Building an Intervention System

Project 2 Report for Machine Learning

Author: Matteo Piombo

Classification vs Regression

Goal of the model is to identify students who might need early intervention. This is a classification problem since we'll have to classify students between two well defined classes.

Our predictor will predict True or False given student's features:

- True: student will graduate -> No need for early intervention
- False: student will not graduate -> Needs early intervention

Data Exploration

Total number of students: 395
Number of features: 30*
Number of graduates: 265
Number of non-graduates: 130
Graduation rate of the class: 67.09%

* The number of features does not include the target label which is one, i.e.: a discrete value "yes" or "no" indicating if student successfully graduated.

Training and Evaluating Models

Decision Tree Model

At first I try a Decision Tree model, since the kind of data we have seems well suited for this type of model and the problem at hand.

Complexity: The decision tree training algorithm iterate over the features choosing each time the one that gives the *best split* among all the dataset. At each iteration it computes statistics over the possible feature dataset splits. In the worst case the complexity grows exponentially as the number of features and samples grows. Anyway it could stop pretty fast in case it finds a *good* split in a few iterations. Thus the actual complexity can vary hugely depending on the data. Limiting the maximum

depth for the decision tree will *hard limit* the number of iterations. Increasing the minimum number of samples in a *leaf* split could also reduce the training complexity, even though this parameter is typically used to prevent overfitting. Depending on how the data get splitted, every iteration may or may not need to iterate over the entire dataset. In the end we should consider that the entire training dataset is needed during training. Once the Decision Tree is trained, we no longer need the training data in order to make predictions/classifications. Prediction complexity will depend on the final decision tree, but can be considered roughly linear with the number of levels in the trained tree.

About the model: Decision Trees are generally used for supervised classification where data input is discrete. Variants of the decision tree algorithm can be used with continuous inputs in which case the decisions will be made over input inequalities (ex.: $x < 0$). Generally speaking this algorithm is not good at predicting smooth continuous curves. Although Decision Tree Regression algorithms are available. An important strength of this model is that it starts from the most important feature and optimises for the lowest number of levels. The implementation of a trained decision tree could be pretty fast and easily hardware accelerated since the prediction process involves simple binary decisions.

Why this model: I started with this model since we can argue that just a few features can be *precursors* of a fail to graduate. As a byproduct the decision tree can give a hint at which kind of intervention can better help the student, since we see from the trained tree *why* the student is predicted to fail. I started with a pretty low maximum tree depth.

Model Performance Metrics

Decision Tree	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.002
Prediction time (secs)	0.000	0.000	0.000
F1 score for training set	0.831	0.823	0.806
F1 score for test set	0.795	0.797	0.825

It appears there's just one most relevant feature and it is *failures*. Other features are picked up by the trained model on a random base among repeated training, keeping the max depth fixed to just three levels.

Gaussian Naive Bayes Model

Now I explore with the Gaussian Naive Bayes model. Since we've already seen that there is a feature that strongly dominate over the others, it might be that this model is not the best. Anyway let's give it a try.

Complexity: This model is based on statistical maximum likelihood theory. It will use all the training data in order to calculate a posteriori probability distributions and correlations. These calculations will grow linearly with the number of samples, since adding one sample will add one *point* to the same statistic calculations. Anyway in this case there is no possible *shortcuts* that can derive from the particular samples in the data (like it is for a Decision Tree). This algorithm will need to calculate over all the data in order to calculate statistics. Once trained the model will no more access training data and will only calculate the most probable label given the new sample and the previously calculated statistics. So prediction could be considered O(1).

About the model: The model applies Bayes rule in order to get the maximum likelihood label. Finally it will predict the labels based on the most probable label given the new sample and an a posteriori sample and label distribution calculated on the training data. In this case the accuracy of the estimated distribution will strongly depend on the number of features and training samples. In general, the more features, the more samples are needed. This model is less robust against outliers, since every outlier will bring its equal contribution to statistics. This is in contrast with the decision tree algorithm, where a small amount of outliers is easily accepted inside a leaf split.

Why this model: Since we have 30 features and 300 training samples, we could argue that this model will not have enough data to reach good performance. Anyway comparing this model with decision tree could validate this hypothesis.

Model Performance Metrics

Gaussian Naive Bayes	Training set size		
	100	200	300
Training time (secs)	0.001	0.005	0.004
Prediction time (secs)	0.001	0.001	0.001
F1 score for training set	0.803	0.777	0.77
F1 score for test set	0.713	0.752	0.73

Support Vector Machine Model

Finally I test a support vector machine model with a non linear kernel, maybe it could perform better than the decision tree.

Complexity: It uses all the data points. Complexity of this algorithm is cubic since all distances between the data points has to be evaluated and then filtered. Among the models tested so far it is the most computationally expensive. Once the model is trained the training dataset is not used anymore. Prediction complexity could be considered constant since it requires computation among the nearest supporting vectors.

About the model: This model explores all the distances between all the samples in order to find the best decision boundary. Given the maximum distance between two parallel hyperplanes separating two classes, the decision boundary is the hyperplane in the middle. This decision boundary will separate the two classes without fitting to much in favour of one or the other. Non linear decision boundaries can be found using different kernel functions. These kernel functions will add dimensionality to the dataset space without adding any information. A linear separation is then searched in the new dimensions.

Why this model: So far the decision tree is the best model, but decision trees are not good at finding smooth separations. Maybe a smoother separation could bring better performances. Thus I try a support vector machine with a non linear kernel.

Model Performance Metrics

Support Vector Machine	Training set size		
	100	200	300
Training time (secs)	0.003	0.06	0.014
Prediction time (secs)	0.002	0.002	0.003
F1 score for training set	0.847	0.853	0.857
F1 score for test set	0.838	0.851	0.859

Choosing the Best Model

The model having the best F1 score is the one using Support Vector Machine. Decision tree is close.
I choose the one based on Support Vector Machine since:

- it reaches the best F1 score
- even though it is more computationally expensive, we can argue that the number of students will not grow rapidly. Thus updating the model will continue to be computationally acceptable for a reasonable time. (I assume this model is meant to be used in a single school, and not scaled up to thousands of schools)

Anyway there might be good reasons to choose the decision tree model:

- F1 score is not that bad
- it can scale more easily since it is less computationally expensive
- it can be graphically represented which, in case of low depth, could help teachers understand why students need intervention.

Describing the Model in Layman's Terms

The Support Vector Machine model will look at the labeled data trying to find smooth coast lines that separate islands of data points having the same class label. Then it will retain just the data points (samples) needed to define the coastlines. These samples are the so called *support vectors*. The boundaries among different classes will be in the middle of the relevant coastlines. Training the model means making it find the best coastlines given the training data. Making predictions with the model means looking in which island's boundaries a given sample is. We can think of the island's boundaries as the line in the middle of the *river* separating them from other islands.

Model Tuning

The model fine tuned via grid search will be a Support Vector Machine having the following relevant parameters:

- kernel: rbf
- C: 0.9
- gamma: 0.6
- decision function: one vs. the rest

Tuned F1 Score

The tuned F1 score on the test set is approximately 0.8, pretty close to the first attempt.