Building a Student Intervention System

Uirá Caiado

April 26, 2016

Abstract

There exists a push from educators and administrators to raise the likelihood of students successfully complete their programs. The aim of this project is to identify students who need intervention before they drop out of school. My goal is to use concepts from supervised machine learning to find the most effective model with the least amount of computation costs that identify such students.

1 Introduction

1.1 Some Background

As stated by Udacity[1] in the description for this project, as education has grown to rely more and more on technology, more and more data is available for examination and prediction. Logs of student activities, grades, interactions with teachers and fellow students, and more are now captured through learning management systems.

Within all levels of education, there exists a surge to help increase the likelihood of student success without watering down the education or engaging in behaviors that raise the probability of passing metrics without improving the actual underlying learning. Graduation rates are often the criteria of choice for this, and educators and administrators are after new ways to predict success and failure early enough to stage effective interventions, as well as to identify the effectiveness of different interventions.

1.2 The Goal

The goal for this project is implementing a student intervention system using concepts from Supervised Machine Learning. I am going to choose and develop a model that will predict the likelihood that a given student will pass, thus helping diagnose whether or not an intervention is necessary.

I will suppose that the data available to create the model is a representative but a small sample of the dataset that I would have access in the production environment. So, besides the typical accuracy requirements for any machine-learning project, I am also going to look for models that are efficient in the use of resources (computation time). Thus, the model will be evaluated on three factors:

[1] Source: `https://goo.gl/i8TQwZ`

- Its F_1[2] Score, summarizing the number of correct positives and correct negatives out of all possible cases. In other words, how well does the model differentiate likely passes from failures
- The size of the training set, preferring smaller training sets over larger ones. That is, how much data does the model need to make a reasonable prediction?
- The computation resources to make a reliable prediction. How much time and memory is required to correctly identify students that need intervention?

1.3 Classification vs Regression

The model that will be developed is a Classifier. According to [1], the distinction in the output type has led to a naming convention for prediction tasks: *regression* when we predict quantitative (continuous) outputs, and *classification* when we predict qualitative outputs (discrete). As the goal of this project is to identify if the student will succeed, the problem posed is a classification problem once it requires a binary answer (passed, failed).

2 Exploring the Data

In this section, we will explore the data to look for insides about the features.

2.1 Basic Facts

Let's go ahead and execute a basic description of the student dataset (Table 1):

	Value
Total number of students	42
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 %

Table 1: Facts About the Dataset.

All the 30 features in the dataset are discrete and Qualitative. There are 13 binary data, 13 ordered categorical variables and other 4 Categorical features. Among them, there are variables such as Father's Job, Family Size, if the student wants to take higher education or if he/she is in a romantic relationship. The data offers a pretty comprehensive profile of student life. At this point, it is hard to say what features are relevant or not. A complete description of each variable can be found on the Github project page[3].

Due to the dataset imbalance, as pointed out by the Code Reviewer, the model will be evaluated using F_1 instead of accuracy. Also, there is no missing data on this dataset and no presence of "real" outliers.

[2]Source: `https://en.wikipedia.org/wiki/F1_score`
[3]Source: `https://goo.gl/dkHKPV`

2.2 How the features are spread out

Before moving on, I am going to plot each feature to see how each inner classes of each variable are divided between the target labels "Passed" and "Not Passed". In the Figure 1 is plotted the Categorical features of the dataset. Here is possible to see that when the guardian is not the mother nor the father, it is more likely that the pupil not pass. Also, when the mother's occupatsion is teaching (*Fjob*), is likely that the student to pass.

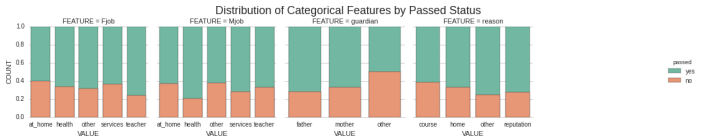

Figure 1: Categorical Data.

Looking at the Ordered Categorical Features (Figure 2), where there is an ordering of the values, the features *absences*, *age* and *failure* stand out. The *failures* feature is the number of past class failures (n if $1 <= n < 3$, else 4) and absences is the number of school absences. In all these variables, the bigger the number, the higher the likelihood of the student does not pass.

