Portfolio-Exam

This is the description (6 pages) of the portfolio exam for the Data Science module MADS-ML – Machine Learning. In this document, the structure of the exam, the portfolio tasks, several rules, and evaluation criteria are described. Please also read the document `lecture_datasets.pdf` from March 02, 2025.

1 Submission

Compressed into one **ZIP**-file, submit

- a Jupyter-Notebook with all your experiments (Section 3) called `experiments.ipynb` and a PDF export of that notebook called `experiments.pdf`,
- an additional notebook called `assembly.ipynb` with all code for assembling the dataset **if** necessary (Section 2),
- a data folder **if** your data is not simply available as a single file online, and
- a resources folder **if** you have additional resources (e.g. images).

Upload your results to Moodle **before 11:59 pm (23:59 o'clock German time) 30.06.2025**.

2 Rules and Conditions

Programming Language Use Python and (where possible) scikit-learn for the submission.

Text Language Choose either English or German for all textual content.

Dataset The dataset may be chosen according to the following three requirements:

- Use **real** data from real applications/sources, no artificial datasets!
- Do **not** use datasets from the lectures. The document `lecture_datasets.pdf` from March 02, 2025 contains a list of datasets that are used in the modules MMS, ML, and DL. **None** of these datasets nor derivatives are allowed for this exam.
- The data must be available: There are two possible availability scenarios:
  - It is available online with proper license. In that case indicate in your notebook where the data can be downloaded.
  - You (legally!) obtain a dataset and share it with me via Moodle. Such data should not come with any form of NDA or other obligations. If you are not sure about these criteria regarding the dataset you consider, please contact me before you invest too much time in the experiments. If you write your own code to acquire data (e.g. querying an API), create a **separate notebook** for that purpose only and submit everything in a zip file.

No Teamwork This exam is supposed to be done during the self study time of the module. Students are allowed to exchange ideas. However, this is **NOT a teamwork exercise**. Every student must derive and write up their own solutions in their own words and programming style.

Code from other Sources You may reuse all the code from this module's lectures and exercises. Copying (and adapting) from other sources is allowed in small quantities – e.g. a function from stackoverflow. Quote the respective source. WARNING: Copying code in large quantities will be treated as intent to deceive and result in a score of zero points.

3 The Portfolio Tasks

Think of the portfolio exam as a report of a (small) project which you – as a data scientist in a data science company – conduct for a customer (a company, an institute, some organization). For that purpose, use a real or fictitious situation in which you (data scientist) solve a problem by conducting experiments according to the following tasks:

Task 1 – Context

This task sets the stage for the subsequent tasks as it determines the context for the experiments. For that purpose:

1. You present a real scenario or a (realistic) fictitious situation, in which you are the data scientist. You propose to solve a **classification or regression problem** and want to convince a customer to green-light your project.
2. You explain the value (monetary or otherwise) of solving the problem and what the customer could do with the model you train (if successful).
3. Name two quality criteria that you will use in the following tasks to evaluate your models. Explain your choice!

Task 2 – The Data

Load and present a dataset (respect the conditions in Section 2). This will be the raw data, that you'll exploit in the subsequent tasks.

- Explain the dataset itself (e.g., what do the features and target represent?).
- Explain how the dataset is suitable for the project from Task 1.

Task 3 – IDA

Conduct an initial data analysis.

- Present some relevant statistical properties that inform the reader about the dataset or that are relevant for your project.

Task 4 – EDA and Preprocessing

Bring the dataset into the form that you need for the experiments.

- Explore the data and conduct necessary transformations.
- Visualize or summarize aspects of the dataset (statistics, ranges, distributions, ...).
- If necessary, use different means of preprocessing until the dataset is suitable.
- If you change data, do not forget to present and summarize relevant properties and distributions of the result.

Task 5 – First Impressions

- Determine an appropriate simple baseline B (no trained model) for your task.
- Select *two* ML algorithms A_1 and A_2 from the lectures (or their regression counterparts) and use their implementations in `scikit-learn`.
- Briefly discuss relevant dataset transformations that have to be used with each of the algorithms. Apply them in **all tasks** where necessary and appropriate!
- In a simple train test split, learn and evaluate the baseline B and both algorithms, A_1 and A_2, in their respective default parametrization. Use the quality measures from Task 1. Present the results in a table and discuss them.
- Evaluate the same, this time using a simple (non-nested) cross validation.
- Present the new results together with the previous ones in a suitable way – select a good representation that supports the following discussion. Compare and discuss the results. Among others adress overfitting, quality, and stability of results.

Task 6 – Hyperparameter Optimization – Nested Cross Validation

Setup a proper **nested** cross validation experiment to assess and compare the performance of the two algorithms and the baseline (A_1, A_2, and B as chosen in Task 5).

