

# Independent Study Final Report

Advisor: Dr. Mine Çetinkaya-Rundel

Avery Hodges

## Section 1: Literature Review

### Review Content

#### Source 1: “Generative AI for Data Science 101: Coding Without Learning To Code.”

Bien and Mukherjee (2025) taught students how to write Chat prompts into GitHub Copilot that turns it into R code that can be run during a Fall 2023 one-semester course at USC, “Data Science for Business”. Unlike traditional point and click software, which function similarly to a tour guide in a foreign country, and starting from scratch, similar to learning the language and having complete immersion in a foreign country, the GitHub Copilot tool functioned as a happy medium with LLMs, like a translator in your ear. As Github Copilot was embedded in the coding environment, it eliminated the possibility of a too-wide searching net cast that would occur with using LLMs like ChatGPT.

Throughout the semester-long course, the research team encountered some challenges in the learning process of their students. First, Copilot’s outputs for students were seemingly semi-random, meaning that some prompts that worked earlier in a session would no longer work or output different responses. Copilot also had a tendency to fabricate variable names if they were not given, and make incorrect conclusions in response to the prompt and data given to it. In terms of formatting, Copilot frequently switched between tidyverse and base R coding style, which could confuse students. Lastly, there was little transparency about how Copilot arrived at the coding answers it did, leaving students without the necessary context of why their answers make sense. Overall, while the tool was able to foster students learning through guided feedback and tailored content, there was a concern that students wouldn’t be able to tell if their Copilot code was correct or not, leaving educators to teach students how and when to check their output.

### **Source 2: “Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study.”**

In Lyu et al. (2024) , a similar study was conducted in the computer science field, where a semester long study of 50 students were separated into an experimental group that had access to Code Tutor, an LLM powered assistant made by the research team, and a control group with no access to the tool. At the end of the semester, those in the experimental group had higher test and assignment scores than those in the control group, but as time increased, how much students agreed with the tool decreased and their preference for traditional human assistants went up. Despite these results, no other factors were controlled for in the study.

### **Source 3: “A New Era of Learning: Considerations for ChatGPT as a Tool to Enhance Statistics and Data Science Education.”**

Ellis and Slade (2023) focuses on the potential and cautions surrounding ChatGPT and its use in data science education, and only includes discussions about these topics, not a study of some kind. Despite the controversy about ChatGPT’s lack of “common sense knowledge or the ability to understand context beyond what it has learned from its training data”, the immense amount of information that ChatGPT trains on, such as books, articles, journals, websites, and other digital resources, LLMs like this can access a wide array of information that a user normally could not in the same time period. However, not all sources on the internet are created equal, and inaccuracies in these sources, especially in more complex topics, can reveal themselves in the LLM’s outputs. Despite this, some use cases in the data science educational field include generating lecture notes summary, practice quizzes and exam question, or code conversions between languages such as R and Python.

### **Source 4: “Empowering Education: ChatGPT’s Role in Teaching and Learning Statistics and Data Analytics.”**

Similar to Source 3, Pan and Gu (2023) did not use experimental methods, but rather includes a discussion about ChatGPT and its role in data science education. The article emphasized that since ChatGPT’s introduction to the market in November 2022, there has been a transitional emphasis on coding, shifting from an isolated job market to one that can be available to everyone. Sped-up by the COVID-19 pandemic, in which there was limited in-person and synchronous education time, virtual tools and their development were streamlined. Students no longer have to code from scratch, leading us to a future where “coding will be a new form of literacy, with the expectation that students in the 21st century will acquire this skill.” (page 43). In addition to this, the article explained how the GPT Builder tool developed by OpenAI can make personalized models based on the material given to it, a potential resource and framework that educators can take advantage of when structuring their own courses.

**Source 5: “The Use of Generative AI in Statistical Data Analysis and Its Impact on Teaching Statistics at Universities of Applied Sciences.”**

In another discussion and study on the future of education in data science and LLMs, Schwarz (2025) emphasizes the lack of experimentation in testing different types of LLM education platforms, claiming that there has only been theoretical discussions on the benefits and challenges of these systems. In their exploration, relatively simple statistical concepts were explained well by ChatGPT specifically, while the model struggles with explaining more complex topics. The researchers found that code written and produced in R and Python were better explained, as there is more digital material for the LLMs to train on.