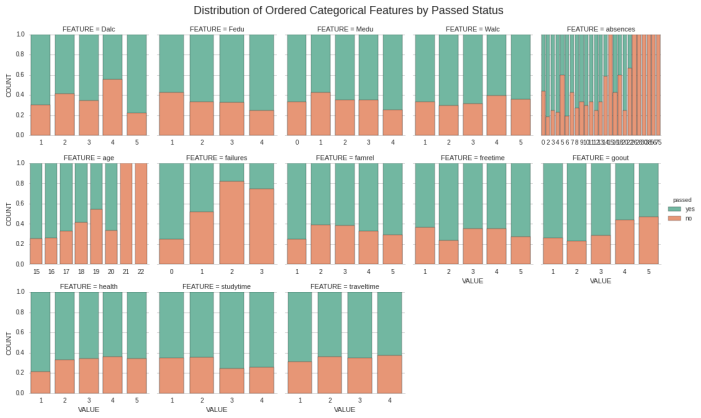

Figure 2: Ordered Data.

Finally, looking at the Binary Data (Figure 3), what more draw the attention are the features *higher* and *schoolsup*. The feature *higher* is if the student wants to take higher education and *schoolsup* is if the student has extra educational support. Some features, as *activities* and *nursery*, presented a very similar distribution between the classes

As can be seen, there are a lot of features that might add up to the classification process, although some features may be not so relevant It is important to notice that the model developed with this amount of data (365 data points) and variables (30) can suffer from the *curse of dimensionality*[4]. Many of the features will be converted into dummy variables, what could increase the amount of data needed to build an accurate model.

[4]Source: `https://en.wikipedia.org/wiki/Curse_of_dimensionality`

Figure 3: Binary Data.

3 Preparing the Data

In this section, I will prepare the data for modeling, training and testing.

3.1 Identify feature and target columns

It is often the case that the data you obtain contains non-numeric features. This can be a problem, as most machine learning algorithms expect numeric data to perform computations with.

```
Feature column(s):
==========================
```

school	sex	age	address
famsize	Pstatus	Medu	Fedu
Mjob	Fjob	reason	guardian
traveltime	studytime	failures	schoolsup
famsup	paid	activities	nursery
higher	internet	romantic	famrel
freetime	goout	Dalc	Walc
health	absences		

```
Target column:
==========================
passed

Feature values:
==========================
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	
0	GP	F	18	U	GT3	A	4	4	
1	GP	F	17	U	GT3	T	1	1	
2	GP	F	15	U	LE3	T	1	1	
3	GP	F	15	U	GT3	T	4	2	
4	GP	F	16	U	GT3	T	3	3	...

3.2 Preprocess feature columns

As can be seen above, there are several non-numeric columns in the dataset that need to be converted. The easiest case is when are only two categories, as "yes" or "no", e.g. *internet*. These are often represented by single binary digit as 0 or 1.

According to [1], when there are more than two categories, the most commonly alternative used is coding via *dummy variables*[5]. This method consists in representing a K-Level qualitative variable by a vector of K binary variables, where only one of which is "on" at a time.

So, for example, the *Fjob* feature will be split into 5 new features (*Fjob_teacher*, *Fjob_other*, *Fjob_services*, etc.), and will be assigned a 1 to one of them and 0 to all other columns.

This transformation will be performed using the pandas function 'get_dummies()'. Initially, all binary data that the classes do not correspond to *yes/no* will be converted into dummies.

```
Processed feature columns (48):
```

school_GP	school_MS	sex_F
sex_M	age	address_R
address_U	famsize_GT3	famsize_LE3
Pstatus_A	Pstatus_T	Medu
Fedu	Mjob_at_home	Mjob_health
Mjob_other	Mjob_services	Mjob_teacher
Fjob_at_home	Fjob_health	Fjob_other
Fjob_services	Fjob_teacher	reason_course
reason_home	reason_other	reason_reputation
guardian_father	guardian_mother	guardian_other
traveltime	studytime	failures
schoolsup	famsup	paid
activities	nursery	higher
internet	romantic	famrel
freetime	goout	Dalc
Walc	health	absences

3.3 Split data into training and test sets

So far, I have converted all *categorical* features into numeric values. To be able to judge if the model chosen in the next section will generalize well from its experience, I will hold out part of the data to measure how the algorithms are performing on yet-unseen examples. In this next step, I am going to split the data (both features and corresponding labels) into training and test sets. I will use the function *cross_validation.train_test_split()* from *scikit-learn* for that.

