Practical 1: Credit Approval

CS5014 Machine Learning

Due date: Fri 5th March (Week 6) 21:00
40% of the coursework grade

Aims

The aim of this practical is to gain experience in applying machine learning methodology to a real dataset. The focus is on good understanding and justification of steps. A successful submission will demonstrate the understanding of:

- how to load, clean, and process a dataset;
- how to train a standard algorithm;
- how to report and interpret the results; and
- how to write clear, concise, and re-usable research code.

Dataset

The dataset for this practical is provided on studres in the directory named `data`. It contains two files: `crx.names` contains the description of the different features in the dataset, while `crx.data` contains the actual data, in the CSV format. It contains anonymised data pertaining to individual credit card applications, as well as the final outcome (successful or unsuccessful). Read `crx.names` to familiarise yourself with the values. You will notice that some of the values are numerical (also referred to as 'continuous'), and some are categorical. With categorical inputs, the description will list all valid labels. The target variable (outcome of the application) uses '+' for successful and '-' for unsuccessful applications.

Task

You will create a machine learning model that can predict credit card approvals from the described dataset. You will create two deliverables: the source code for your solution, and a brief report which answers specific questions about your solution. You should follow the steps outlined in this spec and use the questions to guide your progress. Both code and the report are important: you must specifically answer each question listed below and will be evaluated on how well your answers demonstrate understanding of the topics covered in lectures.

In this practical, you will be marked based on your use of logistic regression. There are no extra points for using more advanced algorithms. If you experiment with other classifiers, please separate this work into new Python files and clearly identify them to ease marking.

Part 1: Data Processing

Start by loading the dataset using Pandas. You may want to drop or clean some of the values, change the encoding, apply scaling. You will also need to separate the dataset into a training and testing set. In your report you should clearly answer the following questions about your data processing:

(a) How did you load and clean the data, and why?

(b) How did you split the data into test/train set and why?

(c) How did you process the data including encoding, conversion, and scaling, and why?

(d) How did you ensure that there is no data leakage?

Part 2: Training

After loading the data, you should train a logistic regression classifier to predict the output from inputs. You should use the LogisticRegression algorithm for this. Make sure you are familiar with all the parameters offered by this implementation, and what they mean. In this part, make sure to set the `penalty` parameter to `'none'` to get the basic, unregularised version of logistic regression. In your report you should clearly explain how you performed each of the following tasks during the training process:

(a) Train using `penalty='none'` and `class_weight=None`. What is the best and worst classification accuracy you can expect and why?

(b) Explain each of the following parameters used by LogisticRegression in your own words: `penalty`, `tol`, `max_iter`.

(c) Train using balanced class weights (setting `class_weight='balanced'`). What does this do and why is it useful?

(d) LogisticRegression provides three functions to obtain classification results. Explain the relationship between the vectors returned by `predict()`, `decision_function()`, and `predict_proba()`.

Part 3: Evaluation

After successfully training your model on the training data, you should evaluate your model on the testing data. It is fine to use built-in sklearn functionality like `accuracy_score`, but you will have to understand what such functions do. In your report, you must clearly explain the following:

(a) What is the classification accuracy of your model and how is it calculated? Give the formula.

(b) What is the balanced accuracy of your model and how is it calculated? Give the formula.

(c) Show the confusion matrix for your classifier for both unbalanced (2a) and balanced (2b) cases. Discuss any differences.

(d) Plot the precision-recall curve and report the Average Precision (AP) for your algorithm. What is th erelationship between AP and the PR curve?

Part 4: Advanced Tasks

Once you have successfully completed Parts 1-3, you can try some advanced tasks listed below. These are required for 17 and higher, but they cannot make up for poor performance in previous tasks.

(a) Set the `penalty` parameter in LogisticRegression to `'l2'`. Give the equation of the cost function used by LogisticRegression as the result. Derive the gradient of this l2-regularised cost.

(b) Implement a 2nd degree polynomial expansion on the dataset. Explain how many dimensions this produces and why.

(c) Compare the results of regularised and unregularised classifiers on the expanded data and explain any differences.

(d) This question requires you to read some relevant research paper and combine with what you have learnt to form an answer. Note that the available solvers to train the model include: "newton-cg", "lbfgs", and "sag". "newton-cg" stands for Newton Conjugate Gradient, "lbfgs" stands for limited memory Broyden Fletcher Goldfarb Shanno (BFGS) [1], and "sag" stands for Stochastic Average Gradient [2]. Read the relevant references and answer: which of the three methods are first order optimisation and which are second order optimisation methods? What are the differences between these two methods (first order and second order optimisation)? Why does the package suggests that "sag" is faster for large datasets?

Code Quality

Your code will evolve as you tackle individual parts of this practical. At the end, you will have code that produced all your results. This is research code so you should focus on the code quality aspects that support research. Your code should be:

(a) correct,

(b) clean and understandable,

(c) concise and elegant, and

(d) repeatable and easy to modify.

You will not need to write a lot of code and should avoid overcomplicating. Focus on how easy it is for someone else to take your code, understand it, reproduce your results, and make modifications to support further experiments. Our marking will be based on how well your code meets these criteria. We encourage you to keep these factors in mind from the beginning, but it is also OK to focus on correctness first and clean up the code later.

Submission

Hand in via MMS, by the deadline of 9pm on Friday of Week 6:

- The source code of your application which works in the Python3 virtual environment set up in the school labs. This **must** be in the form of human-readable .py files, **not** the binary .ipynb notebook format!

- A brief report in the PDF format. The report must contain sections which correspond to the four parts described in this specification and it must address each of the questions associated with each part.

Create a single `.zip` file containing all of these and submit this to MMS. Do *not* include the dataset, your python virtual environment, or git repository.

Marking and Extensions

This practical will be marked according to the guidelines at
https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html
It will be based on the quality of your answers to the questions and the quality of your code. The report is the most important part of the submission – your answers should be brief, but they have to demonstrate understanding of the underlying algorithms. The code will be evaluated based on the criteria listed in Part 5.

Some examples of submissions in various bands are:

- A *basic implementation* **in the 11–13 grade band** will complete Parts 1-3, but with significant weaknesses. Examples include a messy implementation, unexplained differences between the code and the description in the report, or incorrect or incomplete answers to questions in Parts 1-3.
- An implementation **in the 14–16 range** should complete all parts of the basic specification comprising Parts 1-3, including answers to all associated questions. The code should be of good quality, and the answers should be mostly correct and insightful, and demonstrate understanding of lecture materials.
- An implementation **in the 17–18 range**, must include a high-quality solution to Parts 1-3, and some work on Part 4. Excellent answers to all attempted questions are strictly required for this grade band.
- A grade of **19 and higher** requires an excellent solution to all four parts with exceptionally clear code and insightful answers to questions which evidence deep understanding and independent study.

Note that the goal is *solid machine learning methodology and understanding* rather than a collection of extensions – a good scientific approach and analysis are difficult, whereas running many different scikit-learn algorithms on the same data is easy. Also note that:

- We will not focus on software engineering practice and advanced Python techniques when marking, but your code should be sensibly organised, commented, and easy to follow, as described above.
- Standard lateness penalties apply as outlined in the student handbook at https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html
- You must reference any external sources used. Guidelines for good academic practice are outlined in the student handbook at https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html

References

[1] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995. URL: http://users.iems.northwestern.edu/~nocedal/lbfgsb.html.

[2] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. URL: https://hal.inria.fr/hal-00860051/document.