P2 - Building a Student Intervention System

Michaël Lambé
mic0331@gmail.com

Goal : Model the factors that predict how likely a student is to pass their high school final exam, thus helping diagnose whether or not an intervention is necessary.
The model is developed based on a subset of the data and it will be tested against a subset of the data kept hidden from the learning algorithm, in order to test the model effectiveness on data outside the training set.

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?

Our problem is posed as a binary classification task where we refer to our two classes as 1 (passed) and 0 (failed).
The algorithm could then be used to predict if a sample belonged to one class or the other.

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
Number of features: 30
Graduation rate of the class: 67.09%
```

The dataset is very small with unbalanced classes (more people passed vs. failed)

- As we are dealing with a very small dataset with unbalanced classes, we have to use a special case of k-fold cross-validation that is "stratified k-fold cross validation", which can yield better bias and variance estimates, especially in cases of unequal class proportions like in our case. In stratified cross-validation, the class proportions are preserved in each fold to ensure that each fold is representative of the class proportions in the training dataset. ==> Each set contains approximately the same percentage of samples of each target class as the complete set.

- The standard value for k in k-fold cross-validation is 10,which is typically a reasonable choice for most applications. However, as we are working with relatively small training sets, it can be useful to increase the number of folds. If we increase the value of k, more training data will be used in each iteration with results in a lower bias towards estimating the generalisation performance by averaging the individual model estimates. However, large value of k will also increase the runtime of the cross-validation algorithm and yield estimates with higher variance since the

training folds will be more similar to each other. In this context we choose k = 15 as the number of partitions. Interesting discussion are available at "choice of k in k-fold cross validation" and "For k-fold cross validation, what k should be selected".

- As the class distribution is unbalanced, accuracy is considered a poor choice as it gives high scores to models which just predict the most frequent class. On the other hand, the F1 score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially as we have an uneven class distribution. Also, as we are interested in a score that conveys a balance between the precision and the recall, the F1-score is the ideal candidate. ==> Thus moderately good performance on both will be favoured over extremely good performance on one and poor performance on the other.

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.

Done (cfr. 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: What is the theoretical O(n) time & space complexity in terms of input size? 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 F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant. Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.
Note: You need to produce 3 such tables - one for each model.

F1 score : Precision (PRE) and recall (REC) are performance metrics that are related to true positive and true negatives rates. In practice, we use a combination of precision and recall, the so-called F1-score.

F1 = 2(PRE x REC)/(PRE + REC)

Model 1 : Logistic regression (regression algorithms)

- **Introduction**

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using the logistic function (cumulative logistic distribution = sigmoid function)

- **Applications of this model**

Logistic regression produce a binary decision. It is used to ascertain the probability of an event. This event is captured in binary format, i.e. 0 or 1.

- **Strengths**

- This model performs very well on linearly separable classes.
- Logistic regression is a simpler model than SVN and can be implemented more easily.
- The model can also be easily updated, which is attractive when working with streaming data.

- **Weaknesses**

- Logistic regression works well for predicting categorical outcomes. It can also predict multinomial outcomes. However, logistic regression cannot predict continuous outcomes
- Logistic regression tries to maximise the conditional like hoods of the training data, which makes it more prone to outliers than SVN.
- Logistic regression requires that each data point be independent of all other data points. If observations are related to one another, then the model will tend to overweight the significance of those observations.
