

Applied Data Science in Fintech

{ 'Summer School 2025': Augmented Analytics workshop }



mario.gellrich@zhaw.ch

Dr. Mario Gellrich

Program for today

Program: Agent-Based Credit Risk Modeling Workshop

- Introduction, organization of groups, installation of software, brainstorming
- Introduction to augmented analytics with Python
- Working in groups on the tasks; expert sessions
- Presentation of results

*breaks are included as needed

Workshop procedure and philosophy



Image Credit: <https://revistaempresarial.com>

- No continuous presentations by the lecturer
- Group work (2 students per group)
- The groups are mixed (BSc/MSc OR beginners/professionals)
- Each "expert group" is responsible for one "topic"
- Methods are explained in detail in "technical sessions"
- Students ask the members of the "expert groups" for help
- The more experienced students help the others
- At the end of the day, the groups present their results
- The presentations are graded pass/fail.

Prerequisites

Software:

- GitHub Codespaces (<https://github.com/mario-gellrich-zhaw/summerschool>)
 - Python 3.11
 - Jupyter Notebook
 - Visual Studio Code

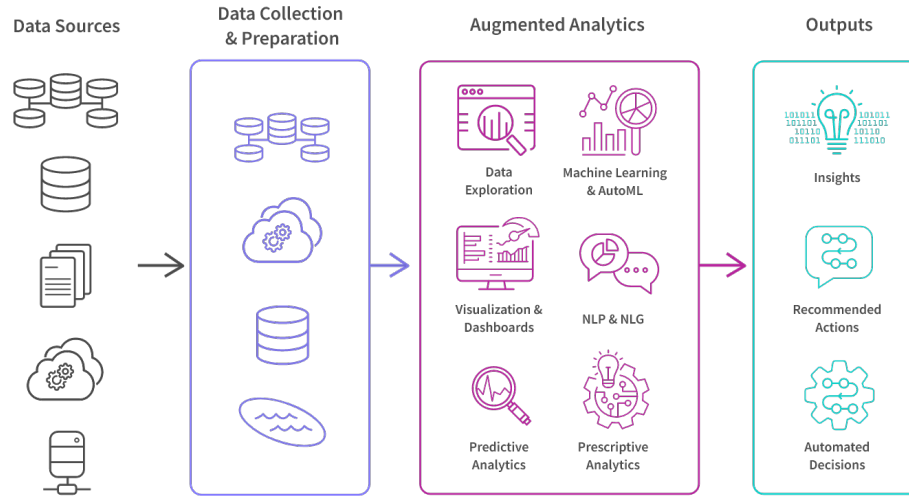
Material for exercises

- Jupyter Notebooks (will be used to guide the exercises)

Mandatory reading

- What is Augmented Analytics in Data Science (2025). <https://www.geeksforgeeks.org/what-is-augmented-analytics-in-data-science>

Definition



Augmented Analytics refers to the use of machine learning (ML), natural language processing (NLP) and artificial intelligence (AI) to enhance the process of analyzing data, generating insights, and enabling decision-making. It automates different aspects of the data analysis lifecycle making analytics more accessible to users who may not have advanced technical skills.

Image Credit: <https://www.qlik.com/us/augmented-analytics>

Keyword pinboard (basis to define group tasks)

(Topic 1) The Data Analytics Process

-
-
-
-
-
-
-

(Topic 2) Using LLMs for Data Analysis

-
-
-
-
-
-
-

(Topic 3) Running LLMs locally with Ollama

-
-
-
-
-
-
-
-

(Topic 4) Using GPT Models via the OpenAI API

-
-
-
-
-
-
-
-

Keyword pinboard (basis to define group tasks)

