AMSE / SAKI SS 2023 - Homework

Advanced Methods of Software and Data Engineering

Please see slide deck A01 on the handling and grading of homework.

Table of Contents

Project Work

You will develop a data science project over the course of the semester.

- The project work is conducted self-organized by the student
- The project has to be based on at least two different open data sources
  - One of them must be from Mobilithek
  - Other portals to consider: https://data.europa.eu/, https://www.govdata.de/
  - Tell your fellow students about interesting data sources in the course forum: https://www.studon.fau.de/frm5003028.html
- The technology stack used for the project is as follows
  - Data engineering: Development stack of your choice
    - The result is a script you commit to your repository (*/data* directory)
  - Data exploration: Python notebook
  - Data analysis: Python notebook

Project Work 1

- Create a public GitHub repository for the module by forking https://github.com/jvalue/2023-amse-template
- Answer Google Form with a link to your repository

    - https://forms.gle/kr63XtCW78eMQkNU8
- Set up Python / Jayvee / Jupyter Notebook environment

Project Work 2

- Submit project plan
    - As markdown file *project-plan.md* in the *project* directory of your repository
    - Follow the template to do so
    - Create coarse-grained issues that layout the working packages you will work on during the semester and link them in section “Work Packages” in the project plan. This plan is allowed to be enhanced and changed over the course of the semester. Please keep it up-to-date.

Project Work 3

- Follow your project plan to build an automated data pipeline for your project
    - Create a **/data** folder in the root of your project repository
    - Write a data engineering script to pull, massage, and store your data
        - Place the script in the **/data** directory
    - Output: local datasets in /data directory (e.g., as SQLite databases)
        - Make sure to **NOT** check in your data sets
        - You can use .gitignore to avoid checking in files on git
- Update the issues and project plan if necessary

Project Work 4

- Continue working on your project
    - Complete your exploratory data analysis
- Add automated tests for your project
    - Add a **/project/tests.sh** file that executes your tests
    - There should be at least one test case on the system-test level
        - Executes your data pipeline
        - Validates that the output file(s) are there
- Update your issues if necessary

Project Work 5

- Continue working on your project
    - Start working on your final report
- CI for your project
    - Use GitHub Actions
    - There should be at least test execution on every push
        - Executes the test.sh (from project work 4)
        - If your tests are too expensive, comment out the execution line
- Update the issues if necessary

Project Work 6

- Final report
    - Your final report should be a standalone file that can be read by someone unfamiliar with your project, to give you more options you can submit your notebook or export it

as either pdf or html

        - If you submit as ipynb, make sure GitHub preview renders your report correctly
    - Commit and push **one of** the following files containing your final report to GitHub
        - **/project/report.ipynb**
        - **/project/report.pdf**
        - **/project/report.html**
- Optional: Project presentation slides
    - If you would like to get a chance to present your project to the class, submit these slides
        - Willingness to hold a presentation will increase your final grade, if you are randomly selected you will be required to present
    - We will choose a random sample of presentations for week 12/13
        - We will announce the selection in StudOn and the Q&A in week 11
    - Presentations should last a **maximum of 10 minutes**
        - **Focus should be your data science question, data sources and the report**, not the implementation details of your data pipeline
        - Done via Zoom, you share your screen and have a microphone
    - Commit and push the following file containing your project presentation slides to GitHub
        - **/project/slides.pdf**

Project Work 7

- Fill out the Course Exit survey to let us know about your learning progress
    - Course Exit Survey Link

Exercises

- Place all exercise submissions into the folder named “**exercises**” in your GitHub repository
    - Name exercise submissions either “exercise<number>.jv” or “exercise<number>.py”
        - For example, if you are in the group using Jayvee for exercise 1, you must submit the following file: **/exercises/exercise1.jv**, if you have to use Python, submit **/exercises/exercise1.py**
- We will grade the version of the exercise that is in your public GitHub repository at the deadline (!)
- Group Assignment: https://www.studon.fau.de/file5191333_download.html