```
Training set: 296 samples
Test set: 99 samples
```

[5]Source: `https://www.moresteam.com/whitepapers/download/dummy-variables.pdf`

4 Training and Evaluating Models

As stated in Section 1, the model will be evaluated on three factors: F_1 Score, The size of the training set needed to build an acceptable model, and the computation resources used. Before selecting the algorithms to test their performance and amount of data needed, I am going to check the resources that different Supervised Learning Algorithms take by training them using different sizes of data sets. After that, I will select three algorithms to describe and analyze their performance under different training set sizes using their learning curves

4.1 Computation Resources Test

I am going to test computation resources needed for six different algorithms: $DecisionTree$, SVM, $K-NN$, $NaiveBayes$, $LogisticRegression$ and $AdaBoost$. I will use the scikit-Learn implementation of these algorithms using their default parameters.

To assess more realistic measurements of the resources used, I will create new datasets with the training and test datasets repeated 8 times. In the next subsection, where I will check the F_1 score of some of these models, I will perform the tests again using the original data.

In the Figure 4, I have measured the time that each algorithm took to fit the model and to perform the predictions. As expected, $K-NN$ took longer in the testing phase than in the training phase. As instead of fit a model to the data, it holds the examples to find the closest neighbors, this behavior was expected.

The SVM presented a curious behavior. The time increased sharply and dropped suddenly. As pointed out in the scikit documentation[6], the fit time complexity of the SVM is more than quadratic.

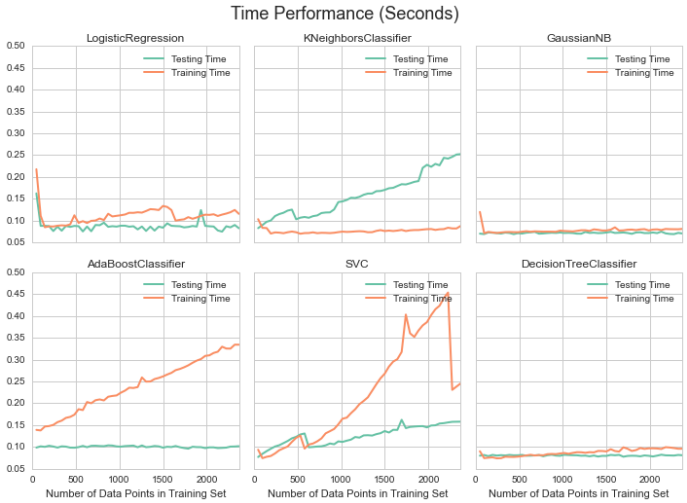

Figure 4: Time Performance of different algorithms

The time complexity of the *AdaBoost* Classifier depends on the number

[6]Source: `http://scikit-learn.org/stable/modules/svm.html#complexity`

of base learners[7] used (*DecisionTree* and the maximum 50, respectively). The time to fit the model has increased with more data, although the time to predict was kept at an acceptable level.

Considering the resources constraints, I will keep testing just the models that presented a relatively stable time both to test and to predict. In the next section, I will analyze the F_1 score of the *GaussianNaiveBayes*, *DecisionTree* and *LogisticRegression*.

4.2 Performance Test

In this subsection, I will analyze each model chosen in the last section, presenting a brief explanation about each one and examining its learning curves.

4.2.1 Naive Bayes

According to [2], one highly piratical Bayesian learning method is the naive Bayes learner. Its performance has been shown to be comparable to neural network and decision tree learning in some domains. The model assumes that each feature is conditionally independent given a target value:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

Where v_{NB} is the label (passed or not passed, in this case) that maximize this calculation. The model classifies any data point by multiplying the probability of a particular class and the likelihood of observing each feature given that class.

[1] says that, while the model assumption is generally not true, it does simplify the estimation drastically. it also indicates that the probability of a given event probability might be wrong, but the conclusion will be right.

In the Figure 5 is possible to see that the F_1 score has stabilized already around 100 data points used to fit the model, but its maximum score it achieved just around 296 data points. The training and test time are pretty close, as shown in the Figure 4.

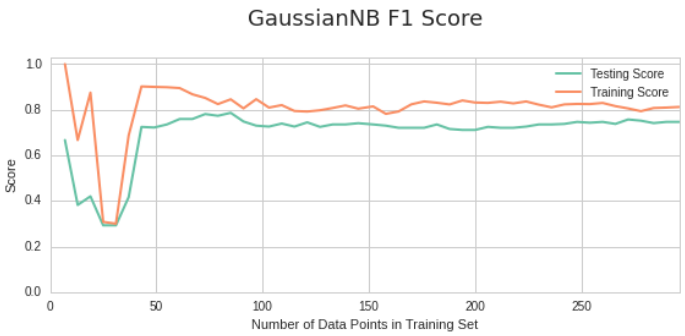

Figure 5: Naive Bayes Learning Curve

[7]Source: `http://goo.gl/QR1gB8`

	Training set size		
	103	200	296
prediction time (secs)	0.0889	0.0801	0.0890