- For each of the two ML algorithms A_1 and A_2, create a reasonable hyperparameter grid.
- Use those grids in a nested cross validation to evaluate algorithms in their optimal hyperparameter configurations.
  - For the outer cross validation, reuse the splits from Task 5.
  - Inside the cross validation, use pipeline objects that include preprocessing steps and predictor.
- Compare your final performance estimates to the previous impressions from Task 5.
- Discuss your results and express a recommendation with regard to the choice of model. Pick one model (algorithm with selected hyperparameters) as the final model M for the task.

Task 7 – Feature Importance / Feature Ablation Study

Familiarize yourself with the idea of feature importance through feature permutation.

- Analyze and discuss the most important features according to feature permutation with the final model M (the result of Task 6).

- Plot the feature importances in a suitable diagram. Analyze and discuss the diagram.
- Ablation Study:
  - Pick the two least important features F_1 and F_2 (according to the above results).
  - Reuse the outer cross-validation from Task 6 to train and evaluate the final model M with different feature sets (no further hyperparameter optimization).
  - For that purpose, add another preprocessing step to the pipeline that removes one or more features from the data.
  - Train and evaluate M using your dataset in the cross-validation while excluding F_1 from the data.
  - Train and evaluate M using your dataset in the cross-validation while excluding F_2 from the data.
  - Train and evaluate M using your dataset in the cross-validation while excluding F_1 and F_2 from the data simultaneously.
  - Present and discuss your results. Particularly compare to the impressions from the feature importance study and explain.

Task 8 – Conclusions and Future Work

1. Summarize and interpret the achieved results.
2. Explain the generated value within the context of your task.
3. Propose ideas for future work (a short sketch or enumeration of ideas is sufficient, no further experiments). The ideas should not be too general (e.g., "try further algorithms") but be specific to the project (e.g., "try Algorithm X, as because of Property Y, it might work specifically well on this dataset").
4. Critically reflect and assess the usability of the applied methodology in your context (as described in Task 1). Explain limitations and pitfalls. What could or even should have been done differently? In hindsight, were the goals realistic? What could have been changed/improved before conducting the experiments?

4 Evaluation of the Exam

Your final score will be composed of 15 points for the context (set in the first task and followed throughout the report), 70 points for the actual experiments, and 15 points for presentation. Note,

- that poor presentation can lead to loss of points in the other two categories as well, e.g. if it's too confusing!
- that an unreasonable or trivial data science problem can lead to loss of points in other categories or even to a rejection of the portfolio!

4.1 Context

The (possibly fictitious but) realistic context of the experiments is set in the first task. Especially the last task should relate to the goals and compare the achieved results against them. The report

should follow the story of Task 1 in a straightforward, comprehensive way. The experiments and their results' interpretation should fit to the goals and values proposed.

4.2 Experiments

When grading your experiments, I consider technical soundness, completeness, and fit to the tasks. I expect (among others):

1. The task is a non-trivial data science task.
2. Your code is executable and yields reasonable and reliably replicable results.
3. All cells of the notebook have been executed.
4. The choices of methodology (different approaches, dataset specific choices, settings of hyper-parameters, etc.) make sense and are reasonably explained.
5. The experiments address the influence of random choices and of class imbalance and they use appropriate steps to mitigate them.

4.3 Presentation

When grading the presentation, I will put myself into the position of your project's customer. I expect (among others)

1. that the code is structured (DRY principle, organized imports, . . .) and commented and free of errors or distracting outputs (e.g., no debug messages).
2. that the code is documented using appropriate means of Jupyter.
3. that results are presented in helpful numbers, in tables, and customized diagrams which are referenced and **interpreted** in the text.
4. that diagrams are easy to understand (appropriate colors, tics, scaling, labelling, legends, . . .).
5. that the text is easy to understand (short, concise sentences, proper references to (previous) results, diagrams, . . .).
6. that I am guided through the different parts of the experiments and told what the purpose of upcoming code blocks will be.
7. that the **visualizations are actually visible in the uploaded notebook**! For example, there are known issues with `plotly` graphics (and other libraries), which only show in notebooks when executed directly, but not afterwards, unless explicitly configured.

5 Advice

- Depending on the project and dataset you choose, some of the portfolio tasks may be more or less extensive. E.g. if your dataset is already cleaned up, preprocessing might be very quick. However, in other situations you might want to make use of various preprocessing steps and iteratively refine the dataset before you run algorithms on it.

- Your notebook should contain **much more text**, introduction, comments, etc. than the notebooks we use to demonstrate methods in the lectures or the exercises!
- **You may use bullet points** instead of continuous text!
- Before you submit, re-read the exam description and make sure that all points are addressed.
- Before you submit, re-run your notebook and check that the results are still as expected.
- Keep in mind that this is **not** your master thesis. The experiments should be comprehensive and created and conducted solely by you. However, limit your work to the portfolio tasks, solving ONE problem – even though along the way you might recognize other interesting angles to follow up on.
- For the portfolio, you will choose between a classification and a regression experiment. In the lectures, we focus mostly on classification algorithms. Regression is only discussed in the last part of the module. However, each of the discussed classification algorithms has a regression counterpart, e.g., `KNeighborsRegressor` is the regression algorithm similar to `KNeighborsClassifier`.