One specific task in which I was involved with LLM-based coding was the creation of an interactive chatbot for my thesis work at Data Science Retreat. Myself and two others aimed to develop a chatbot to assist non-EU medical professionals in obtaining licenses to practice medicine in Germany and, consequently, the EU. We named this model MAG (Medical Advice Generator, https://github.com/ARosenswie/DSR_Final_Project).

The initial step was to gather a comprehensive database of medical question and answer (Q&A) pairings. We then gathered 570K medical Q&A pairings taken from: PubMedQA (Jin et al., 2019), MedQA (Jin et al., 2020), MMLU (Hendrycks et al., 2021), and MedMCQA (Pal et al., 2022). We preprocessed these questions to eliminate potential issues with answers, involving the vetting of multiple-choice answers, resulting in the removal of incorrect answers, leaving only the correct answers. The Q&A pairings were then organized into a JSON dictionary.

After obtaining the Q&As, the next step was to select a model suitable for our project. We knew we wanted to use an LLM and explored various publicly available models, primarily on Huggingface. Initially considering Google Flan T5, we realized we lacked the necessary resources to compile a working model, specifically running out of memory on a 76 Gigabyte RAM hard drive via Google Colab. Faced with the impending thesis deadline, we scoured the internet for potential models that could be employed. Eventually, we came across OpenAI's original GPT models and explored their potential usage. After evaluating potential options, we found that text-davinci-003 had the most suitable parameters for our task, leading us to decide to use it.

With a model now chosen, we had to design an interface that a user could easily access and utilize. Our idea was to use a Streamlit application where a person could ask a question, and the answer would be displayed. We created a prompt-helper, which was then passed into the LLMPredictor. We set the inference temperature to 0.1, aiming for the most concise and precise information from MAG. The results were saved into a vector index file.

The design of the Streamlit application followed the pattern of other documented models from literature. The file with the vector index was passed into GPTSimpleVectorIndex, where the query was posed, and the corresponding answer was displayed. We FINALLY had a working model. Recognizing the need to validate MAG's responses, we gathered 36 questions from five medical textbooks, with known and given answers. We used these 36 answers as a baseline and asked MAG the same questions, saving its responses. Our supervisor, a pharmacist, and our

classmate's wife, a psychiatric doctor, were asked to assess MAG's answers, ranking them based on "Quality of Information" (very good, good, acceptable, poor, and very poor) and Classification (green, yellow, and red). MAG performed well in their evaluation, with 24 green responses and 24 responses rated as good or very good. We then compared MAG's answers against those provided by Chat-GPT for the same questions, using eight metrics from Huggingface: Bleu, Mauve, Meteor, Rouge-1, Rouge-2, Rouge-L, Rouge-L_{sum}, and F1-$score_{avg}$. MAG outperformed ChatGPT in each metric, impressing our mentor and the course supervisor, challenging the public opinion of ChatGPT's omniscience. MAG is now publicly available as a GitHub repository.

Throughout this experience with MAG, I gained valuable insights into how these models, methods, and production processes come to be. From these, I would tell a fellow data scientist like myself these three tips:

1. Be open to trying new models,
2. Use a vector store to store embeddings,
3. Be prepared to encounter many technical problems and associated costs.

The first point, illustrated by our experience, highlights that the originally desired model (Flan T5) could not be used, emphasizing that availability on Huggingface does not guarantee usability. The second point emphasizes the use of a vector store like Pinecone. The file with the embeddings was large and could not be uploaded to GitHub or by git-lfs, requiring us to upload the indexed file as a "Release" on GitHub to bypass this issue. A more viable solution would be to use a vector store/database such as Pinecone. The third point underscores the awareness that this process incurs not only time but also money. Initial expectations of a free process were challenged as each model run incurred costs, not just for an OpenAI key but also for cloud platforms like Google Colab due to GPU usage. Additionally, models may have a limited lifespan, as demonstrated by text-davinci-003 and other OpenAI models being "turned off" on January 4th.