Exercise 1

- Build an automated data pipeline for the following source:
    - https://mobilithek.info/offers/-8691940611911586805
    - direct link to CSV: https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-flughafen-weltweit/exports/csv
- Goal
    - Write data into a SQLite database called “**airports.sqlite**”, in the table “**airports**”
    - Assign fitting built-in SQLite types (e.g., BIGINT, TEXT or FLOAT) to all columns
    - No further data validation is required, do not drop any rows or change any data points
- **Group Assignment**: https://www.studon.fau.de/file5191333_download.html

- Group A, use Jayvee
  - Jayvee version 0.0.15
- Group B, use Python
  - Python 3.11
  - Allowed libraries: https://github.com/jvalue/amse-exercise-feedback/blob/main/requirements.txt
    - Notable pandas, SQLAlchemy
- Submit as **/exercises/exercise1.jv** or **/exercises/exercise1.py** depending on your language
- **Optional, after exercise**, give us feedback by participating in the survey:
  - English https://oss.cs.fau.de/surveys/index.php/758772?lang=en
  - German https://oss.cs.fau.de/surveys/index.php/758772?lang=de

Exercise 2

- Build an automated data pipeline for the following source:
  - https://mobilithek.info/offers/-8739430008147831066
  - direct link to CSV: https://download-data.deutschebahn.com/static/datasets/haltestellen/D_Bahnhof_2020_alle.CSV
- Goal
  - Write data into a SQLite database called "**trainstops.sqlite**", in the table "**trainstops**"
  - First, drop the "Status" column
  - Then, drop all rows with invalid values:
    - Valid "Verkehr" values are "FV", "RV", "nur DPN"
    - Valid "Laenge", "Breite" values are geographic coordinate system values between -90 and 90
    - Valid "IFOPT" values follow this pattern:
      - <exactly two characters>:<any amount of numbers>:<any amount of numbers><optionally another colon followed by any amount of numbers>
    - Empty cells are considered invalid
  - Use fitting SQLite types (e.g., BIGINT, TEXT or FLOAT) for all columns
- **Group Assignment**: https://www.studon.fau.de/file5191333_download.html
- Group A, use Python
  - Python 3.11
  - Allowed libraries: https://github.com/jvalue/amse-exercise-feedback/blob/main/requirements.txt
- Group B, use Jayvee
  - Jayvee version 0.0.15
- Submit as **/exercises/exercise2.jv** or **/exercises/exercise2.py** depending on your language
- **Optional, after exercise**, give us feedback by participating in the survey:
  - English https://oss.cs.fau.de/surveys/index.php/443882?lang=en
  - German https://oss.cs.fau.de/surveys/index.php/443882?lang=de

Exercise 3

- Build an automated data pipeline for the following source:
  - Link to data offer: https://mobilithek.info/offers/-655945265921899037
  - Direct download link: https://www-genesis.destatis.de/genesis/downloads/00/tables/46251-0021_00.csv

- Goal
    - Write data into a SQLite database called “**cars.sqlite**”, in the table “**cars**”
    - Pick suitable encoding:
        - Make sure to preserve the german special letters like “ü” or “ä”
    - Reshape data structure
        - Ignore the first 6 lines and last 4 lines as metadata
        - Keep only the following columns, rename them to the new name given here (M-BU contain summary data)
            - Column A: date
            - Column B: CIN
            - Column C: name
            - Column M: petrol
            - Column W: diesel
            - Column AG: gas
            - Column AQ: electro
            - Column BA: hybrid
            - Column BK: plugInHybrid
            - Column BU: others
        - Drop all other columns
    - Validate data
        - date/name are strings, no need to validate date
        - CINs are Community Identification Numbers, must be strings with 5 characters and **can have a leading 0**
        - all other columns should be positive integers > 0
        - drop all rows that contain invalid values
    - Use fitting SQLite types (e.g., BIGINT, TEXT or FLOAT) for all columns
- Group A, use Jayvee
    - Jayvee version **0.0.16**
        - For instructions how to update your interpreter and VSCode extension, see the Introduction to Jayvee lecture: https://www.youtube.com/watch?v=X8fOK8JIBZA
- Group B, use Python
    - Python 3.11
    - Allowed libraries: https://github.com/jvalue/amse-exercise-feedback/blob/main/requirements.txt
- Submit as **/exercises/exercise3.jv** or **/exercises/exercise3.py** depending on your language
- **Optional, after exercise**, give us feedback by participating in the survey:
    - English https://oss.cs.fau.de/surveys/index.php/892835?lang=en
    - German https://oss.cs.fau.de/surveys/index.php/892835?lang=de