training time (secs)	0.0708	0.0862	0.0803
F1 score on test set	0.7259	0.7111	0.7465
F1 score on training set	0.8092	0.8315	0.8120

Table 2: Nayve Bayes Measurements

The sklearn documentation[8] states that one of the common use of the algorithm is in classification and spam filtering. The model requires a small amount of training data to estimate the necessary parameters.

Some of the pros of this model are: inference is cheap; the model has few parameters; empirically successful. A negative side is that The assumption that the attributes be conditional independent of each other is too strong. For example, it can't learn interactions between labels[9].

Given the resource consumption of the Naive Bayes to train and predict and the F_1 Score achieved, this model seems a strong candidate for the final model.

4.2.2 Decision Tree

[2] explained that Decision tree is a method for approximating discrete-valued functions that are robust to noisy data and capable of learning disjunctive expressions. The model classifies instances by sorting them down the tree from the root to some leaf node (which is the label of the data). It has been applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

The *Scikit-learn* uses an optimized version of the CART (Classification and regression trees) algorithm, that basically constructs binary trees using the feature and threshold that yield the largest information gain at each node. So, as shown by [1], given that there are 2 classes in the dataset (or regions R_1 and R_2):

$$f(x) = \sum_{m=1}^{2} c_m I(x \in R_m)$$

Where:

- $\hat{c}_m = ave(y_i | x_i \in R_m)$
- $R_1(j, s) = \{X | X_j < s\}$ and $R_2(j, s) = \{X | X_j > s\}$

$I(.)$ is an indicator function[10] and $\hat{c}$ is what should be minimized by tweaking the combination of the splitting feature and the split point (j, s). The procedure to minimize $\hat{c}$ can be found in *Scikit-learn* documentation[11].

The figure 6 shows clearly that the model overfits the Training Data set, although it seems that didn't affect the F_1 score in the test set. Instead, it

[8]Source: `http://scikit-learn.org/stable/modules/naive_bayes.html`
[9]Source: `http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/`
[10]Source: `https://en.wikipedia.org/wiki/Indicator_function`
[11]Source: `http://scikit-learn.org/stable/modules/tree.html#tree`

continues to grow significantly when more data is added. The time to train and test is similar to the $GaussianNB$ results.

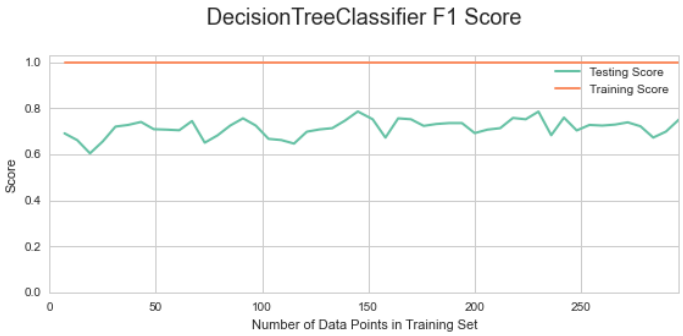

Figure 6: Decision Tree Learning Curve

	Training set size		
	103	200	296
prediction time (secs)	0.0888	0.0788	0.0924
training time (secs)	0.0767	0.0818	0.0842
F1 score on test set	0.6667	0.6917	0.7500
F1 score on training set	1.0000	1.0000	1.0000

Table 3: Decision Tree Measurements

Still according to the documentation, some advantages of decision trees are: Decision Trees are simple to understand and to interpret, as they can be visualized; Requires little data preparation (like normalization); Performs well even if its assumptions are somewhat violated by the true model; The cost of using the tree is logarithmic in the number of data points used in the training phase.

The disadvantages include: it can create over-complex trees that do not generalize the data well; and if some classes dominate,the tree can be biased. Many disadvantages of this algorithm is mitigated by using decision trees within an ensemble, like $AdaBoost$.

An advantage of the $DecisionTree$ algorithm over the $GuassianNB$ is that it could be visualized, what would make it easier to explain to someone without knowledge about Machine Learning. However, It took more data to fit a similar model regarding F_1 score when compared to the last method. Also, this particular dataset is imbalanced (there are 60% of passed and 30% of not passed), and it can be an issue. Using the model within an ensemble method, as suggest in the last paragraph, is not feasible given the time to train the model, as shown in the Figure 4.

