B(E)3M33UI — A brief introduction to Machine Learning

Jiří Spilka, Petr Pošík

February 19, 2019

1 Introduction

The goal of this exercise is to create a simple machine learning model for a small dataset. We use a simple data set to introduce ML concepts (learning and testing) and build a machine learning model using the scikit-learn package.

2 The project

Suppose that we were contacted by a university administration to solve the inefficient and time-consuming exams. The current situation of a written exam and oral exam is high time demanding and stressful for both teachers and students. Thus, the goal is to replace or redo exams so that it is efficient, but it also guarantees that students have learned the required topics. In the context of AI and ML in particular, the specific task is to learn a model using the past exams and student data and to provide a prediction whether a student might pass an exam before actually taking it [1].

3 Dataset

A dataset of AI course results and information provided by students is available for the years of 2015-2018, cf. file: `data_ai_course_2015_2018.csv`. The dataset contains the following main attributes:

- *student_name* - str, anonymized
- *year_course* - int, year when the course was taken (2015-2018)
- # *lecture attendance* - int, number of attended lectures
- # *seminar attendance* - int, number of attended seminars
- # *points_st1* (int, number of points from the first semester task),
- # *points_st2* (int, number of points from the second semester task),
- # *points_seminars* (int, number of points in total from seminars),
- # *hours_per_week* (int, approximate number of hours per week spent by study)
- # *hours_per_week_ai* (int, approximate number of hours per week spent for AI course)
- # *hours_for_exam* (int, approximate number of hours spent for AI course exam)
- # *points_exam* (int, number of points from exam)
- # *points_total* (int, total number of points from seminar and exam)
- # *grade* (str, final grade, A, B, C, D, E, F)

The attributes above are called features and compose feature matrix X where rows represent individual students and columns attributes (features). The target variable is typically y. We make our situation a bit easier and create a *binary* classification problem such that $y = 0$ when a grade is one of D, E, F and $y = 1$ when a grade is one of A, B, C.

[1] At first glance, it might be weird to replace the exam and human decision by machine learning model, but this is already a reality in many different fields, e.g. decision about credit risk when a banker needs to decide whether a loan for a client might default (or what should be the interest rate for the client).

Also, for the ease of exposition, we select only *points_st*1 and *points_st*2 as the features. We will use the other features during the semester.

4 Methods

Given the feature matrix X and the target variable y we would like to find a function $f : X \rightarrow y$ such that classification error, ϵ, for the current semester is minimal. The problem is that we do not want to wait till the end of the semester to determine ϵ. Thus we use the past data from 2015-2017 for model search and data from 2018 for estimating ϵ.

4.1 Machine learning from scratch

In this case, we want to implement everything from scratch. The advantages and disadvantages are clear: we would learn a lot by implementing data manipulations, algorithms, performance evaluation but it would take us a long time before we could get some results.

4.2 Python scikit-learn library

The scikit-learn is the most used python machine learning library (as of Feb 2019) and contains everything one would need for solving a typical machine learning task. It has a rich API and includes tools like preprocessing (outlier detection, normalization), dimensionality reduction, classification, regression, model selection and evaluation, etc.

The package is reusable so it can be used in various context and different project. It allows us to quickly stitch together different components that we need for a specific task.

The code for this exercise is in: `ml_intro_scikit_learn.py` - the main script, `utils.py` - data manipulation.

4.3 Auto ML (auto-sklearn)

Using the automatic machine learning (Auto ML) we are primarily interested in results only. We do not care what algorithm is used and how it is used as far as performance on test data is satisfactory. It typically takes a much longer time to run. Also, debugging and interpretation is more tricky, but it can save a time when we are exploring set of models that might work or we are solving the same task for hundred and first time just with a different data set. The code for this exercise is in `ml_intro_auto_scikit_learn.py`.
The output of the auto-sklearn run can be observed in folder: /tmp/auto_sklearn_*. Note that the auto-sklearn can not be used under the Windows platform but we will not need this package anyway.

5 Conclusion

This simple exercise showed how to apply a machine learning model to a real-life scenario. We have shown that we can predict whether a student will pass an exam with grade *C* or better with an accuracy above 70%. We have demonstrated two different ways how to do it: first using scikit-learn and second using Auto ML.

In this course, we will use parts of scikit package as it is, but we will also implement critical parts from scratch to expose the problems in detail We certainly do not want to use Auto-ML approach, we need to understand machine learning methodology and algorithms to be able to improve performance and debug our solutions.