- Logistic regression attempts to predict outcomes based on a set of independent variables, but logit models are valuable to overconfidence. That is, the models can appear to have more predictive power than they actually do as result of sampling bias.

- **Why choosing this model for the dataset?**

As our “y" variable is categorical, logistic regression seems appropriate in order to ascertain the probability of success or failure of a student

- **Outcome tables**

Logistic Regression	Training set size		

	100	200	300
Training time (secs)	0.007	0.006	0.005
Prediction time (secs)	0.001	0.001	0.000
F1 score for training set	0.9932	0.8763	0.8393
F1 score for test set	0.6215	0.7549	0.7647

Model 2 : SVM (Support Vector Machine)

- **Introduction**

  This model is an extension of the perceptron model. using perceptron algorithm, we minimised misclassification errors. However, in SVMs, our optimisation objective is to maximise the margin. The margin is defined as the distance between the separating hyperplane (decision boundary) and the training samples that are closest to this hyperplane, which are the so called support vectors.

- **Applications of this model**

  This model offer good prediction however it can be computationally complex to run, meaning the infrastructure to support such model could require large server.

- **Strengths**

  - less prone to outliers
  - SVM is capable of doing both classification and regression.
  - The model is able to capture complex relationships between datapoint.
  - Effective in high dimensional spaces.
  - Still effectives in cases where number of dimensions is greater than the number of samples.
  - Vesatile: different kernel functions can be specified for the decision function.

- **Weaknesses**

  - Require data transformation and resulting boundary plane are very difficult to interpret. this is why it's often called a black box.
  - Training time can be large as it's much more computationally intensive (kernel trick)
  - If the number of features is much greater than the number of samples, the method

is likely to give poor performances.

- SVMs do not directly provide probability estimates, these are calculated using an extensive five-fold cross-validation.

- **Logistic regression versus SVM**

  In practice, linear logistic regression and linear SVMs often yield very similar results. Logistic regression tries to maximise the conditional likelihoods of the training data, which makes it more prone to outliers than SVMs. The SVMs mostly care about the points that are closest to the decision boundary (support vectors). On the other hand, logistic regression has the advantage that it is a simpler model that can be implemented more easily.

- **Why choosing this model for the dataset?**

This model is interesting as it is a good way to identify if we have a dataset that is linearly separable or not.

In the notebook, model 2 is performed for a ‘linear' and ‘rbf' kernel. Clearly, the performance are better for the `rbf` kernel version indicating that the dataset is not linearly separable.

- **Outcome table (‘rbf' kernel)**

SVM (rbf kernel)	Training set size		
	100	200	300
Training time (secs)	0.001	0.004	0.007
Prediction time (secs)	0.003	0.005	0.003
F1 score for training set	0.9299	0.9103	0.9071
F1 score for test set	0.7885	0.7736	0.7662

Model 3 : Decision Tree

- **Introduction**

  Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG). In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure.

- **Applications of this model**

  This model is interesting when a visual representation of a decision situation is needed. It's a great communication tool. The branches of a tree explicitly show all those factors within the analysis that are considered relevant to the decision (and implicitly this that are not).
  More subtly, a decision tree generally captures the idea that is different decisions were to be taken then the structural nature of a situation (and hence of the model) may have changed dramatically.

- **Strengths**

  - Attractive model if we care about interpretability. The model breaks down the data by making decisions based on asking a series of questions.
  - Requires little data preparation
  - the cost of using the tree is logarithmic in the number of data points used to train the tree.
  - Able to handle both numerical and categorical data.
  - Able to handle multi-output problems
  - Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
  - Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
  - Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

- **Weaknesses**

  - In practice, we can result in a very deeps tree with many nodes, which can easily lead to overfitting. Thus, we typically want to prune the tree by setting a limit for the maximal depth of the tree.
  - Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
