Exercise 1 - Language Identification with sklearn

From Linear to Deep :-)

Deadlines

Deadline for Exercise 1 is **16.10.2022, 23:59 (Zurich Time)**.
Deadline for the peer review is **24.11.2022, 23:59 (Zurich Time).** You will find instructions for the peer review process at the end of this document.
Deadline for feedback to your peer reviewers is **28.10.2022, 23:59 (Zurich Time).**

Learning goals

This exercise consists of two parts: the first part aims at deepening your understanding of linear models. The second part will already target a simple kind of multi-layered network; the multilayer perceptron (MLP). Don't worry if you don't know anything about MLPs when you read this. We will cover all you need to know to do the second part of this exercise next week in class and in the tutorial session. By completing this exercise, you should …

- … understand linear models and use them for multiclass classification tasks.
- … be able to implement different machine learning models, including MLPs, in scikit-learn.
- … understand the role of hyper-parameters, regularisation, and the problems of class imbalance.
- … perform an error analysis of machine learning models.
- … observe the most important features that lead to a prediction of a specific class and explains to some degree what the model does (XAI).

Please keep in mind that you can always consult and use the exercise forum if you get stuck (note that we have a separate forum for the exercises).

Deliverables

We request for you to hand in your solutions as self-contained Jupyter Notebook ipynb files. That way, your reviewers can view and execute your code without potential dependency problems and/or installing new packages or versions of packages.

Please hand in your solutions for part 1 and part 2 in separate notebooks. Name your notebooks exactly in the following fashion:

- ex01_lr.ipynb
- ex01_mlp.ipynb

The notebooks contain your well-documented, EXECUTED, and EXECUTABLE code. You will also have to write a lab report. The lab report should contain a detailed description of the approaches you have used to solve this exercise. Please also include results. We highlight places in this description with green where we expect a statement about an issue in your lab report.

We encourage you to solve the exercise on Google Colab and download the ipynb file when everything is done. .
Besides submitting the required ipynb files, you should also submit a lab report in PDF format.
Please submit a zip folder containing the following 3 files, named in exactly the following way:

- ex01_lr.ipynb
- ex01_mlp.ipynb
- ex01_labreport.pdf

We assume that the data is in the same folder as the scripts, e.g.

- ex01_ml4nlp1.zip
    - ex01_lr.ipynb
    - ex01_mlp.ipynb
    - Uniformly_sampled.tsv

Please note:

- The cells must have already been run and the output must be visible to anyone checking your notebook without having to run the code again.
- Your competitors must be able to run your code. If it doesn't work, you can't get the maximum score.
- DO NOT submit the data files!

Data

For both parts of this exercise, you will work with the same data. The goal is to classify the language of Tweets. This is a challenging extension of the problem described in Goldberg, chapter 2. However, we will work with more languages than just six and the text segments we need to classify are much shorter. The material folder in the exercise section of OLAT contains the two files "train_dev_set.tsv" and "test_set.tsv". The files are also published under these two links:

- train_dev_set.tsv: https://docs.google.com/spreadsheets/d/e/2PACX-1vTOZ2rC82rhNsJduoyKYTsVeH6ukd7Bpxvxn_afOibn3R-eadZGXu82eCU9IRpl4CK_gefEGsYrA_oM/pub?gid=1863430984&single=true&output=tsv
- test_set.tsv: https://docs.google.com/spreadsheets/d/e/2PACX-1vT-KNR9nuYatLkSbzSRgpz6Ku1n4TN4w6kKmFLkA6QJHTfQzmX0puBsLF7PAAQJQAxUpgruDd_RRgK7/pub?gid=417546901&single=true&output=tsv

To make the start a little easier, you can go to this notebook in Google Colab which loads the two files using the public links. If you like, you can just continue the exercise in your own copy of that notebook. If you choose to work locally, download the two files on your computer.

Inspect the data and see how it is distributed.

- What are the properties of the data? Describe the main distribution properties of this data set.

