Introduction to Data Science — Final Project Phase 2

Instructors: **Dr. Bahrak, Dr. Yaghoobzadeh** TAs: محمد امانلو، محمد امین یوسفی، محمدرضا محمدهاشمی، حمید سالمی، متین بذرافشان، امیرمهدی فرزانه

Deadline: 9th Ordibehesht

Introduction

In this second phase of the Data Science project, your team will further enhance your practical data science skills by implementing core professional practices related to databases, AI pipelines, CI/CD (Continuous Integration and Continuous Delivery), and optionally, MLOps practices. This phase will help you structure your data processes, automate workflows, and familiarize you with industry-standard tools and methodologies.
The phase is divided into clear, well-defined tasks. You are expected to perform these tasks on your personal computer systems.
Please follow each section step-by-step carefully, documenting each step clearly and thoroughly, as this documentation is crucial for your assessment.

Section 1: Database Implementation and Data Querying

In this step, your team must transfer your previously cleaned dataset into a structured database system on your local computer. You will be working with a database such as SQLite, PostgreSQL, or MySQL.

Tasks to perform:

1. Choose a suitable database based on your dataset requirements (SQLite recommended for simplicity, PostgreSQL or MySQL if you prefer a more robust environment).
2. Database schema design:
    - Design a clear, logical schema (tables, fields, primary/foreign keys) suited to your dataset.
    - You should present your schema clearly, explaining each table, column, and relationships between tables.
3. Import your cleaned dataset into the database.
    - Use Python scripts (e.g., using Pandas and SQLAlchemy) to automate the data import process.

4. Database Queries and Explorations:
   Execute and document at least 5 meaningful SQL queries on your database, including:
   - Selection queries (`SELECT`) to explore data.
   - Queries demonstrating filtering (e.g., WHERE clauses), sorting, and aggregation functions.
   - Joins between multiple tables (if applicable).
   - Queries highlighting relationships and insights within your dataset.
   - Provide screenshots or clearly formatted query results in your report.

Section 2: Advanced Feature Engineering, Data Preprocessing, and Preparation for Modeling

In this section, your team must thoroughly prepare your dataset for the final modeling phase. You will conduct detailed feature engineering and complete professional-level preprocessing, carefully structured to match the standardized project folder format introduced previously. You must prepare your processed dataset based on insights from your exploratory analysis and EDA from Phase 1 (e.g., using Power BI visualizations). This dataset must be ready for direct usage in modeling tasks (e.g., classification, regression, clustering, recommendation system, NLP tasks, or any chosen task).

Detailed Step-by-step tasks:

1. **Review Initial Insights (EDA)**
   - Revisit and document key insights gained from your Phase 1 visualizations and exploratory data analysis.
   - Clearly define the features you plan to engineer based on these insights.
2. **Perform Advanced Feature Engineering**
   Implement sophisticated feature engineering methods relevant to your data, such as:
   - Creating new meaningful features (e.g., ratios, interaction terms).
   - Text vectorization (TF-IDF, embeddings) if textual data is used.
   - Encoding categorical variables appropriately (one-hot encoding, ordinal encoding).
   - Handling time series (if applicable): extracting date/time-related features, lag features, rolling statistics, etc.
   - Image data: Basic preprocessing, resizing, normalization, and extraction of visual features if applicable.

   Clearly document the rationale behind each engineered feature.

3. **Comprehensive Data Preprocessing**

- Handle missing data professionally (imputation or removal clearly justified).
- Normalize or standardize numeric features appropriately.
- Remove irrelevant, noisy, or highly correlated features based on statistical reasoning.

Section 3: Creating an AI Pipeline (Data Science Pipeline)

In this section, your goal is to create an automated, structured pipeline that can reliably prepare and process your data for modeling purposes. The pipeline must clearly demonstrate structured workflow practices and be implemented through modular scripts.
Tasks to perform:

1. **Organize your project directory clearly**:
   Your pipeline should follow a clear project structure. Below is an example you can follow on your local computer. Project Folder Structure (if you are using SQLite):

```
Project_P2_[Std_numbers]/
|
├── database/
|       └── dataset.db (your database file)
|
├── scripts/
|       ├── import_to_db.py
|       ├── database_connection.py
|       ├── load_data.py
|       ├── preprocess.py
|       └── feature_engineering.py
|
├── pipeline.py
├── requirements.txt
└── README.md (Explain how to run your pipeline step by step)
```

Detailed steps:

- **`scripts/database_connection.py`**
  Create a Python script for connecting to your database easily from other scripts.
- **`scripts/load_data.py`**
  Create a Python script that queries your database and loads data into Pandas DataFrames.