Exercise 4

- Build an automated data pipeline for the following source:
    - Link to data offer: https://mobilithek.info/offers/526718847762190336
    - Direct download link: https://www.mowesta.com/data/measure/mowesta-dataset-20221107.zip
- Goal
    - Download and unzip data
        - Use the “data.csv” in the zip file
        - *for Python, consider using ‘urllib.request.urlretrieve’ instead of the request*

        library to download the ZIP file
        - *for Jayvee, if you use the FilePicker, do not use a leading dot in file paths, see this bug: https://github.com/jvalue/jayvee/issues/381*
    - Reshape data
        - Only use the columns "Geraet", "Hersteller", "Model", "Monat", "Temperatur in °C (DWD)", "Batterietemperatur in °C", "Geraet aktiv"
        - Rename "Temperatur in °C (DWD)" to "Temperatur"
        - Rename "Batterietemperatur in °C" to "Batterietemperatur"
        - There can be multiple temperature measurements per row
            - discard all columns to the right of "Geraet aktiv"
    - Transform data
        - Transform temperatures in Celsius to Fahrenheit (formula is (TemperatureInCelsius * 9/5) + 32) in place (keep the same column names)
            - Columns Temperatur and Batterietemperatur
    - Validate data
        - Use validations as you see fit, e.g., for Geraet to be an id over 0
    - Use fitting SQLite types (e.g., BIGINT, TEXT or FLOAT) for all columns
    - Write data into a SQLite database called "**temperatures.sqlite**", in the table "**temperatures**"
- Group A, use Python
    - Python 3.11
    - Allowed libraries: https://github.com/jvalue/amse-exercise-feedback/blob/main/requirements.txt
- Group B, use Jayvee
    - Jayvee version 0.0.16
- Submit as **/exercises/exercise4.jv** or **/exercises/exercise4.py** depending on your language
- **Optional, after exercise**, give us feedback by participating in the survey:
    - English https://oss.cs.fau.de/surveys/index.php/239397?lang=en
    - German https://oss.cs.fau.de/surveys/index.php/239397?lang=de

Exercise 5

- Build an automated data pipeline for the following source:
    - Link to data offer: https://mobilithek.info/offers/110000000002933000
    - Direct download link: https://gtfs.rhoenenergie-bus.de/GTFS.zip
- Goal
    - Work with GTFS data
        - *for Python, consider using 'urllib.request.urlretrieve' instead of the request library to download the ZIP file*
        - *for Jayvee, if you use the FilePicker, do not use a leading dot in file paths, see this bug: https://github.com/jvalue/jayvee/issues/381*
    - Pick out only stops (from stops.txt)
        - Only the columns stop_id, stop_name, stop_lat, stop_lon, zone_id with fitting data types
    - Filter data
        - Only keep stops from zone 2001
    - Validate data
        - stop_name can be any text but must maintain german umlauts
        - stop_lat/stop_lon must be a geographic coordinates between -90 and 90, including upper/lower bounds

        - Drop rows containing invalid data
    - Use fitting SQLite types (e.g., BIGINT, TEXT or FLOAT) for all columns
    - Write data into a SQLite database called “**gtfs.sqlite**”, in the table “**stops**”
- Group A, use Jayvee
    - Jayvee version 0.0.16
- Group B, use Python
    - Python 3.11
    - Allowed libraries: https://github.com/jvalue/amse-exercise-feedback/blob/main/requirements.txt
- Submit as **/exercises/exercise5.jv** or **/exercises/exercise5.py** depending on your language
- **Optional, after exercise**, give us feedback by participating in the survey:
    - English https://oss.cs.fau.de/surveys/index.php/284175?lang=en
    - German https://oss.cs.fau.de/surveys/index.php/284175?lang=de