Use a 90/10 split for your train and development set[1]. Of course, it is forbidden to peek into the test data ;-). You should only run on the test set this when you evaluate your model that performs best on the dev set.

Part 1 - Language identification with linear classification

Scikit-learn is a useful Python library for all kinds of machine learning tasks. In the following, you will train several models in sklearn to solve this task. The aim is to become acquainted with a few different classifiers, as well as with the basic functionality of sklearn.

1. Create a suitable pipeline in sklearn to preprocess the data. Think about extending the feature space. What other features could you use to determine the language? Please include extra features linguistic features to your machine learning model for this task.
2. Train the following classifier: LogisticRegression
3. To find the optimal hyperparameter settings for the classifier, use sklearn's GridSearchCV. You are supposed to experiment with the following hyperparameters[2]:
    a. Penalty (Regularisation)
    b. Solver
    c. Experiment with parameters of the Vectoriser (not required, highly advised)

Report the hyperparameter combination for your best-performing model on the test set.
What is the advantage of grid search cross-validation? Use a confusion matrix to do your error analysis and summarize your answers in your report.

Now that you have your best model, it's time to dive deep into understanding how the model makes predictions.
It is important that we can explain and visualize our models to improve task performance. Explainable models help characterize model fairness, transparency, and outcomes. Let's try to understand what our best performing logistic regression classification model has learned. Generate a feature importance table for the top ten features (please have the features named) for the languages ['en, 'es', 'ja']. What is more important, extra features or the outputs of the vectorizer, discuss.

We recommend using the ELI5 library as it supports sklearn pipelines to explain the model weights. For more details, see their documentation on dealing with text classification. We will accept answers from any explanation library/method as long as the explanations for the model weights are provided in a structured/clear way.

---

[1] Generally use the 90/10 split. When using MLP with early stopping, sklearn will create its own internal validation split. In these cases, you are free to decide if you supply just the 90% training data or also the additional validation data.

[2] In general, we expect you to engineer the full GridSearch optimisation Pipeline. However, if runtimes take long, we recommend taking shortcuts on the optimisation methods (only 1 option per hyperparameter, max_iter being a ridiculously low number). Performance is not critical but being able to build the complete system is, which is what we want to see in the code and in the report.

Part 2 - Your first Multilayer Perceptron (MLP)

Let's see if you can beat the best linear model you've trained with sklearn with a non-linear MLP.

1. Train an MLP classifier. You can also use GridSearchCV, but be aware that training an MLP model takes much more time. Play around with 5 different sets of hyperparameters including layer sizes, activation functions, solvers, early stopping, vectorizer parameters, and report your best hyperparameter combination Do you achieve higher performance? Why/why not. Important: use the same data splits as for Part 1.

If you need help on that, Raschka's [2015] chapter 2 provides an introduction to MLPs. The Google Machine Learning Crash Course also offers good material.

Peer Review Instructions

If you are not already registered on Eduflow follow this link https://app.eduflow.com/join/GXHN93 and register with the E-mail address you use for OLAT. Then you should be added to the course page automatically.
As soon as the deadline for handing in the exercise expires you will have time to review the submissions of your peers. You need to do **2 reviews** to get the maximum number of points for this exercise.
Here are some additional rules:

- If you do not submit 2 reviews, the maximum number of points you can achieve is 0.75 (from a total of 1).
- Please use full sentences when giving feedback.
- Be critical, helpful, and fair!
- **All reviews are anonymous: Do not put your name into the python scripts, the lab report or the file names.**
- You must also give your reviewers some feedback. The same criteria as above apply.
- If you consistently provide very helpful feedback, you can be awarded a bonus of 0.5 in total in case you didn't achieve the full 6 points from all exercises. A maximum of 6 points from the exercises can go into the final grade.

Groups:

- You can create groups of two to solve the exercise together.
- Both students should submit the solutions separately.
- If you did not already work together for the previous exercise, write a small post in the "Groups"-thread in the exercise forum on OLAT to notify the instructors about the group.
- As a group member, you still have to review two submissions with your own eduflow account. However, you may work together in the group to write all 4 reviews.