4.2.3 Logistic Regression

Contrary to its name, logistic regression is a classification method that performs a regression in a logistic (or sigmoid)[12] function , as said in [3]. In [1] book is

[12]Source: `https://en.wikipedia.org/wiki/Generalised_logistic_function`

said that this model is widely used in biostatistical applications where binary responses occur quite frequently. For instance, patients survive or die, have heart disease or not and so on. This model is usually fit by maximum likelihood, using the conditional probability of G given X. The general log-likelihood that should be maximized to θ(that will result in the β parameters of the function) is:

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

Where $p_k(x_i; \theta) = Pr(G = k|X = x_i; \theta)$ and the probabilities sum up to 1. The default implementation of *Scikit-Learn* solves the maximization problem using the C++ library "*liblinear*" and a penalization function called L2. More detail can be found in the documentation[13].

Looking at the Figure 7, the F_1 score stabilize just after 200 data point in the training set, increasing a little bit when the data available grows to 296 points. The time both to training and to predict is less stable than the last models, but it still holds in a acceptable level.

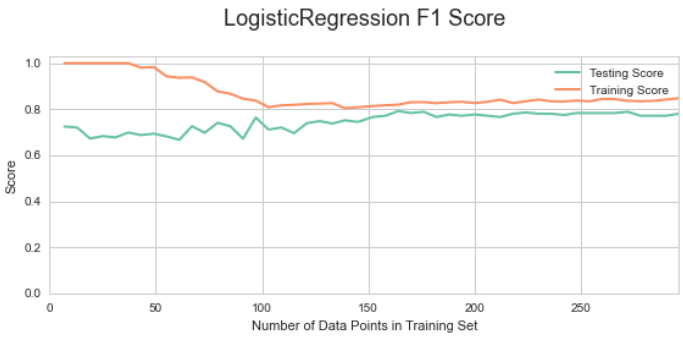

Figure 7: Logistic Regression Learning Curve

	Training set size		
	103	200	296
prediction time (secs)	0.0868	0.0922	0.0662
training time (secs)	0.0820	0.0889	0.0839
F1 score on test set	0.7111	0.7770	0.7801
F1 score on training set	0.8088	0.8269	0.8474

Table 4: Logistic Regression Measurements

Some advantages[14] of Logistic Regression are that the model is fast, is less sophisticated and easy to inspect. Some of the downsides[15] are are that, as any linear model, it is not robust to outliers, and can't handle imbalance data efficiently.

[13]Source: `http://goo.gl/T7h0Hz`
[14]Source: `https://goo.gl/ICWc4P`
[15]Source: `http://d-scholarship.pitt.edu/7034/1/realfinalplus_ETD2006.pdf`

5 Choosing the Best Model

In this section, I will choose the model that best filled the requirements of limited resources and performance based on the experiments I have performed earlier. Thus, I will describe the algorithm in Layman's terms and, finally, tune it using Grid Search.

5.1 Choosing the Optimal Model

The *LogisticRegression* model is the simplest of all algorithms tested and presented a remarkable performance, especially considering the imbalanced dataset used to train the model. Even if this model had presented a performance similar to the other algorithms, it should be considered as the best one due to the Occam's Razor: "Among competing hypotheses, the one with the fewest assumptions should be selected".

Considering the time performance, the results of the *LogisticRegression* was less stable than the other algorithms, when we look at the tables. For instance, the records show that it took 0.0992 seconds to predict with 200 data points in the training set, against 0.0788 of the $DecisionTree$ and 0.0801 of the $GaussianNB$. However, the prediction time of the *LogisticRegression* was the best one when used 296 data points. It dropped to 0.0662 seconds to predict, against 0.0924 from the $DecisionTree$ and 0.0890 from the Naive Bayes. Overall, the time performance of the algorithm remained at an acceptable level, as can see in the Figure 4.

Comparing *LogisticRegression* to $DecisionTree$, it has already achieved a F_1 score greater than 0.70 using just 100 data point to fit the model, while the $DecisionTree$ took around 200 data point to achieve that. Comparing to $GuassianNB$, using 200 data points the Naive Bayes presented a F_1 score of 0.7111. The *LogisticRegression*, 0.7770. Also, it seems that this amount of data is enough for the model, given that adding more 96 data points just increased the F_1 score just 0.003 units.

