INF264
Project 2:
Digit recognizer

Deadline: November 8th, 23.59
Deliver here:
`https://mitt.uib.no/courses/19532/assignments/22282`

Projects are a compulsory part of the course. This project contributes a total of 20 points to the final grade. You need to upload your answer to MittUiB before 23.59 on the 8th of November.

Grading will be based on the following qualities:

- Correctness (your answers/code are correct and clear)
- Clarity of code (documentation, naming of variables, logical formatting)
- Reporting (thoroughness and clarity of the report)

Especially, weight is put to correct use of model selection and evaluation procedures.

Deliverables. You should deliver exactly two files:

1. a PDF report containing an explanation of your approach and design choices to help us understand what you have done. You can include snippets of code in the PDF to elaborate any point you are trying make.
2. a zip file of your code. We may want to run your code if we feel necessary to confirm that it works the way it should. Please include a README.txt file in your zip file that explains how we should run your code. In case you have multiple files in your code directory, you

must mention in the README.txt file which file is the main file that we need to run to execute your entire algorithm. DO NOT INCLUDE THE DATA FILES but refer to them using a relative path.

Programming languages. The course staff supports Python users. Other allowed languages are Java, C/C++ and Matlab.

Code of conduct. Using libraries such as `sklearn` and `keras` is allowed. However, it is not allowed to copy-paste code from online tutorials etc.

Late submission policy: All late submissions will get a deduction of 2 points. In addition, there is a 2-point deduction for every starting 12-hour period. That is, a project submitted at 00.01 on September 9th will get a 4-point deduction and a project submitted at 12.01 on the same day will get a 6-point deduction (and so on). All projects submitted on September 11th or later are automatically failed. (Executive summary: Submit your project on time.) There will be no possibility to resubmit failed projects so start working early.

1 Task: Digit Recognizer

Scenario: You are working for a small company that provides machine learning solutions for its customers. The postal office is developing an AI system to automatically deliver mail. As a part of the system, they need a computer program that recognises handwritten digits. Your company is providing this program and as a machine learning expert, you have been asked to develop such a program.

Task:

1. Write code that produces a classifier
2. Write a report that describes what you have done

1.1 Data

You can download the data from here: `https://mitt.uib.no/files/2017886/download?download_frd=1` and `https://mitt.uib.no/files/2017888/download?download_frd=1`

The data is in the files `handwritten_digits_images.csv` and `handwritten_digits_labels.csv`. The images have the shape

$(70000, 784)$, where each row represents a 28×28 pixel grayscale image ($28 \times 28 = 784$). Each pixel has a value of $0 - 255$ (white to black). The images can be reshaped to a $(70000, 28, 28)$ array, in Python (Numpy) with `x_data = x_data.reshape(x_data.shape[0], 28, 28)`; an image can be visualized using a command `matplotlib.pyplot.imshow(img, cmap="Greys")`. The labels have the shape $(70000,)$, where each row is the label for a corresponding image (labels are $0 - 9$).

1.2 Code

The goal is to produce a classifier that predicts the labels of handwritten digits as well as possible (It is up to you to decide a reasonable way to measure goodness of the solution).

This is not an implementation project and thus you are free to use libraries such as `sklearn` and `keras`.

You should try at least 3 different types of classifiers before choosing the final one. Note that showing effort to optimize performance will affect the grade positively.

It is important to have correct model selection and evalution procedures. Remember training-validation-test splits!

Your results should be reproducible. That is, your customer should be able to verify your claims (Meaning that they should be able to easily run your code and get exactly the same numbers that you give in your report). Thus, you should write an automated test pipeline that runs all of your tests (given enough time). That is, training and assessment of all models and hyperparameters. If you use Jupyter notebooks, make sure that your results can be reproduced after restarting the kernel and running all cells in order.

1.3 Report

The report should consist of two parts (in the same pdf):

1. a summary
2. a technical report.

The summary is should give a short, non-technical overview of your project. You should also argue, based on your results, whether or not the machine learning approach is appropriate for this task and what is your expectation of its performance in real-life.

The technical report that tells what you have actually done and why. It should contain detailed information of your design choices and experimental design. The technical report should contain at least the following information:

- Preprocessing steps
- Candidate algorithms and choice of candidate hyperparameters (and why were the others left out?)
- Chosen performance measure. Justify your choice.
- Model selection schemes that you used. Justify your choices
- What is your final classifier and how does it work. Justify why is it the best choice.
- How well it is expected to perform in production (on unseen data). Justify your estimate.
- Measures taken to avoid overfitting
- Given more resources (time or computing resources), how would you improve your solution?

Our main goal is learning, so it is also ok to report failed experiments. Especially, it is appreciated if you can explain why things didnt work as you initially expected.

Figures and plots are appreciated.

Note: whenever you report performance measures, clearly state which data set you used to compute them.