

Data Mining 2022/2023

Lab exercises

1 Python

I assume that you know the basics of Python. If you have only a little experience in Python, exercises in this section will guide you through the process of creating a convenient programming environment. At the end of this section you should be able to write and run Python code in Jupyter Notebook and also know basic Jupyter Notebook commands. You should also know how to install new packages and switch environments.

Exercise 1 — Download and install [Anaconda](#). This is distribution of the Python and R programming languages for applications related to data science (see [wiki](#)). Read a quick [user guide](#) and [conda-cheatsheet](#) to learn how to create and switch environments and install new packages.

Exercise 2 — Using Anaconda run Jupyter Notebook and see *Help* → *User Interface Tour, Keyboard Shortcuts*. Learn how to create, move and run cells. Learn how to get information about objects and methods and how to print their source code (e.g. can use *tab*, *shift+tab*, *shift+tab+tab* or write *'?* and *'??'* before a method name).

Exercise 3 — Recall what [data structures](#) are available in Python. Pay special attention to list comprehensions as often they help to write more readable and more efficient code.

Exercise 4 — Install and familiarize yourself with `numpy`, `pandas`, `matplotlib` packages.

- Read [official numpy user guide](#).
- Read [official pandas user guide](#).
- Read [official matplotlib user guide](#).

2 Word-count problem (deadline: 3rd lab)

Exercise 5 — Find the source of your favorite book and save it in UTF-8 format. Load the book and split it into single words. For example you can use a construction like: (5p)

```
with open("Catch_22.txt", encoding="UTF-8") as f:
    words = [word
    for line in f
    for word in line.split() ]
```

Change all words to lower case, remove punctuation and remove stop-words. You may try constructions like:

3. Determine the [tf-idf](#) weights of all words in all documents:

$$tf-idf(t, d, D) = tf(t, d) \times idf(t, D),$$

where t denotes a term (word), d denotes a document and D denotes the collection of all documents. Term frequency $tf(t, d)$ is the number of times a term t appears in document d . Inverse document frequency $idf(t, D)$ is often defined as

$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}.$$

There are packages that make it easy to find tf-idf weights, but try to implement the appropriate procedure yourself.

4. For each document separately build a word cloud using obtained tf-idf weights.

Exercise 7 — Write a function that takes a word as an input and use tf-idf weights to create the list of chapters of your book most matching to that word (i.e. it should return a list of chapters sorted according to appropriate tf-idf weights). (5p)

Exercise 8 — For each given word in your book make a list of five most common words that appear directly after considered word (but ignore stop-words). Use this summary to generate a random paragraph that resembles a paragraph of your book. (5p)

3 Linear Regression (deadline: 4th lab)

Exercise 9 — Using Python solve applied exercises 13 and 14 from Section 3.7 in [ISL book](#). (20p)

Exercise 10 — Read lab1.ipynb and download Auto.csv from my webpage. Read it as dataframe and change the `origin` column to the `category` type. Split the data into the training and validation set. (20p)

- Use `statsmodels` library for linear regression with `mpg` as the response and `horsepower` as the feature. Be prepared to explain parameters returned by `summary()` method that we have discussed, in particular: confidence intervals, p-values, T-statistic, F-statistic and R-squared.
- Create a scatterplot matrix which includes all of the variables in the data set. You can use `pandas.plotting.scatter_matrix(...)`. Compute the matrix of correlations between the variables, you may use `corr()` function for `pandas` dataframe.
- Perform a linear regression with `mpg` as the response and all other variables (except `name`) as the features. Try defining different models with `patsy` library, use symbols `+`, `*`, `:` and different transformations of the variables like for example `I(np.log(X))` or `I(np.sqrt(X))`. For which model you get the best generalization error?
- Try to look for outliers and remove them from the data (see e.g.: residual plot, Z-Score). What are high leverage points? How can you detect them (for example see [here](#))? Retrain your models on cleansed data and compare the results.