# Introduction to Data Science

## WS25/26 Assignment Part 2

December 15, 2025

Prof. Dr. Wil van der Aalst
A. Küsters, N. Graves, L. Liss, C. Rennert, C. Pitsch, C. Schwanen

Chair of Process and Data Science
RWTH Aachen University

## Introduction

In this assignment, you will consider multiple real-life datasets to gain hands-on experience with the topics you learned about in the lecture. Please find an overview of the datasets and the questions they are used for below.

| Dataset | Question and Purpose |
|---|---|
| `groceries_basket.csv` | Q1: Basket items for market basket analysis |
| `frequent_itemsets.csv` | Q1: Frequent itemsets discovered from the basket data |
| `fines_event_log.xes` | Q2: Event log for process mining |
| Files in folder `nlp` | Q3: Transcripts of the lecture recordings |
| `air_traffic.csv` | Q4: Historical US flight data |
| `flights_train.csv` | Q4: Train split for forecasting |
| `flights_test.csv` | Q4: Test split for forecasting |
| `event_log.csv.gz` | Q5: Very large event log |

## Assignment Details

- Total number of points obtainable: 100 (20 % of final course grade)
- Group size: 2–3
- Input:
  - JupyterLab template, and
  - the data as described above
- Deadline: 22.01.2026, 23:59:59 (CET)
- Please make sure to join a group in Moodle until the 18.12.2025 23:59:59 (CET)
- Try solving group issues internally. However, as a last resort, send an email **with all group members in CC** to the IDS email address if there are any problems with group formation that require our involvement. Please note that we will not change any groups after 12.01.2026.
- Deliverables:

  - PDF report (max. 20 pages)
  - Jupyter notebooks (one per question).

**Report** Your written report is the main basis for grading. In your report, you should present your methods, motivation, results, and explanations. Doing so, **clearly indicate which answer belongs to which question**. Please order your questions according to the numbering of this assignment to avoid confusion. Moreover, it should be **self-contained** (i.e., it should not require references to the notebook). The length should be at **most 20 pages**, including the title page. Make sure to include **all group members' names on the title page**. Please respect the following reporting criteria (severe problems may lead to a point deduction of up to 10 points):

- Proper spelling, punctuation, readability, and comprehensive structure
- Use of **adequate** visualizations for showing (aggregated) results or illustrating methods
- The precision of all decimal numbers should be up to two decimal digits
- Figures have captions, axes have labels, and diagrams have headers.
- Figure quality (e.g., resolution and relevance)
- All figures, tables, and similar are numbered and referred to in the text
- All your comments, descriptions, discussions, and interpretations should be explicit and concise. Usually, no more than 2-3 sentences are needed to answer individual parts of the question.

**Notebooks** Your Jupyter notebooks will be used for reference and potentially for testing your code. Therefore, it should satisfy the following requirements:

- Commented and structured code (if not, this will be penalized in the style points)
- Questions separated by markdown headers
- Top-to-bottom runnable cells to reproduce your results
- DO NOT CLEAR THE OUTPUT of the notebooks you are submitting!
- It should be runnable in the bundled conda environment.
- Ensure that the code in the notebook runs if placed in the same folder as all the provided files, delivering the same outputs as the ones you submit in the notebook and report on in your report.

Do not re-upload the data. Besides, notebooks that intentionally access files outside the notebook's directory or are in any way harmful will be graded with zero points.

## Optional Resources

- Jupyter: `https://jupyter.org/index.html`
- Jupyter Lab/Notebook installation guide on Moodle

| Question: | 1 | 2 | 3 | 4 | 5 | Total |
|---|---|---|---|---|---|---|
| Points: | 23 | 17 | 20 | 23 | 17 | 100 |

## Question 1: Market Basket Analysis (23 points)

For this question, you will work with an excerpt of a real-life dataset from an online grocery store. Every row in the dataset (`groceries_basket.csv`) describes the placement of a product into a shopping basket (i.e., an order) by a customer. The dataset contains 573,124 rows, refers to 64,864 orders, and 26,548 different products. It contains the following columns:

| Column | Description |
|---|---|
| `product_id` | A short identifier for the product added |
| `order_id` | The identifier for the order |
| `product_name` | The name/description of the product |
| `category` | The category the product belongs to (e.g., dairy,...) |
| `add_to_cart_sequence_index` | Index of the product being added to the cart |

Until stated otherwise, only use the products added to the cart without considering their sequence index, i.e., all products with the same *order_id* form a transaction. In this task, you will explore the dataset, extract and evaluate frequent itemsets and association rules, and compare the frequencies of sequences.