To test the limitations of LLMs, the research team, using GPT Data Analyst - an OpenAI product that lets you upload data and evaluate it - generated 4 simple artificial data sets to evaluate how well the LLM could explain concepts to a user that had no statistical knowledge. Some of the data sets were generated in a way that certain statistical model assumptions were violated, and some had serious violations. At the end of the study, it was determined that ChatGPT Data Analyst points out that you need to check for application assumptions, and it is capable of doing that, but didn't carry out the step. A person with no statistical knowledge wouldn't catch this error and make incorrect conclusions. From these results, the research team suggested that ChatGPT Data Analyst should be used as an evaluation software that makes students' work easier instead of generating the code for them. Finally, the research team noted an ethical concern about data privacy, as data uploaded into the Data Analyst is no longer private, and data that contains sensitive information may not be able to be used in the tool.

**Source 6: “What Should Data Science Education Do with Large Language Models.”**

In another discussion on data science education and Large Language Models, Tu et al. (2024) dives into some theoretical use cases in the educational sphere. The researchers begin with a comparison to real-world jobs, claiming that the “evolution of roles is reminiscent of the transition from a software engineer to a product manager.” Instead of writing code, we now oversee it, emphasizing the need for a different perspective for educators to take as they approach LLMs. To prevent plagiarism and pure coding assignments that can be streamlined, assignments should have personalized reflections, critical thinking pieces, or unique approaches to solving a problem that AI can't easily replicate. Educators can take advantage of plagiarism detection tools, but these can have high false positive rates. The new strategy that teachers should take, according to the research team, is one that nurtures creativity and critical thinking, guided by an LLM that provides hints or feedback as students move through the learning and planning process.

## **Source 7: “Incorporating LLM-Based Interactive Learning Environments in CS Education: Learning Data Structures and Algorithms Using the Gurukul Platform”**

To end the literature review, this last article published by Virginia Tech University, Rachha (2024), introduces the Gurukul platform, a coding system developed by the research team that contains “Retrieval Augmented Generation and Guardrails” that prevents the model from providing explicit solutions to students. As LLMs can “hallucinate” or give inaccurate answers as fact, students can be led astray by the inaccuracies of an LLM, inhibiting the learning process. In addition, it can expose students to cheating and prevent students from the critical trial and error step of the learning process, damaging their knowledge down the line. To solve this, the research team implemented this tool that provides hands-on learning, but gives guided feedback and does not provide the correct answer immediately, allowing students to still learn as they progress.

## **Section 2: Project Overview & Progress**

### **Project Goal**

My independent study, LLMs for assessment feedback, sought to create a question and rubric bank that will serve as the basis for a tool that leverages LLMs to provide feedback for student work in intro data science courses. During this process, we planned on evaluating the performance of various LLMs and also build a tool for students to use within their IDE, specifically within R-studio.

### **Stage 1: Building the Question Bank**

Before the various LLMs could be tested, data to train on was needed for the LLM to provide feedback to. In the first stage of the semester, our team pulled lab assignment questions from the Sta199 Fall 2024 course and separated the 62 questions into modeling, wrangling, tidying, and visualizing questions. We created individual R markdown files for the questions, suggested answers, general rubrics (that did not specify exact numerical answers), and detailed rubrics that the model would have access to when providing feedback to the student. We then simulated sample answers to a question and separated them into good (correct answers), mediocre (mostly correct answers), and bad (serious errors) files.

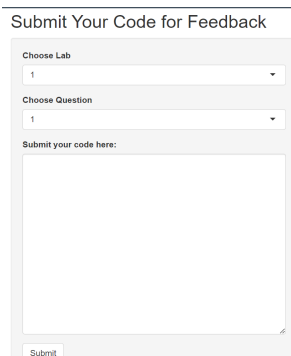
### **Stage 2: Feedback Format & Testing OIT Models**

After the questions were created, the feedback structure that the LLM would provide to students needed to be formatted in a way that wouldn’t give students the correct answer explicitly, but constructively guide them to the right answer in the context of the rubric items. To do this, I set up the general format that the LLM would follow to give students feedback in response to the code they submit for a specific lab assignment question, shown in the Feedback\_Format.txt file in this repository.

