# Introduction to Data Science 2024
# Assignment 2

Available from Tuesday, February 20th. Your report should be uploaded to Absalon no later than Monday 26th, at 15:00 latest. Total points: 50. Responsible TA: Daniel Expósito Patiño (depa@di.ku.dk).

Guidelines for the assignment:

- **The assignments in IDS must be completed and written individually.** This means that your code and report must be written by yourself.
- Exercises in blue are coding exercises and will be evaluated as such: do provide a working code! Code files must be handed in a single zip file. If some code templates are provided in Absalon, please use them.
- Upload your report as a single PDF file (no Word .docx file) named `firstname.lastname.pdf`. What do include in the report? Check the deliverables of each exercise!
- Upload your code as `firstname.lastname.zip`. It should consist of one or more Python scripts (text files with `".py"` extension) or Jupyter/IPython notebooks (with `".ipynb"` extension). Do not upload the report and source code in a single zip archive. This makes it impossible to use Absalon's SpeedGrader for annotating the reports.
- We will grade using Python 3.10. In some assignments, the specific package versions might be specified in `requirements.txt`. Other versions may work or break; using these versions is safest. Using Anaconda, create an appropriate `conda env` with:
  `conda create --name IDS-A1 python=3.10`
  then activate it with: `conda activate IDS-A1`,
  then install the packages with: `pip install -r requirements.txt`

## Academic Code of Conduct

You are welcome to discuss the assignment with other students, but sharing of code is not permitted. Copying code or text from the report directly from other students will be treated as plagiarism. Please refer to the University's plagiarism regulations if in doubt. For questions regarding the assignment, please ask on the Absalon discussion forum.

In short, plagiarism means copying text or ideas from others without acknowledging the underlying sources. Crucially, this does not mean that you are prohibited from building on others' ideas or use external sources, but rather that you have to properly acknowledge all sources used in your work. This holds for instance for building on code from lectures

or lab sessions. If in doubt, we recommend erring on the side of over- rather than under-acknowledging sources.

While guidance on AI assistance usage is not covered in this course, you are welcome to take a look at the Absalon course on Learning resources for Digital Literacy.

You are also welcome to use AI assistance (e.g., ChatGPT, GitHub Copilot) for tutoring purposes, augmenting the TAs for help with questions and issues. However, keep in mind that their output is not guaranteed to be either comprehensive, true or aligned with the course scope and expectations. Always check with the TAs in case of doubt. Importantly, the use of AI assistance while writing the assignment is allowed only for the following purposes:[1]

- As coding tools (e.g., GitHub Copilot): no restrictions.
- As writing tools to improve the writing of original content, i.e. when the prompt you write contain all the ideas to be formulated: no restrictions.
- As search tools to identify related literature: no restrictions. Usual citation requirements apply (see plagiarism note above): you must cite the original work you identify, even if you used an LLM to find it. Just like you do not cite Google Search for papers you find using it, you should not cite ChatGPT for this either. In particular, always make sure that the citations it provides actually exist—LLMs are known to often generate plausible but nonexistent references.
- As generation tools for *new* ideas: generated content must be clearly highlighted even if post-edited by yourself. All prompts/transcripts from the tools used must be included as an appendix at the end of the submission in PDF, after the references.

For all uses of AI assistance, the purpose, tool, and version[2] must be stated in the submission—e.g, in a dedicated section. Here is such a statement for example:

```
ChatGPT August 3 Version was used as a writing assistance tool and as a
serch tool to identify related literature.  GitHub Copilot July 14 Version
was used while developing the code for...
```

## $k-$Nearest Neighbors Classification

In Assignment 2 you will work with the *nearest neighbors classifier*, a non-linear, non-parametric method for classification. On top of it, we will fine-tune our model using *cross-validation* for selecting optimal hyperparameters and will see how preprocessing our data affects the model's results when we implement *data normalization*.