(a) (1.5 points) First, you want to get an overview of the products and their frequencies. Answer the following questions:

1. How often must a product be ordered (i.e., part of how many orders) for it to have a support of at least 1%?
2. What is the mean support count across all products?
3. What is the median support count across all products?

(b) (2 points) Provide the 10 most frequent product IDs and their absolute and relative frequency. Considering this list, what is the maximum support that could be achieved for an itemset of size 3? Which product IDs would be part of this itemset?

(c) (1.5 points) Now, consider the itemset of size 3 you identified in the last task. How many of the orders in the dataset contain all three products? Considering the product names of these products, how would you explain this low number of orders (1-2 sentences)?

(d) (2 points) Next, you want to understand what kind of products the company offers and sells. Provide the following two plots concerning the product categories:

- a plot showing the **total** number of sold products per category (basket items), and
- a plot showing both 1) the mean and 2) the median number of items **per order** for each product category.

Order the categories on the x-axis alphabetically.

(e) (1 point) What do mean and median tell you about the purchase behavior regarding beverages? Answer in 2-3 sentences.

The file `frequent_itemsets.csv` contains all frequent itemsets with a support count of at least 100. The itemsets were determined using the FP-Growth implementation of the `mlxtend`-library. Use these frequent itemsets for the remaining tasks.

(f) (2 points) Give answers to the following:

1. Provide the names of the products in the largest frequent itemset, i.e., the most products.

2. How many frequent itemsets of size 2 or larger are there (absolute and relative).
3. Which item appears in the most frequent itemsets and in how many frequent itemsets does it appear?
4. How many products appear in only a single frequent itemset?

(g) (1 point) Consider the frequent itemsets of **size 2 or larger** and the product categories of the involved products. Create an equivalent *category set* for each frequent itemset showing the categories of the involved products, i.e., a frequent itemset containing products $\{banana, apple, milk\}$ would correspond to the category set $\{produce, dairy\}$. Provide a plot showing the different category sets on the x-axis and their frequency on the y-axis. Make sure the x-axis tick-labels are readable, clearly label the category set, and ordered by frequency (descending).

(h) (3 points) Considering the category sets from the previous question you make the following three observations:

1. The majority of frequent itemsets of size 2 or larger contain only *produce* products.
2. *Produce* products are in nearly all frequent itemsets (all but 10).
3. Despite many orders not including *beverages* at all, there are four frequent itemsets only containing *beverages*[1].

Explain why each of these observations is not unexpected in 1-2 sentences, and what observation 3 implies about the diversity of sold beverages. *Hint: Observations from previous tasks help you here.*

(i) (2 points) Determine all association rules with a minimum support count of 100 and a confidence threshold of 0.2 using the `mlxtend`-library. Answer the following questions:

1. What are the mean and median lift across all rules?
2. How many association rules contain at least one of the top-10 most frequent products in the antecedent?
3. How many association rules contain at least one of the top-10 most frequent products in the consequent?

(j) (1.5 points) Provide and interpret the association rule with the highest confidence in no more than 1-2 sentences. Furthermore, explain in max. 3 sentences why it is not unexpected to see the product with the highest frequency in the rule as the consequent rather than the antecedent.

(k) (1.5 points) Look at the association rules that do not contain any of the top-10 most frequent products in either the antecedent or the consequent. What is striking about their lift compared to the overall set of rules? Explain why this is not surprising in 2-3 sentences.