With the question bank and feedback format established, the various OIT models at our disposal could be tested on how well they copied the format created. In coordination with Mark McCahill, access to the OIT platform was given, and the feedback format was evaluated against the models available. From this evaluation, the ChatGPT-4o-mini model copied the feedback format instructions the best, in terms of not giving students the numerical correct answers, restating the correct rubric items, and determining whether the student's answer was correct. Other models in the OIT system could often perform a couple of these tasks well, but had major formatting issues, gave incorrect answers, or did not state the rubric items correctly or did so inconsistently. Lastly, with ChatGPT 4 models specifically Couch (n.d.) predicts that future continuations of these models will have better tailored coding abilities for real-world instruction. After this evaluation, ChatGPT-4o-mini was chosen as the best model to move forward with.

### Stage 3: Connecting Questions to the OIT LLMs

The final stage of the project in the semester was to connect the question bank data with the LLM from the OIT platform. To do this, I researched Wickham, Cheng, and Jacobs (2025) at the suggestion of Dr. Mine Çetinkaya-Rundel, an R package that provides the framework and functions to send queries to LLMs using API requests. After reviewing the documentation, I created a shiny gadget using Chang et al. (2024) that would be registered as a r-studio add-in that students could choose a specific lab assignment, a specific question, and paste their code into a text box to submit their code and narrative that would send a query to the OIT large language model, shown in gadget\_feedback.R. and in the image below:



Submit Your Code for Feedback

Choose Lab  
1

Choose Question  
1

Submit your code here:

Submit

### Learning Acquired

Before this semester, my knowledge of LLMs was lacking in many ways. As this independent study has progressed, I have learned how these models work, which are better at formatting feedback in specified ways, how the Wickham, Cheng, and Jacobs (2025) package can connect a query to an LLM model and send its output back to the user, how to work with API requests, and more. Rather than a surface knowledge of how LLMs can be used to help students, I got to start a back-end pipeline that will help students gain access to valuable feedback as they progress in their data science education.

### Section 3: Next Steps

Looking forward, there are some steps that the team was able to start during the semester, but need additional time and resources in order to move forward.

First, the process of connecting the OIT LLM with the private question bank repository in GitHub is in the process of completion. To start this process, certain variables had to be registered in the .Renvirom file, these variables are listed below, with each of these variables acquired through OIT's contact Mark McCahill (except GITHUB\_PAT) and listed in the ellmer\_test.qmd file in the question-bank repository owned by Dr. Mine Çetinkaya-Rundel:

- OPENAI\_API\_TYPE: the organization/provider of the model
- OPENAI\_API\_BASE: URL to the OIT modeling platform
- OPENAI\_API\_VERSION: version of the model
- OPENAI\_API\_KEY: API key to access OIT models
- GITHUB\_PAT: personal API key to access GitHub repositories, but needs further work/research

The file structure set up to access the various GitHub folders has been determined, and the functions in the ellmer\_test.qmd and gadget\_feedback.R files in the question-bank repository reflect this structure. For future testing, ensure that your files follow this relative structure (note that the names of specific files/folders will be different, but when structuring the system, the shiny gadget will accept this structure):

```
/LabAssignments
/Lab1
/Q1
  question.qmd
  rubric.qmd
  detailedrubric.qmd
  answer.qmd
/Q2
  question.qmd
  rubric.qmd
  detailedrubric.qmd
  answer.qmd
...
/Lab2
/Q1
  question.qmd
  rubric.qmd
  detailedrubric.qmd
  answer.qmd
...
...
```

Lastly, due to the permission errors with the private GitHub repository, question-bank, the function to create the lookup table, called `file_lookup.csv`, still needs to be created and the function needs to be modified. For a specific use case, a manually created lookup file has been created and placed in the `ellmer_test.qmd` file in the question-bank repository. Once this has been created correctly for all question folders in the question-bank repository, the shiny gadget in `gadget_feedback.R` needs to be updated so that the lookup table can be read in and the gadget can access it to then pull questions from the question-bank repository based on the specific lab assignment/lab question combination a student chooses.

