Elements of Machine Learning – Practical Assignment

Fall 2023, Instructors: dr. Giacomo Spigler, Fred Atilla

Introduction

For this assignment you will perform a classification task on the dataset provided along with this description. You will need to submit your code, as well as a report describing your decisions and results.
The assignment is worth 30% of your final grade. The grade will be based on the quality of your work as judged by the instructor from your report and code. A 5.5 is required to pass this assignment. There will be a retake if you fail, but then your maximum assignment grade is limited to a 6.

Task and Dataset

In this assignment you will perform multi-class classification on a dataset of "diamonds" to identify their type. The dataset is available for download on the same Canvas page as this assignment. The dataset consist of 5645 diamonds of 4 types with each diamond having 12 features.

The labels are numeric and correspond to the diamond types shown in the image above. All features are numeric and represent various information measured from the diamonds such as their carat (physical weight), hardness, clarity, cut (it's shape), transparency, size (in millimeters), etc. It is not important to understand each feature for this task, because you are not a jeweler but a data scientist. Your goal is to outperform an expert jeweler, who achieves 70% accuracy, with the help of machine learning.

The file contains 2 numpy arrays, 1 being the features, and the other being the labels. You can load the file by using the following line of code if you put the file in the same folder as your code file (e.g., notebook):

```
data = np.load('data.npz')
X = data['features']
y = data['labels']
```

If this line of code does not work, you will have to load the file using it's absolute path. You can find out how to do that on Windows and Mac, but you can also ask your classmates or instructors.

The task is to train two models to classify the diamond type, and compare model performances. You can choose any two models for comparison (K-Nearest Neighbor, Decision Trees, Logistic Regression, Neural Networks, etc.). You will have to preprocess your data using a couple of the methods we learned in the practicals. It is up to you to find out which steps are necessary and which are optional, but might help improve your model performance.

Remember to use:
- training data to train your models
- validation data to find the optimal hyperparameters of your models and choose the best model
- test part to evaluate the final tuned and chosen model

Method

There are four important restrictions on the method used:

1. The method should be fully automatic, that is, re-running your code should re-create your final accuracy. To achieve this, make sure you set the random state argument wherever possible to a fixed number, so that I don't get different results from those you report.
2. The method should not be hard-coded in any way. That is, if we provide a separate set of test data as input to your code (all the steps from loading it to testing the model on it), it should run and predict the labels without any changes in the code, and achieve a similar final accuracy as you got on you test data.
3. Every software component used should be open-source and possible to install locally. This means that you cannot use proprietary closed-source software or access to a web service to carry out any data processing.
4. The method should not use any external dataset. You can only work with the dataset you are given.

Report

You need to write a report accompanying your code. Your report should be 3 pages maximum (no minimum) in **PDF format** and should include the following:

- Your name and student number
- Description of your code and results (see Criteria & Grading below).
- References or appendix (does not count towards the 3 page maximum)

There is no required format e.g. font or font size. You can have whatever format you want as long as it is eligible. Note 3 pages are more than enough, so it really is not necessary to make text super small just to fit everything. Figures can be put in the appendix if you need more space to write.

Criteria & Grading

The grading detailed below is based on the report, but your code should be aligned with the report (e.g. when we review your code, we should be able to see the parts for HP tuning, training, etc.). Code without report or report without code will not be evaluated.

22 points total for:

- For each ML model, you can receive 8 points for:
    - Model choice (2 points): choose a model that works for this task, implement it correctly in the code, and briefly explain in your report what the model does and why it was chosen for this assignment.

      Note: You can NOT compare a Neural Network with N hidden layers and a Neural Network with Z hidden layers as two different models. In that case, your work will be evaluated for only using one model. You can however compare a Recurrent NN with a Feedforward NN (they are two different ML models). Same for comparing MLP from sklearn to a Keras neural network.

    - Input to classifier (2 points): How do you represent your data? Have you applied any preprocessing such as changing or transforming the data? If yes, explain what they are and why you performed them. Note it is up to you to perform the same or different steps for the 2 models.
    - Hyperparameter tuning (2 points): Which hyperparameters are tuned and how? For each hyperparameter you tuned, what were the candidate values you tried and why these values? Discuss the impact of the different values on the performance (choose at least 2 hyperparameters with each at least 2 values to try).

  - Model training (1 point): Give any additional information about training (e.g. compiling, regularization, cross-validation, etc.) if there is any. Report the best settings you found for the model.
  - Model evaluation (1 point): Report the performance of the tuned model (with the best hyperparameters).
- Comparison of ML model 1 and ML model 2 (2 points)
  Compare the performances of both models (from the Results section above). Which model performed better? And which model was faster to train? (Exact times can be computed or retrieved from `cv_results_`).
- Final test accuracy (2 points): Report the final generalization performance. 1 point for performing at/above chance level (25%), and 1 more point if you perform at/above 70% (beating the expert jeweler).
- Final results (2 point): Provide a classification report and/or confusion matrix as a figure or table for your final model. Comment on which diamond type was the easiest and which the most difficult to classify. Did any types get confused for each other?

Submission

You can submit your assignment via Canvas until **Sunday 3 December 23:59**. Your submission only needs to contain the following 2 files:

• Report (.pdf)

• Code: can be either a plain python script (.py) or a Jupyter Notebook (.ipynb), depending on which software you program in. It should be possible to run all your code at once to come to a similar final accuracy you achieved. This means you should comment out any unnecessary lines of code you experimented with. Lines that plot or print information are fine as those would not interfere. Make sure to test whether your code can run all at once by simply running all the code (most software has this option, including Jupyter Notebook).

Code reuse rules

Remember this is an individual assignment. You are not allowed to use or share code with other students. You are however allowed to ask each other general questions to help. Submissions will be checked for plagiarism. You are also allowed to use:

• code from the practical notebooks

• open source libraries available for Python, e.g. numpy, keras, etc.

• open source code from scientific papers, Github, Stackoverflow or similar websites, as long as it is credited in your report or code (as a comment) with a link to the source.