(l) (2 points) Now you consider every order as a sequence of products added to the cart in the order specified by the *add_to_cart_order* column. Consider all association rules where both $A \implies B$ and $B \implies A$ meet the desired threshold. From your own experience in shopping, you know that sometimes adding one item to the cart reminds you to buy something else. You now wonder whether this might also be the case for the products in these bi-directional rules. For each pair of products in the bi-directional rules (in all of them the sets only contain a single item, i.e. $A = \{a\}$ and $B = \{b\}$), use the *add_to_cart_order* column to determine the support count for the possible subsequences (i.e., $\langle a, b\rangle$ and $\langle b, a\rangle$). Provide the support count for each subsequence in a table. You can use the function to determine containment you implemented in the exercise notebook (added to the assignment notebook).

[1] and there are only three categories with frequent itemsets of only one category

(m) (2 points) Briefly explain why it is not possible to find an association rule $A \implies B$ with the following metrics using the provided dataset:

- $support\_count(A \implies B) \geq 100$,
- $confidence(A \implies B) > 0.2$, and
- $lift(A \implies B) < 1$.

*Hint: You might want to revisit the definitions of the metrics and your answers for task b) for this.*

# Question 2: Process Mining (17 points)

The file `fines_event_log.xes` contains a sample of a real-life event log for a process of handling traffic fines in a different country. The excerpt contains 267,252 events of 71,522 cases, includes 11 different activities, and an overall description of the process is as follows: After an offence is detected, a fine is created. If the offender does not pay directly, the fine is sent to the offender via post; this causes a postal expense which the offender has to pay as well. Not paying the due amount within a certain time period leads to an additional penalty that has to be paid as well. An alternative to paying the due amount is to get the fine dismissed by appealing to a judge or the prefecture. If a due amount is not paid, the case is passed to the credit collection. In the data, every row describes an event and you find the following columns:

| Column | Description |
|---|---|
| `case:concept:name` | Case ID |
| `concept:name` | Activity name |
| `time:timestamp` | Timestamp of the event |
| `case:fine` | The fine the offender has to pay (case attribute) |
| `expense` | The expense caused by the postal fee |
| `penalty` | The penalty added because the charges were not fully paid |
| `paymentAmount` | The amount of money paid by the event offender |
| `dismissal` | Not NA if the case is dismissed |

In the following questions, you will work with the provided event log to gain more insights into the process and create a process model describing most de facto, end-to-end process executions. Use the `pm4py` library for discovery and conformance checking. Make sure to discover all models with the standard *inductive miner*, i.e., leave all parameters at their default values (no noise thresholds!).

(a) (3 points) To get an overview of the described behavior, you decide to discover a process model (process tree or Peri net) from the full event log with the inductive miner. You know that the inductive miner guarantees that every trace in the log can be completely replayed on the discovered model, i.e., every behavior seen in the data will be allowed by the model. Answer the following questions about the process only based on your discovered model:

1. With which activities can the process start?
2. Which activities *must* be executed in a trace containing an appeal to a judge and in what order?
3. Which activities can be executed more than once?
4. Is it possible that the credit collection is involved without the fine being sent to the offender via the post?

(b) (2.5 points) You notice that the discovered model allows for a lot of behavior, some of which you suspect not to be present in the data. Provide the number of cases in the event log that exhibit the following behavior (all of which is allowed by the model):

1. A payment is made but the case is still sent for credit collection.
2. A penalty is added before the fine is sent to the offender via post.
3. An appeal to the prefecture or a judge is made after a payment.
4. The offender is notified about the result of an appeal without any appeal having been made.

(c) (4.5 points) After these insights, you would like to investigate the different variants in the event log. Provide the following:

1. A chart showing the cumulative variant frequency, i.e., percentage of cases covered by the fewest variants. The $x$-axis should show the number of variants (ordered from left to right in increasing frequency) and the $y$-axis should depict the relative accumulated number of cases. Add a count of zero cases for a non-existent 0th variant (very first entry on the x-axis) to make the plot start at $(0, 0)$.
2. An interpretation of the chart (2 sentences).
3. The five most frequent variants and the number of cases they cover.

*Hint: To determine each variant's value on the y-axis you need to sort the variants by the number of cases they cover (descending) before determining the accumulated sum of the percentage of cases covered.*

In the process, a case is considered as closed if one of the following situations arises:

- The offender has paid at least the full due amount, i.e., fine + expense + penalties.
- The case has been dismissed (the `dismissal` attribute is non-na).
- The case has been sent for credit collection.

