Learning from Data – Assignment 1

General remarks

The aims of this assignment are the following:

- to get you acquainted with running basic machine learning experiments with scikit-learn
- to experiment with different feature sets
- to start reflecting on how certain learning algorithms work (Naive Bayes, Decision Trees, Random Forests, KNN, SVM)
- to start getting used to writing up experimental results in a research-oriented way
- to practice writing clean, commented and easy to use code

Practicalities:

- assignment files are on Brightspace
- you should complete this assignment in **groups** of 3 people - only 1 person submits
- what to hand it (please, upload on Brightspace):
  - modified code (`lfd_assignment1.py`)
  - a report written as a short research paper including experimental details, comments, and answers to questions (`pdf`). For the report, you should use the template that we have prepared, and that you can find on Brightspace. The code (and comments) are 20% of your grade, the full report the other 80%.
- deadline: **16th of September, 09:59 AM**.

Data

You are given two files for this assignment:

- `reviews.txt`: a corpus of reviews. Each review is on one line, and is headed by two meaningful tags:
  - a tag that specifies one of six topics: `books, camera, dvd, health, music, software`

  - a tag which indicates the sentiment expressed by the review, in terms of a positive or negative value: `pos, neg`.

  A third column contains the id, and the rest is the review's text. The text has already been tokenised. You have to split the data in train/dev/test yourself!

- `lfd_assignment1.py`: a script to run a Naive Bayes classification on this data, using the scikit-learn libraries. Of course, not all functionality you might need is already in this script. You'll likely have to use Google and the scikit-learn documentation to find what you need.

Setting up

It's a good idea to set up a virtual environment for this class. I suggest doing this using Conda. This example code (for Ubuntu) should take care of the set up for this assignment.

```
conda create -n lfd python=3.10
conda activate lfd
pip install scikit-learn
```

Now check if you can run the script succesfully. Also run it once with the -h argument, to see what your options for command line arguments are. Try to understand what they mean and what you can do with them. You have to add your own arguments throughout the assignment.

```
python lfd_assignment1.py
python lfd_assignment1.py -h
```

1.1 – Multi-class Classification

As you can see from the data, each review is tagged with a sentiment label (binary) and with a topic label (multi-class). It is good to write the code such that it can handle both types of classification. However, for this assignment you **only** have to look at the topic label, and thus perform multi-class classification. Please only report on this task in the report.

1.2 – Measures

The script as it stands only outputs the general accuracy of the system. You will also have to produce *precision*, *recall*, and *f-score*. Remember that these have to be calculated (and printed) **per class**. You should also output a confusion matrix. In the report, please include a few comments on the results. For example: is performance on one class better than performance on the other(s)? Can you speculate on why this is the case and potentially what could be done to change things?

1.3 – Algorithms

The script currently only implements Naive Bayes, whose properties we have seen in class. In week 1, we also have seen Decision Trees, Random Forest, KNN and SVM algorithms. Find the relevant documentation from sklearn and implement them in your script. Code wise it's

nice to make sure you can call each algorithm via a command line argument. Hyperparameters you experimented with should preferably be a command line argument (with sensible defaults).

To give you a head start: you probably want to check out DecisionTreeClassifier, RandomForestClassifier and KNeighborsClassifier. In scikit learn there are also two SVM implementations: `svm.SVC` and `svm.LinearSVC`. By checking the documentation online, and by testing both implementations for your experiments, please write up a short paragraph in your report where you explain what the differences between the two implementations are, and any usage recommendation you might have (when to use one implementation or the other).

In the report, we at least expect baseline scores for the NB, DT, RF, KNN and SVM algorithms. For each of them, you should experiment with the relevant hyper-parameters and report the best settings.

1.4 – Features

As discussed in class, there are many different possibilities for the feature sets, outside of just using the words. You are expected to at least experiment with some different feature sets. You have full freedom to come up with your own features, but here are some ideas:

- Look at the possible arguments of CountVectorizer and TfidfVectorizer and experiment with them
- Combine different vectorizers
- Experiment with n-gram sizes
- Filtering features based on relevance or frequency
- Lemmatizing or stemming the input words (NLTK, SpaCy)
- Using a tagger (Named Entity, POS-tags) to add as input features
- Using character-level features
- Adding the label of the other task as a feature (i.e. add the sentiment for the topic classification task)

You don't have to provide scores for all algorithms for all your feature sets. Sometimes less is more! You have to think about what scores are actually *interesting* to report. Generally speaking, it is fine to only take the best performing baseline algorithm from the previous section for your feature experiments.

