current problem, it would be interesting to understand the reasons why a student is failing.	They can easily overfit the data.
The prediction cost is very low; we just need to ask a series of simple questions.	They are instable; the tree can change for small variations in the training set. They are, therefore, not very robust to noise.
Requires little data preprocessing and can deal both with numerical and categorical data.	A global optimal solution cannot be guaranteed.

DECISION TREES

	Training set size		
	100	**200**	**300**
Training time (ms)	2	2	3
Prediction time (ms)	0*	0*	1
F1 score (training data)	1	1	1
F1 score (test data)	0,64	0,66	0,71

* even with 6 decimal points, the prediction time is so short that it cannot be estimated.

- *SVM*

The second model will be a Support Vector Machine. This method works both for classification and regression and I selected it because it can deal with many dimensions and it is very easy to implement. SVM also allow us to define custom kernels, thus giving us a lot of flexibility.

ADVANTAGES	DISADVANTAGES
Effective in high dimensions.	They do not perform well in large datasets. Training takes a lot of time.
It allows us to specify custom kernels.	They do not work very well with overlapping clusters.
It is memory efficient because it uses only a subset of the training points in the decision function.	Problem needs to be formulated as a 2-class classification task.

SVM

	Training set size		
	100	**200**	**300**
Training time (ms)	2	4	9
Prediction time (ms)	1	2	1
F1 score (training data)	0,87	0,84	0,87
F1 score (test data)	0,80	0,80	0,80

- *NAIVE BAYES*

In one of the lectures, it was mentioned that SVM's do not perform well when there is a lot of noise causing clusters to overlap. It was also mentioned that in those situations it would be better to apply a method as Naive Bayes given that it can also give us some estimation on the uncertainty of the results. I chose to employ Naive Bayes as the third and last method in order to compare its performance with respect to SVM and in order to cover both possible situations: well defined and easily separable clusters, and overlapping classes.

ADVANTAGES	DISADVANTAGES
If the independence assumption actually holds, the classifier will converge very fast, thus requiring less training data.	It assumes conditional independence between the features, thus losing accuracy when this does not hold.
Easy to implement	It cannot learn interactions or relations between the features.

NAIVE BAYES

	Training set size		
	100	**200**	**300**
Training time (ms)	2	2	1
Prediction time (ms)	2	1	0
F1 score (training data)	0,79	0,81	0,81
F1 score (test data)	0,63	0,73	0,75

5. Choosing the best model

According to the previous results, a Support Vector Machine would have the best performance achieving a F1 score on the testing data set of around 0.80. The Decision Tree and the Naive Bayes classifier present a very similar performance on the testing set, achieving a F1 score of 0.71 and 0.75 respectively. However, when we compare the time efficiency, we notice that the SVM would be quite expensive in terms of computational cost given the exponential growth of the training time with respect to the training set size (9ms for 300 samples).

In contrast, the other two methods are much more time efficient and present very low training and prediction times. Although the training time of a decision tree is slightly longer for large training sets (3ms for 300 samples) compared to a NB classifier (1ms for 300 samples), its prediction time is remarkably short and it was not even possible to estimate it using the available data.

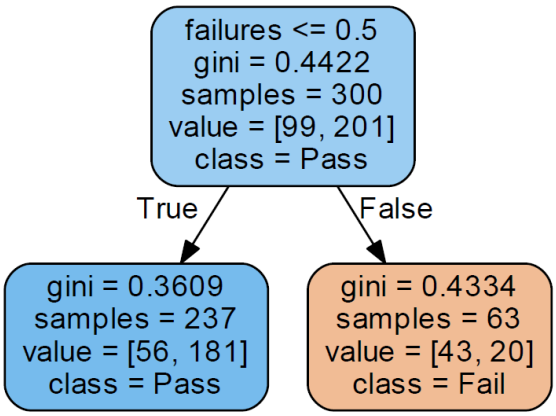
In my opinion, the model which should be implemented for this particular problem should be a Decision Tree. First, it presents the best time efficiency and would be the cheapest option in the long run. Second, although its performance is low in comparison to the SVM, the results indicate that there is some overfitting going on and it should therefore be possible to improve its

performance by properly tuning the model. Finally, even if the classification accuracy of a SVM is higher, its results will not provide the supervision board with any insights about the reason why a student should require early intervention. A Decision Tree, in contrast, would help us to understand which features are more relevant when determining the success of a student and will allow the school to focus on very specific problems when trying to help the students.

Decision Trees are schematic tree-shaped diagrams which, by means of very simple questions (i.e. does the student have access to internet at home? Is the student involved in a romantic relationship? Etc.), allow us to classify our data into different categories. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label is reached. Determining which questions to ask, however, is a key problem in decision tree classifiers and different methods have been proposed. In general, the best questions are those which lead to a homogeneous class distribution in the nodes. The larger the degree of purity after splitting, the better the class distribution.

Once the decision tree has been built, predicting whether or not a student will require intervention is quite simple. Starting from the root node, we answer the first question and follow the appropriate branch according to the given answer. This process is repeated until we get to a leaf node, where no more questions are presented and a class is assigned to the evaluated record.

After tuning the model, we ended up with the following decision tree:



	Training Set	Testing Set
F1 Score	0.76	0.82

As it can be seen, by tuning the model we ended up with a depth of 1. The results indicate that 'Failures' is the most relevant feature and it is enough to determine if a student will pass or fail. I tried to analyze larger trees but I was not able to find another relevant feature. Increasing the depth of the tree resulted only in overfitting to the training set.