In the provided data, every case falls into exactly one of these categories or is still open (none of the above applies).

(d) (2 points) Provide a pie chart showing the share of cases that are: 1) still open, 2) closed because the full amount was paid, 3) closed because the case was dismissed, and 4) closed because it was sent for credit collection. Include the relative frequencies for each category in the chart. *Hint: From your previous analysis you know that no case includes two or more expenses/penalties, but there can be multiple payments.*

(e) (3 points) Create a sublog with only the closed cases (57756 cases). Filter this sublog to include only the five most frequent closed-case-variants. Provide a Petri net discovered from the filtered log. Comment (3-4 sentences) on the differences you observe between this model and the one discovered from the full event log (part a)). Focus on 1) the activities involved, 2) the timing of the payments, and 3) the relation between making a payment and sending for credit collection. *Hint: If you are uncertain you classified the cases correctly provide the variants from the event log you used for the discovery.*

(f) (2 points) Determine the fitness of the full log by applying token-based replay on the model discovered from the filtered log in the previous part. Provide the percentage of perfectly fitting traces and the log fitness value. Explain why the log fitness is so much higher than the percentage of perfectly fitting traces.

# Question 3: Natural Language Processing (20 points)

In this question, you will analyze transcripts extracted from the IDS lecture recordings. The relevant datasets are in the `nlp/` folder. Each transcript is saved as a CSV file and contains multiple automatically generated *segments* as rows. For each row, the dataset has the following columns:

| Column | Description |
|---|---|
| `start` | The start timestamp of the lecture segment in seconds. |
| `end` | The end timestamp of the lecture segment in seconds. |
| `text` | The text of the lecture segment. |

First, load all CSV files and concatenate them to one large dataframe, adding a column `lecture` to indicate the lecture number and name (e.g., `01-introduction`).

*Hint: The full dataframe should have a shape of* $(1793, 5)$.

(a) (2 points) Plot the 25 most frequent words in the whole dataset as a histogram. As words, consider space-separated sequences of characters without any additional preprocessing. Point out two problems of this very basic approach that manifest in the histogram and how they can be addressed.

(b) (2 points) Next, preprocess the dataset by normalizing all casing to lowercase, removing punctuation, and tokenizing the text using the `nltk punkt_tab` tokenizer. The preprocessed tokens should be added as the column `tokenized_text` in the dataframe.

Create a histogram containing the 25 most frequent tokens used in `01-introduction`. Don't include stopwords.

*Hint: Write a function that takes a string as input and returns a list of tokens after preprocessing. You will need it for the next parts as well.*

(c) (2 points) Create a stacked histogram showing the frequency of the following tokens in the dataset, grouped and colored by lecture: `data`, `decision`, `predict`, `derivative`, `network`, `easy`, `database`. Show the resulting figure and briefly describe what you observe in 2–3 sentences.

(d) (6 points) Implement an n-gram language model to generate text in the style of the IDS lectures. The implementation should, based on a given text input (seed text), generate a sequence of next words[2]. For your implementation, consider the following:

- Construct n-grams from the tokenized text from all lectures, treating each lecture segment separately. Use the padding tokens `<s>` and `</s>` for the start and end of sentences.
- Initialize a `ConditionalFreqDist` from `nltk` to store the counts of word sequences. In particular, for each n-gram, it should contain the counts of the next token given the previous $n-1$ tokens.
- Implement a function to predict the next word given a context of $n-1$ words, by sampling from the distribution of possible next words. For that, use the `random.choices` function with the weights derived from the `ConditionalFreqDist` model. Seed the random selection using the seed 3213[3] before each random choice selection, and sort all possible next words lexicographically (e.g., using `sorted(...)`) before sampling.

[2] The generated text may not always be coherent, but it should reflect the style and vocabulary of the lectures.

[3] Seeding helps ensuring reproducible and consistent results.

- Implement a function to generate a sequence of words given an input context (seed text). Make sure to apply the same preprocessing (lowercasing and tokenization) to the input context as you do to the dataset. In particular, pad the context with `<s>` tokens if it is shorter than $n-1$ words. During generation, stop if the model predicts the `</s>` token or if the context has never been seen before.