  - The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
  - There are concepts that are hard to learn because decision trees do not express

them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

- **Why choosing this model for the dataset?**

As explained previously, this model has been chosen primarily for it's representation power.

- **Outcome table**

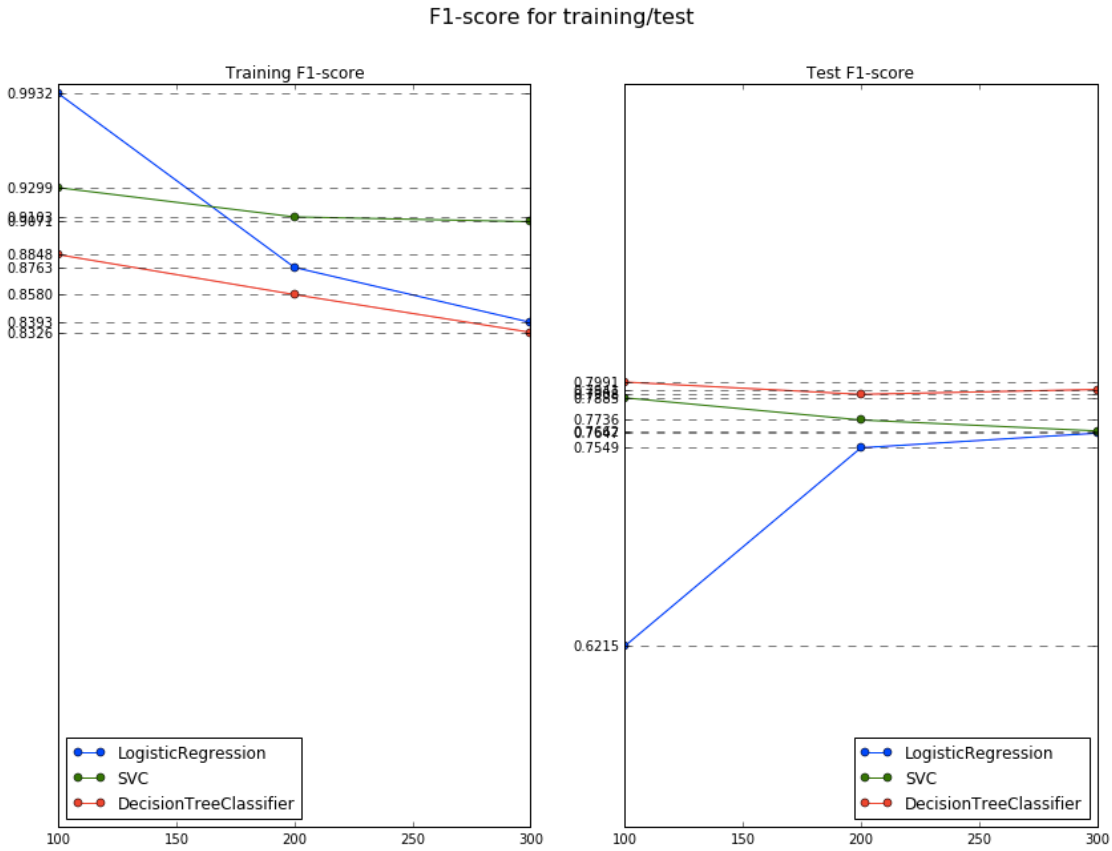
Decision Tree	Training set size		
	100	200	300
Training time (secs)	0.000	0.001	0.001
Prediction time (secs)	0.000	0.001	0.000
F1 score for training set	0.8848	0.8580	0.8326
F1 score for test set	0.7991	0.7898	0.7941

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 F1 score?

So far the three models experienced are giving similar performance in term of F1-score.
The following graph is showing a comparison of the F1-scores for each classifier for the training & test set with different size experienced in this project.
We can see clearly that DecisionTree is getting better results on the test set than on the training set.
LogisticRegression is getting much better results on the test set when the size of the set is increased. SVC on the other hand is getting lower F1-score on both the training and test set when the size increase.
Nonetheless, the magnitude of the differences between each classifier is very small. We

should not focus our judgment only on the F1-score. This is especially true due to the limited data for this project.



From an execution time point of view, we clearly see that decision tree is the most efficient. As the model is getting more complex, the training time and prediction time are both increasing (SVM being the model requiring the most time to execute)

With respect to the context of the problem, the decision tree model is chosen. Using a decision tree algorithm, we start at the tree root and split the data on the criteria that results to the largest information gain. This iterative process repeat the splitting procedure at each child node until the leaves are pure.

This model offer a good interpretability. For example, the model breaks down the data by making decisions based on asking a series of questions. One can investigate the break-down rules and eventually act on the important factor conducting students to failed. From a performance stand point, we can see that training time is not increasing a lot from each training set size increment. This means that the model can scale pretty well on reasonable machine. The model will not require major investment in computer power. Last but not least, if in a latter stage it is decided to add additional output (not just 1 or 0), the model will react and work correctly with such new setup. For example one can implement new criteria such as “passed”, “at risk”, “improving”, “failed” making class separation even more powerful !

Noticed that SVM is offering better scoring however, as the performance is a key factor to consider for this project, the choice has been made to ignore this model. Also, SVM model interpretability is not guarantee.

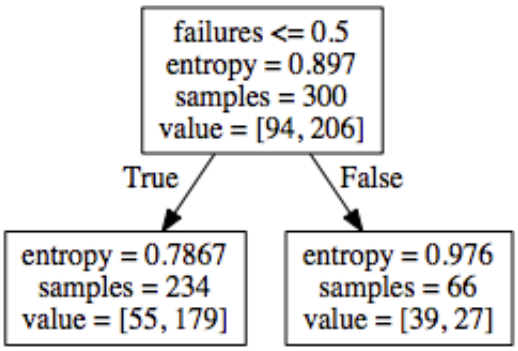
After running the selected model with the grid we end-up with an F1-score of `0.814/0.803` respectively for the training and test dataset.

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_
depth=1,

max_features=None, max_leaf_nodes=None, min_samples_leaf=10,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=1, splitter='best')
```

For illustration purpose, bagging has been implemented in order to introduce randomisation into the initial training dataset. More details about this procedure can be found in the notebook.
This step is not improving the model what so ever however, it could be a good way to reduce the variance of the estimator, obviously not in this dataset.

The tree can be easily represented (using Graphviz).



We noticed that the main feature used for splitting the data is the *failures* feature. This feature indicate the number of past class failures (n if 1<=n<3, else 4)
This clearly indicates that students who have history of class failure are very likely to fail in the future. Therefore, teachers should probably implement some support/remediation in order to help those students.

Interpreting the decision tree is easy, if a student get failures > 0.5 (meaning if it get at least a failure), it goes to the right leaf, otherwise to the left.
In the left leaf, we have 55 students in category 1 (class/target 0 ==> failure) and 179 in category 2 (class/target 1 ==> success) **==> 76.5% chance of success**
In the right leaf, we have 39 students in category 1 (failure) and 27 in category 2 (success) **==> 40.9% chance of success**

So clearly, more than one exam failure increase considerably the chance of the student to not succeed the final exam.