5.2 Describing the Model in Layman's Terms

As explained before, logistic regression is a classification method that performs a regression in a logistic function. It fits a model by calculating the maximum likelihood, using the conditional probability of G given X.

In layman terms, it means that the model estimates its predictions[16] using probability. The function approaches[17] 0 and 1, getting closer and closer to those values as the feature value grows, but never reaching them. So, the answer to the model always stays within the [0, 1] range.

To illustrate the way the model works, in the Figure 8 below, at the right, I plotted an artificial feature value on the X axis with the corresponding class range, either 0 or 1, on the Y axis. What logistic regression does is measure the probability that a given value of a feature to belong to the class 1. Once the model is fitted to the data, I could predict class 1 if $P(feature) > 0.5$, or class 0 otherwise. It is illustrated in the chart at the left.

[16]Source: `https://www.quora.com/What-is-logistic-regression`
[17]Source: `https://www.quora.com/What-is-logistic-regression/answer/Alaka-Halder`

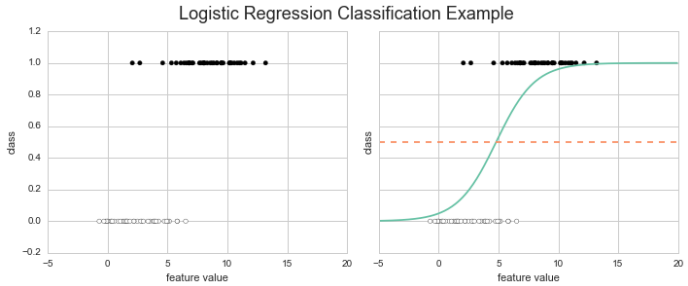

Figure 8: Ordered Data.

So, for example, using the model above, a feature value of -1 would be assigned a probability of 4% of the observation is part of class 1. As it is below the 0.5 line (the dotted line), the model would assign class 0 to this instance. If the feature value were 7, the probability would be 89%, and the model would say that the observation is part of class 1.

In the context of this project, 0 means that the model is entirely sure that the student will fail to pass, and 1 means that it is absolutely certain that student will pass. Any value above 0.5 means that the model is pretty sure that the students will succeed, so it classifies as "passed."

5.3 Model Tuning

In this section, I performed a Grid Search through a parameter space looking for the set of parameters that yields the best result. This parameter space is all possible combinations of the following *LogisticRegression* parameters:

- C: $[0.01, 0.1, 0.5, 1., 2., 10., 100.]$
- *solver*: ['newton-cg', 'lbfgs', 'liblinear']

Where C is the Inverse of regularization strength and *solver* is the algorithm used in the optimization problem. To determine the "best result," Grid Search use a 3-fold Cross-validation procedure. Basically, it separates a sample into three subsets and uses one partition to train the model and another to verify its performance. Then, the procedure is run multiple times using different subsets to train and validate the model. The results are averaged over the rounds.

Calculating the F1 score using the classifier already tuned I got:

	Value
Prediction time (secs)	0.080
Peak memory (MiB)	97.92
F1 score for training set	0.826
F1 score for test set	0.787

Table 5: Final Model

As was expected, after tuning the model, it performed just slightly better than the default parameters, achieving a F_1 score in the test set of 0.787 against 0.7801 obtained before. The prediction time got a little worse than before (0.0662).

6 Reflection

Although I have found an appropriate model to the task at hand, I believe that I should have done some feature engineering or dimensionality reduction. I believe that the most of the models tested would benefit from a dataset with less dimensions. It would require the implementation of some new steps, including repeat what was done so far some more times. It may seem odd perform the whole process again and again, but in the data science work-flow, it is expected.. After all, as I have been learning at Udacity courses, Machine Learning can be very iterative.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009.

[2] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[3] W. Richert. *Building Machine Learning Systems with Python*. Packt Publishing, 2013.