Data Science Toolkit Project

- Arrange yourself preferably in teams of 4-5. If needed, teams of 2 or 6 may be allowed with an exception.
- Deadline: 3 weeks
-

Project Requirements:

1. Get any dataset from Kaggle or the internet (Tabular, Not Tabular, Images, Audio, or even NLP and LLMs)
2. Create a GitHub repository and add the whole team as collaborators. Different features of the app will be as separate branches that will be merged to main via Pull Requests
3. Use a package manager to install and manage package dependencies (uv or any, not just pip), and have files indicating what packages need to be downloaded.
4. Write code for data loading, validation, preprocessing, EDA, model training, and evaluation as functions or classes in separate modules outside the jupyter notebook. Use any libraries you want e.g. (pandas, fireducks, polars, spark, pytorch, tensorflow, sklearn, any gradient boosted tree library, openai, LangChain, ...)
    - Bonus: Use MLFlow, Weights & Biases, or any cloud experiment tracking offering to log any model training experiments including hyperparameters, metrics, and artifacts.
5. Save the trained model(s) to a suitable code accessible cloud storage solution.
6. Create configuration files to store any default variables in your code outside the code.
7. Create a .env file if you have any secrets or sensitive variables in your code.
8. Write inference code, using FastAPI for inference
    - Bonus: add detailed logging to the inference code.
    - Bonus: Include basic key authentication using fastapi.Security
    - Bonus: extract the OpenAPI.json file API specification for your app.

9. Place the inference code and FastAPI server in a docker image to save on Docker Hub or any Cloud Image Registry
10. Deploy your containerized app to a cloud Virtual Machine server, or as a container on any cloud PaaS (AWS ECS, Google Cloud Run, Azure Container Service)
11. Bonus:
    - create a front end using Streamlit or plotly dash for inference.
    - Make sure any functions you create are documented using docstrings, and use type annotations (and elements from the typing module) to annotate all function parameters and return types.
    - Add lint and type validation tests using pylint, flake8, ruff, or mypy. You can also use “black” to format your code.

Excel Project:

Create a very simple dashboard using functions or pivot tables from your training data (if tabular), model metrics, training experiment data, or whatever data.

Bonus: use power query for data loading

Bonus: If you choose tabular data, call your deployed model from excel on some of the test data.

Discussion will be a large meeting with the whole track, or team by team (you choose)

In the discussion, everyone will be asked how they contributed and will be asked separately about some technical details.