Using your implementation, generate text sequences for the input text `"introduction to data"`. Generate a sequence of at most 30 tokens (i.e., 33 tokens including the seed text). Provide the resulting text for $n \in \{3, 4, 5, 2^4\}$. Comment on the differences. What behavior do you expect from $n > 5$ and why?

(e) (8 points) Implement hierarchical TF-IDF-based timestamp retrieval for the IDS lectures. Given some input query, first retrieve the top-$k$ most relevant lectures, and then within these lectures retrieve the top-$m$ most relevant timestamps. Return these nested results as a list of $k$ lectures with their names (e.g., `01-introduction`) and with a list of $m$ timestamps (i.e., start and end in seconds) each. Use the entire tokenized dataset (all lectures) to build the model. Consider the following requirements:

- Use the `TfidfVectorizer` from `sklearn`.
- Use the preprocessed tokens, but additionally apply stopword removal and stemming, considering the source language (English).
  *Hint:* Pass a custom `analyzer` function to `TfidfVectorizer`.
- For stemming, use the `SnowballStemmer` from `nltk`.
- For stopword removal, use the stopwords list of `nltk`.
- The corpuses to consider on the different levels are defined as follows:
  1. On the first level, the corpus consists of one document for each lecture, aggregating all the tokens of its lecture segments by concatenation.
  2. On the second level for a given lecture, the corpus consists of one document for each lecture segment of the lecture.

  Remember to consider the correct corpus when computing the TF-IDF representations. Given the size of the dataset, it is also acceptable to newly create and fit `TfidfVectorizer`s for each query.
- Make sure to apply the same preprocessing (lowercasing, tokenization, stopword removal, stemming) to the input query as you do to the dataset.

*Hint: Use the `dot` function of the sparse matrix to calculate the cosine similarity.*

Apply your implementation to the following queries using $k = 2$ and $m = 2$.

1. `gradient descent approach`
2. `beer and diapers`

Report the retrieved lectures and timestamps for each query in the retrieval order (best match first). Moreover, report the score of the retrieved lectures. Comment on the results you observe for each query in 1–2 sentences, also examining the text of the matching segments.

[4] You can of course also try out more values!

# Question 4: Time Series Analysis (23 points)

In this question, you will investigate historical data on US flights. The dataset contains the monthly counts of domestic and international flights for each year. A subset of the columns is explained in the table below.

| Column | Description |
|---|---|
| `Year` | The year of the record |
| `Month` | The month of the record |
| `Dom_Pax` | The number of passengers on domestic flights |
| `Int_Pax` | The number of passengers on international flights |
| `Pax` | The overall (domestic and international) number of passengers on domestic flights |
| `Dom_Flt` | The number of domestic flights |
| `Int_Flt` | The number of international flights |
| `Flt` | The overall (domestic and international) number of flights |

(a) To begin, you explore the data.

i. (1.5 points) Load the dataset `time-series/air_traffic.csv`.

1. What time range does it cover?
2. How many months are tracked?
3. How many flights are included overall?

ii. (2 points) Create and include a time series plot for the overall number of flights per month and describe it.

iii. (2 points) The dataset also contains information on the number of passengers. Plot the number of passengers against the number of flights in a time series plot and compare them. Do they show similar or different trends?

(b) Now that you have an overview of the dataset, it is time for a more in-depth analysis of the time series.

i. (3 points) To further investigate the seasonal effects in the time series, create and include yearly seasonal plots for the flights and passengers time series, and describe them in 2-3 sentences. Are there consistent seasonal effects? If yes, are there any years that do not follow this pattern?

ii. (2 points) Compute and interpret the correlation coefficient between the time series for passengers and flights. How does this relate to your findings in subtask a.iii?

iii. (4.5 points) Using the `statsmodels` package, create and include correlograms for the *flights* time series after applying differencing $\{0, 1, 2\}$ times. Include lags up to 24 months. Describe the correlograms. For which lags (apart from 0) is the correlation the most significant in all correlograms?

iv. (3 points) Using the `statsmodels` package, perform a Season-Trend Decomposition (`STL`) using *LOESS* for the *flights* time series. Use a period of 12 months. Is the residual time series stationary? Provide a figure showing the trend, seasonal, and residual components.