We actually have a **held out test set** on which we will run your best model. The best performing groups on this set get a bonus on the assignment! Make sure to let us know how exactly we can run your best model (i.e. what command line arguments to use).

There are a few things that are not necessarily *expected* of you, but are a nice addition to the report:

- Some of the algorithms allow outputting a probability of a class. Find some nice examples in which the algorithm was wrong by a large margin. Was this to be expected? Can you spot the features that lead to this? Make sure to show the examples themselves in a nice way in the report.
- You can also output the *best features* of the vectorizers you used. You can print, for example, the top 10 (perhaps per class) and show them in your report. Reflect on what this tells you: was this to be expected? Does this point to overfitting?

1.5 – Code

As stated before, you are expected to hand in your code. In this case, you can probably fit everything in a single script. Make running the code as self-explanatory as possible. Remember that we will run your code on the held out test set, so it should be straightforward for us to run your best model on it! You should either include this information in the report, or add it as a comment on Brightspace.

Some guidelines for writing the code:

- Use functions! Have your code be as modular as possible, making subparts easy to re-use in future projects
- Add a lot of comments! You will be surprised how fast you forget what your own code does
- Each function should have a detailed docstring description
- Add command line arguments for all the functionality (which algorithm, which features) and relevant hyperparameter settings. We should be able to reproduce your results without having to change things in the actual code
- Include the code for all functionality in the submission
- Clean up old code and functions
- Look for existing sklearn implementations for certain tasks (e.g. cross validation, data splitting, grid search, confusion matrix). No need to reinvent the wheel

Also note that some things are **not** required. I don't expect functions to have typing hints and explicit input/output types in the docstring. Of course, that might still be good practice, it is just not required in this course.

1.6 – Writing the report

You are asked to write a report where you explain what experiments you ran (and why), what the results are and if this was to be expected. You should also describe all experiments you have run for the exercises above, and include any comments and/or answers that you were asked to provide above. You should also briefly describe the properties of the five algorithms you have used. Also, remember the advice from the slides of Week 2!

Important — Please note that we have prepared a template that you should use for completing your report. The template is structured along the lines of a research paper, and you can fill each appropriate section with the relevant information. The idea is that you get used to using the standard format adopted in research to report on experiments. At times this might feel a little stretched in the context of homework and the exercises you are asked to complete (especially in terms of motivation and related work), but give it a try! I'm sure you can motivate why we would want to work on topic classification of reviews. And surely there is some related work to find as well. Don't get too hung up about what should go where: try to make decisions, and we will give you feedback. Since you will be working in a group, I suggest using Overleaf.

Report structure

To help you a bit, this is a non-exhaustive overview of what is typically expected in a research paper:

- **Title:** pick a relevant one, not simply "LFD assignment 1"
- **Abstract**: a brief summary of your work
  - Why did you do what you did?
  - What exactly did you do?
  - How did you do dit?
  - What did you find?
  - What are the implications of this result?
- **Introduction**
  - Describe the problem, not the assignment
  - Motivate why we want to work on this problem
  - Describe what other people did, and how your work is different (though not too relevant for this homework assignment)
  - Clearly state the research question(s) that you are answering
  - Briefly describe your method and results (a research paper is not a novel)
- **Method**
  - What data set did you use? What are its characteristics? What language is it in?
  - What is your train, dev and test split? Did you use cross-validation?
  - How will you evaluate the results?
  - What algorithms did you use? With what exact settings? How do they work?
  - What features did you use? What did you try? How did you get these features?
  - Reproducibility: other people should be successful in getting the same results
  - Sometimes extra information can go in an Appendix

- **Results**
  - What are your main results? Good to add a clear table with the numbers.
  - Make sure that the tables are self-explanatory, e.g. the caption and headers should contain information about the algorithm, features and whether scores are on dev/test/CV
  - Include a description of the results, do not just refer to a table with the numbers, explain to the reader what it means
  - Do not drown in numbers: make sure that each reported number has a reason to be in the report (sometimes less is more). You don't have to plot a confusion matrix for each experiment. Usually you also do not need precision and recall for *every* experiment, often just the F-score will suffice. There's always the Appendix for less important results
- **Discussion**
  - What are the implications of the findings? Why is that interesting?
  - Error analysis: manually annotate and show a few interesting examples (optional)
  - Extra analysis: maybe performance on a particular class is interesting (optional)
  - Given these results, what would be the next steps for future research?
- **Conclusion**: very similar to abstract, bit more focus on implications of the results and future research