### **Comparison: TA vs. LLM**

Once these next steps have been completed, a student should be able to use the shiny gadget to choose a specific lab assignment/question combination while they are working on that assignment in their IDE, and paste the code/narrative that they are working on into the gadget for evaluation. From there, the gadget should use the lookup table to pull the `rubric.qmd`, `detailedrubric.qmd`, `question.qmd`, and `answer.qmd` from the question-bank repository, the pasted code from the student, and send a request to evaluate the student's answer against the `rubric.qmd` given the other files and the specific format specified in `Feedback_Format.txt`.

Compared to a teaching assistant (TA) grading these assignments, in which the TA can look at the rubric items extensively at the cost of time and energy, we hope that students will be able to use this tool more incrementally: instead of checking their work at the end, this LLM tool would guide students in the right direction without giving them the right answer explicitly as they progress towards their final submission. In our evaluation of this LLM's performance, there are a few differences between a typical TA's grading and the LLM's evaluation:

Specifically for code style, the LLM would often penalize for this rubric item, even if there were no errors that a TA could see. Compared to a TA, the LLM was more consistent in providing feedback, as comments do not change from submission to submission and there are not multiple graders on one question. However, it should be emphasized that the LLM could not give personalized responses to every submission, like a TA could, as there is a specific feedback format based on the rubric items that the LLM was instructed to follow. Lastly, although the LLM may seem consistently correct to students, there are still rubric items in which the LLM gives wrong answers with confidence, introducing the potential to mislead students, whereas as a TA might make these mistakes, but can constantly recheck the rubric items and rely on their past knowledge to grade correctly.

## References

- Bien, Jacob, and Gourab Mukherjee. 2025. “Generative AI for Data Science 101: Coding Without Learning to Code.” *Journal of Statistics and Data Science Education* 33 (2): 129–42. <https://doi.org/10.1080/26939169.2024.2432397>.
- Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2024. “Shiny: Web Application Framework for r.” <https://CRAN.R-project.org/package=shiny>.
- Couch, Simon. n.d. “GPT-4.1 and the Future of Data Science Education.” <https://www.simonpcouch.com/blog/2025-04-15-gpt-4-1>.
- Ellis, Amanda R., and Emily Slade. 2023. “A New Era of Learning: Considerations for ChatGPT as a Tool to Enhance Statistics and Data Science Education.” *Journal of Statistics and Data Science Education* 31 (2): 128–33. <https://doi.org/10.1080/26939169.2023.2223609>.
- Lyu, Wenhan, Yimeng Wang, Tingting (Rachel) Chung, Yifan Sun, and Yixuan Zhang. 2024. “Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study.” *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, July, 63–74. <https://doi.org/10.1145/3657604.3662036>.
- Pan, Youqin, and Jian Gu. 2023. “Empowering Education: ChatGPT’s Role in Teaching and Learning Statistics and Data Analytics.” *International Journal of Technology in Teaching and Learning*. <https://doi.org/10.37120/ijttl.2023.19.1.02>.
- Rachha, Ashwin Kedari. 2024. “Incorporating LLM-Based Interactive Learning Environments in CS Education: Learning Data Structures and Algorithms Using the Gurukul Platform.” PhD thesis, Virginia Tech.
- Schwarz, Joachim. 2025. “The Use of Generative AI in Statistical Data Analysis and Its Impact on Teaching Statistics at Universities of Applied Sciences.” *Teaching Statistics* 47 (2): 118–28. <https://doi.org/10.1111/test.12398>.
- Tu, Xinming, James Zou, Weijie Su, and Linjun Zhang. 2024. “What Should Data Science Education Do With Large Language Models?” *Harvard Data Science Review* 6 (1). <https://doi.org/10.1162/99608f92.bff007ab>.
- Wickham, Hadley, Joe Cheng, and Aaron Jacobs. 2025. “Ellmer: Chat with Large Language Models.” <https://CRAN.R-project.org/package=ellmer>.