- **`scripts/preprocess.py`**
  Create a script for data preprocessing: handle missing data, clean invalid values, encode categorical features, normalize numeric data, etc.
- **`scripts/feature_engineering.py`**
  Write a script that performs meaningful feature engineering (create new columns, aggregations, text embeddings, etc.) and saves the processed data as CSV files or Pickle files.

2. **`pipeline.py`** **(main execution script)**
   Write a main script that sequentially calls the above scripts to form a complete pipeline:

```
import subprocess

subprocess.run(["python", "scripts/load_data.py"])
subprocess.run(["python", "scripts/preprocess.py"])
subprocess.run(["python", "scripts/feature_engineering.py"])
```

- Your preprocess.py and feature_engineering.py scripts should:
  - Load data directly from the database using the `database_connection.py` script.
  - Clearly separate preprocessing and feature engineering into distinct steps and scripts.
- Clearly specify dependencies in `requirements.txt`.
- Clearly document each step of running your pipeline in your **README.md**.

Section 4: CI/CD Implementation (Basic Continuous Integration/ Delivery)

Continuous Integration/Delivery refers to automating your workflows to ensure consistency, reduce errors, and streamline development. In this project, you will experience basic CI/CD using GitHub Actions.

Tasks to perform:

- Set up a GitHub Repository for your project (if not already done).
- Use GitHub Actions to create a simple CI/CD pipeline that automatically checks your code whenever you commit or push changes.
- Your GitHub Actions pipeline should:
  - Set up Python automatically upon each push/pull request.

    - Install all required dependencies using your `requirements.txt`.
    - Run your main pipeline script automatically.
    - Example GitHub Actions workflow (`.github/workflows/pipeline.yml`)

```yaml
name: Data Pipeline CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  pipeline-check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v3
        with:
          python-version: '3.12'
      - run: pip install -r requirements.txt
      - run: python pipeline.py
```

- Submit screenshots clearly showing the successful pipeline execution from GitHub Actions.

Optional Section (Bonus 10%): MLOps (Docker, Kubernetes or MLflow usage)

Advanced (optional) students can also explore introductory MLOps practices using Docker and containerization. Kubernetes is optional and not mandatory:

Detailed optional steps:

- **Docker Containerization**
    - Install Docker Desktop on your personal computer.
    - Create a `Dockerfile` to containerize your pipeline scripts and dependencies.
    - Build and run a Docker container that executes your `pipeline.py`.

    - Include clear screenshots and the Dockerfile.
- **Kubernetes**
    - install a simple Kubernetes distribution such as Minikube on your personal computer.
    - Run your Docker container inside a local Kubernetes cluster using Minikube.
    - Submit screenshots clearly demonstrating your Kubernetes execution.

Completing Docker will grant up to **10% bonus points** for this phase.

Notes

- Upload your work in this format on the website: DS_Project_P2_[Std numbers].zip. If the project is done in a group, include all of the group members' student numbers in the name.
- On the final presentation day, the complexity and sophistication of your overall project (including dataset choice, pipeline complexity, advanced feature engineering, preprocessing methods, use of advanced tools, and modeling approaches) will be evaluated by the jury panel. Based on this evaluation, your team can earn up to **an additional 10% bonus** on your total final project grade.
- Your submitted zip file should clearly include:
    - Documentation of your database schema.
    - Screenshots/outputs of SQL queries.
    - Clearly structured AI pipeline scripts (`pipeline.py`, scripts, README.md).
    - GitHub Actions successful pipeline screenshot.
    - Optional Dockerfile, Docker run screenshots, Kubernetes screenshots (if attempted).
- If the project is done in a group, only one member must upload the work.
- This phase will not be accepted after its deadline i.e. there will be no late and grace policy!

Good luck!