(Topic 5) The LangChain Framework

-
-
-
-
-
-
-
-

(Topic 6) Single-Agent versus Multi-Agent Architecture

-
-
-
-
-
-
-
-

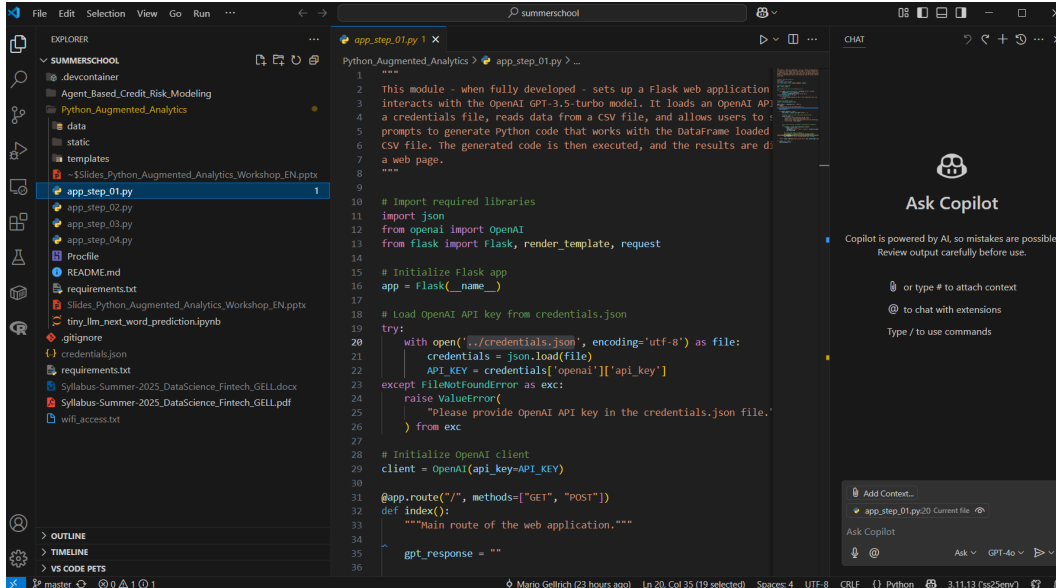
(Topic 7) Building a Web Application with Python and Flask

-
-
-
-
-
-
-
-

(Topic 8) Deployment of a Web Application with Koyeb

-
-
-
-
-
-
-
-

Keyword pinboard (basis to define group tasks)



The screenshot shows a Visual Studio Code editor window. The Explorer pane on the left displays a project structure for 'SUMMERSCHOOL' with folders like 'data', 'static', and 'templates', and files like 'app_step_01.py' through 'app_step_04.py'. The main editor area shows the code for 'app_step_01.py', which is a Flask web application that interacts with the OpenAI GPT-3.5-turbo model. The code includes imports for 'json', 'OpenAI', 'Flask', 'render_template', and 'request'. It defines a Flask app, loads API keys from a 'credentials.json' file, initializes an OpenAI client, and sets up a route for a chat interface. The status bar at the bottom indicates the file is at line 20, column 35, with 19 lines selected.

```
1 """  
2 This module - when fully developed - sets up a Flask web application  
3 interacts with the OpenAI GPT-3.5-turbo model. It loads an OpenAI API  
4 a credentials file, reads data from a csv file, and allows users to  
5 prompts to generate Python code that works with the DataFrame loaded  
6 csv file. The generated code is then executed, and the results are d  
7 a web page.  
8 """  
9  
10 # Import required libraries  
11 import json  
12 from openai import OpenAI  
13 from flask import Flask, render_template, request  
14  
15 # Initialize Flask app  
16 app = Flask(__name__)  
17  
18 # Load OpenAI API key from credentials.json  
19 try:  
20     with open('../credentials.json', encoding='utf-8') as file:  
21         credentials = json.load(file)  
22         API_KEY = credentials['openai']['api_key']  
23 except FileNotFoundError as exc:  
24     raise ValueError(  
25         "Please provide OpenAI API key in the credentials.json file."  
26     ) from exc  
27  
28 # Initialize OpenAI client  
29 client = OpenAI(api_key=API_KEY)  
30  
31 @app.route("/", methods=["GET", "POST"])  
32 def index():  
33     """Main route of the web application."""  
34  
35     gpt_response = ""  
36
```

The Chat Copilot interface is open on the right, showing the 'Ask Copilot' prompt and a list of context items including 'app_step_01.py:20 Current file'.

To solve the tasks, you are encouraged to use ChatGPT, GitHub Copilot or similar software. This includes explanations and Python programming. It is expected that the explanations of LLM-based responds are cross-checked with independent sources, and that the Python code is tested.

Minimal structure of the final presentation (Jupyter Notebook / HTML)

The presentation (provided by every group at the end of the day) must have the following structure.

1. Introduction

- 1.1 Background
- 1.2 Problem
- 1.3 Objectives
- 1.4 Research Question

2. Materials and Methods

- 2.1 ...
- 2.2 ...
- 2.3 ...
- 2.4 ...

3. Results & Discussion

4. Conclusions