(c) Finally, let us consider the task of forecasting. To this end, solve the following tasks using `time-series/flights_train.csv` and `time-series/flights_test.csv` as train and test sets, respectively.

i. (1 point) In 2-3 sentences, comment on the suitability of the flights and passengers time series for forecasting.

ii. (2 points) Using the performance measures **RMSE** (root mean squared error), **MAE** (mean absolute error), and **MAPE** (mean absolute percentage error), fit a `NaiveForecaster` with the “mean” strategy and a `StatsModelsArima` forecaster using the `sktime` package. Experiment with the *order* parameter $(p, d, q)$ of the ARIMA model to find a model that achieves better performance than the naive forecaster in all metrics. Report the parameter setting along with the achieved metrics.
*Hint: Sensible ranges for the ARIMA parameters are* $1 \leq p,\ q \leq 12,\ 0 \leq d \leq 3$

iii. (2 points) Create a plot showing the train and test portions of the time series along with your forecast. Does your model provide a good forecast?

## Question 5: Distributed Data Processing (17 points)

In this task, you will investigate how process mining can be applied to large datasets.

Parts of this task will be solved using *Apache Spark*. For this, we have provided a `docker-compose.yml` file. To start a jupyter notebook with access to an Apache Spark server, run the commands:

```
docker compose up -d
docker exec -it spark-master bash /opt/bitnami/spark/start-jupyter-lab.sh
```

in the directory containing the compose file and notebook. Then, click on the link provided by JupyterLab[5]. Any files in the `big-data` directory will be accessible in the notebook.

In particular, this task will use the `big-data/event_log.csv` dataset, which has the columns `CaseID`, `Activity`, `Timestamp`, and `Resource`. For convenience of computation using the MapReduce paradigm, the header has been removed. However, the columns follow the previously described order.

(a) (4 points) To get an overview of the process, you want to discover a process model. To this end, you first want to extract a Directly Follows Graph (DFG) using MapReduce. How can you compute a DFG using MapReduce? Please provide the mathematical function signatures for the *map* and *reduce* functions you use and explain what they do. For this, you may use the following domains:

- $\mathbb{S}$: The domain of strings
- $\mathbb{N}_0$: integers (e.g., line numbers and counts)
- $C \subseteq \mathbb{S}$: The domain of case ids
- $A \subseteq \mathbb{S}$: The domain of activities
- $T \subseteq \mathbb{N}_0$: The domain of timestamps

*Hint: The input domain of the first map function is $K \times V = \mathbb{N}_0 \times \mathbb{S}$. The key is the line number.*

(b) (5 points) Another technology covered in the lecture is *Apache Spark*, which in part is based on similar concepts as MapReduce. Discover a DFG from `event_log.csv` using *Apache Spark*. List the `pyspark` RDD transformation functions you used to compute the directly follows relation counts and explain on a high level what each transformation did. Additionally, please provide a figure of the discovered DFG.

*Hint: Some of the following snippets and functions may be helpful:*

- `sc = pyspark.SparkContext.getOrCreate()`
- `rdd = sc.textFile("path/to/file.txt")`
- `rdd.flatMap`
- `rdd.map`
- `rdd.groupByKey`
- `rdd.mapValues`
- `rdd.reduce`
- `rdd.reduceByKey`

(c) (3 points) The computed DFG contains many arcs with low weights, making it complex and hard to interpret. It would be easier if the less common arcs were removed.

[5] The link will have the form `http://127.0.0.1:18888/lab?token=…`

1. Briefly explain how you would update your MapReduce solution to remove arcs with a frequency less than 100.
2. Update your Apache Spark implementation to remove directly follows relations with a count less than 100. List and explain any new or changed transformations.
3. Create and include a DFG from the filtered directly follows relations.

(d) (5 points) Finally, the event log also contains *resource information*, i.e., *who* executed the activity. To learn more about the process, we would like to know for each event which resource most commonly executes it. Implement this using `pyspark` and list the transformation functions you used and explain them. Finally, for each activity, provide its most common resource value.

*Hint: Some activities do not have a resource recorded. In this case, it has been filled with the resource "MISSING". For simplicity, you can treat this as an actual resource.*