**Introduction to the problem** Have you ever thought about how some buildings are able to adjust their energy usage based on whether or not people are inside? That's occupancy detection! It typically involves the use of sensors or other types of technology to collect data on movement, heat, or other indicators that suggest the presence of people. The data is then analyzed to determine occupancy patterns and inform decisions on building operations

[1] Note that evaluating LLMs as models on task data is not considered "AI assistance" and is not restricted or affected by the rules here.

[2] If multiple version have been used throughout the assignment, list all of them. In the ChatGPT web interface, the version is specified at the bottom of the screen, under the text input field.

such as lighting, heating, and cooling. Building occupancy detection can help improve indoor air quality, enhance occupant comfort, and most importantly, reduce energy consumption. Different studies for different types of occupancy report savings ranging from 29% to 80%. Also, being able to detect occupancy without resorting to a camera can become necessary in order to avoid privacy concerns and GDPR violations.

**Exercise 1** (Visualization / 10 points)**.** Before training a model to predict the occupancy status of each building we need to take a look at our data. Visualize (1) the distribution of the target variable (occupancy status) in the training dataset (2) a correlation plot to determine which features are strongly correlated to our target variable. Why did you choose these specific plots?

*Deliverables.* Two different plots, the interpretation/description of the plots, and a brief commentary of the reasons behind your plot choice.

**Exercise 2** (1-NearestNeighbor / 10 points)**.**
Apply a nearest neighbor classifier (1-NN) to the data to predict the occupancy status (0-1). You are encouraged to implement it on your own. However, you can also use scikit-learn:

```
from sklearn.neighbors import KNeighborsClassifier
```

After training the model, you are supposed to determine the classification accuracy of your model on the training and test data. One way to do this in Python is the following:

```
from sklearn.metrics import accuracy_score
# given classifier called knn, compute the accuracy on the test set
accTest = accuracy_score(YTest, knn.predict(XTest))
```

*Deliverables.* Accuracy of your 1-NN classifier for train and test data; discussion of the results.

| As a general rule, you must not use the test data in the model building process at all (neither for training, data normalization nor hyperparameter selection - see below), because otherwise, you may get a biased estimate of the generalization performance of the model. |
|---|

### Hyperparameter selection using cross-validation

*Hyperparameters* are settings that are not learned by the model during training but instead are set by the user before training begins. The performance of the $k$ nearest neighbor classifier ($k$-NN) depends on the choice of the parameter $k$ determining the number of neighbors.

Cross-validation involves splitting the available data into several subsets, or "folds", and using each fold as a validation set while training the model on the remaining folds. Repeating the validation step through all folds provides a global accuracy score (correct predictions / all predictions). This provides a way to *select the hyperparameters* that perform best on average and to evaluate the model's ability to *generalize to new data*. Cross-validation is supported in scikit-learn, for example:

```
from sklearn.model_selection import KFold
# create indices for Cross Validation (CV)
cv = KFold(n_splits=5)
# loop over CV folds
for train, test in cv.split(XTrain):
```

```
XTrainCV, XTestCV, YTrainCV, YTestCV = XTrain[train],
XTrain[test], YTrain[train], YTrain[test]
```

**Exercise 3** (Cross-validation / 10 points)**.**

a) You will use a 5-fold cross-validation step to get accuracy scores of a $k$-NN classifier, for $k$ in $\{1, 3, 5, 7, 9, 11\}$. Use only training data.

b) Report $k_{\text{best}}$, the value of $k$ for which the accuracy is the highest and justify: is it always better to consider more neighbors?

*Deliverables.* $k_{\text{best}}$, and a short description of your process + answer to question.

**Exercise 4** ($k_{\text{best}}$ model / 10 points)**.**
To estimate general performance, build a $k_{\text{best}}$-NN classifier using the complete training data set `OccupancyTrain.csv` and evaluate it on the independent test set `OccupancyTest.csv`. Evaluate it also on train data itself and report its accuracy. Interpret the results.

*Deliverables.* Train and test accuracy of $k_{\text{best}}$-NN classifier. Short discussion

## Data normalization

Data normalization is an important preprocessing step that helps transform the data on a similar scale. A basic normalization method is to generate zero-mean, unit variance input data. This ensures that the distribution of each feature is centered at zero with a standard deviation of one, making it easier for the model to learn from the data and reducing the impact of outliers or features with large variances. If you want to learn more about input normalization see the first three pages of Section 9.1 from the chapter available in Absalon Modules Lecture 5 [**?**].

