Take-home Exercise - Mid-Senior ML Engineer

Hello and thanks so much for taking the time to do the take-home exercise for the Mid-Senior ML Engineer role.

Why are we asking you to do this?

We want to talk to you about code - how you write it, how you think about it, how you turn ideas into concrete programs, what makes good code. We want to do this in a way that gives you the chance to show your skills at their finest, in a context that we both understand. Ideally, we're after a snapshot of what your code would look like in day-to-day engineering work.

What are we asking you to do?

You're working in the data team at Ferocia. Your task is to build a tiny but production-ready ML service that predicts whether a customer will subscribe to a term deposit using the provided dataset. There are two parts to this exercise. **Please spend no more than three hours completing this exercise**, and particularly don't spend too much time optimising the model's performance. As a guideline, you should spend 40% of your time on part A and 60% on part B. Please aim to return the exercise within one week - though if you need more time, please let us know.

Note: With limited time, we understand that you are very unlikely to make the perfect solution, if there are places where you want to make concessions to limit the scope, please highlight how you would do it if you had more time. For example, "I'm foregoing comprehensive unit-tests, but each preprocessing function should be unit-tested in the real-world, I wrote one unit-test as an example of what I would do"

Our stance on AI tools

We invite you to use AI tools if you'd like to. We don't expect you to use them, nor do we forbid you from using them, it's up to you. Whether it's GitHub Copilot,

ChatGPT, Claude, or Cursor, if you do decide to use them then you need to demonstrate *how* you are using them.

We are not just assessing the final code; we are assessing your process.

Guidelines:

1. **Git History:** We want to see the evolution of your solution. Please **commit frequently** (e.g., at each logical step) rather than squashing everything into one commit. A granular history helps us distinguish between code you wrote, code you refactored, and code the AI generated. *If we can't see your decision making process and how you iterate on your code, then we will not be able to evaluate your test.*
2. **Transparency:** If you use conversational AI (e.g., ChatGPT, Claude), please save any chat transcripts and include it in your repo. If you use inline tools (e.g., Copilot), a quick mention in your commit messages or PR description of how you used it is great too.

Outputs & Submission

You must submit your solution via a **GitHub Repository**. If you need it to be private, let us know and we will provide handles for you to add as collaborators.

Your repository must contain:

1. **Source Code:** The model training and model hosting code.
2. `README.md`**:** This should include setup instructions, but also a specific section titled **"Tool Stack."** Please list the AI tools/IDE extensions you used (We are genuinely curious about what tools everyone is using)
3. `AI_TRANSCRIPTS`**:** A folder containing export files (markdown/txt/pdf/etc) of your sessions with LLMs, if applicable.

Part A: Model training

Let's first dive into the data, we would first like you to prepare the dataset for training. This code should be reusable, reproducible, and easily portable as the last step of the model will be to package it for serving.

Next step is to train and evaluate an appropriate model, we aren't looking for a Kaggle winner but we're looking for you to identify what sort of ML problem this is and utilise the appropriate techniques and tools to produce an MVP model. Due to the experimental nature of modelling, please don't spend too much time perfecting the model, but please note down any and all commentary of what you would do to improve the model in future iterations.

Lastly, once the model is trained and evaluated, please package the model up locally for serving which will be the next part of your test. Choose the appropriate tool to package the artefacts necessary to productionise the model.

Part B: Model hosting

The next part will be writing a light-weight API to host the model which was produced by the previous section. We would like you to implement a `/predict` endpoint which will take in raw input from a consumer, please implement the necessary code which will produce a response to the consumer.

What we will be looking at?

Assessed aspects:

1. Your understanding of the necessary training and evaluation techniques, and we will dive deeper as to the rationale behind each step and how you understand them in the following round.
2. How you structure your repository and code from a software engineering perspective.
3. How you think about the productionisation of models, this is also something we will spend quite a bit of time talking about in the following round.
4. How you iterate, refactor, and correct course in your git logs and transcripts, whether manually or via AI.

Out of scope:

1. Your model performance, so please don't spend too long optimising this.
2. Cloud hosting, everything should be executed locally.
3. API contracts, versioning working, etc

To submit

You can submit your work either by committing it to a repository and sending us the link, or you can upload a zip file to Greenhouse using the link in this email.

Questions?

Please reach out if you have any questions, or if you need more time.