**Exercise 5** (Data normalization / 10 points)**.** Center and normalize the data and repeat the model selection and classification process from Exercise 3 and Exercise 4. However, keep the general rule from above in mind.

You can implement the normalization yourself. First, compute the mean and the variance of every input feature (i.e. of every component of the input vector). Then, find the affine linear mapping that transforms the training input data such that the mean and the variance of every feature are zero and one, respectively, after the transformation.

You may also use scikit-learn. Here are three different ways how one could apply the preprocessing from scikit-learn, only one of which is correct:

```
from sklearn import preprocessing
# version 1
scaler = preprocessing.StandardScaler().fit(XTrain)
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
# version 2
scaler = preprocessing.StandardScaler().fit(XTrain)
XTrainN = scaler.transform(XTrain)
scaler = preprocessing.StandardScaler().fit(XTest)
XTestN = scaler.transform(XTest)
```

```
# version 3
XTotal = np.concatenate((XTrain, XTest))
scaler = preprocessing.StandardScaler().fit(XTotal)
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
```

Discuss which version is correct and why the other two are not (even if you implemented the normalization without scikit-learn). You can briefly discuss what each normalization approach does, and argue for the best way to normalize this specific dataset.

*Deliverables.* Discussion of the three normalization variants including conceptual arguments why two of them are flawed; parameter $k_{\text{best}}$ found in the cross-validation procedure; training and test accuracy of $k_{\text{best}}$; short discussion comparing results with and without normalization

# A The data material

The data was taken from a research project (**?**) from the University of Mons, Belgium, on the detection of occupancy in buildings for the purpose of potential energy saving. On the files `OccupancyTrain.csv` and `OccupancyTest.csv`, which can be found in Absalon, you will see a $400 \times 6$ matrix, where each column corresponds represents the following parameters, in the given order:

- temperature – continuous value measurement (Celsius)
- relative humidity – continuous value measurement (0-100%)
- Light – continuous value measurement (lux)
- CO2 – continuous value measurement (ppm)
- humidity ratio – derived from temperature and relative humidity, in kg-water-vapor/kg-air (0-1)
- occupancy – class label (0=not occupied, 1=occupied)

The original data have been reduced in size to limit computations. We are going to use the data for training and testing a model, and you can find the corresponding training and test data in the files `OccupancyTrain.csv` and `OccupancyTest.csv`.We have already seen how to read a dataset with the Pandas library. In this case we have to use `.iloc` function to select a subset of the columns and `.values` to convert the selected columns to a NumPy array. This is useful for many reasons, such as being able to perform mathematical operations on the column. If we don't use `.values` at the end, the resulting object would be a Pandas Series instead of a NumPy array.

```
import pandas as pd
# read in the data
dataTrain = pd.read_csv('OccupancyTrain.csv', header=None)
dataTest = pd.read_csv('OccupancyTest.csv', header=None)
# split input variables and labels
XTrain = dataTrain.iloc[:, :-1].values # use all rows and all but the last colu
YTrain = dataTrain.iloc[:, -1].values # use all rows, only the last column
XTest = dataTest.iloc[:, :-1].values
YTest = dataTest.iloc[:, -1].values
```

Since we want the data to be in an array format, another way is to load the data directly using Numpy (this is the same as above, so do not do both):

```
import numpy as np
# read in the data
dataTrain = np.loadtxt('OccupancyTrain.csv', delimiter=',')
dataTest = np.loadtxt('OccupancyTest.csv', delimiter=',')
# split input variables and labels
XTrain = dataTrain[:, :-1] # use all rows and all but the last column
YTrain = dataTrain[:, -1] # use all rows, only the last column
XTest = dataTest[:, :-1]
YTest = dataTest[:, -1]
```

## References


Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data.* AMLbook, 2012.

L. M. Candanedo and V. Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buidlings*, pages 28–39, 2016.