





# Bridging the Programming Skill Gap with ChatGPT: A Machine Learning Project with Business Students

Michael Reiche<sup>1</sup> and Jochen L. Leidner<sup>1,2</sup>

<sup>1</sup> Coburg University of Applied Sciences, Friedrich Streib-Strasse 2, 96450 Coburg, Germany

`michael.reiche@hs-coburg.de`

<sup>2</sup> University of Sheffield, Regents Court, 211 Portobello, Sheffield S1 4DP, UK

**Abstract.** Foundational language models, i.e. large, pre-trained neural transformer models like Google BERT and OpenAI ChatGPT, GPT-3 or GPT-4 have created considerable general media attention. Microsoft’s github.com service has also integrated a foundational model (CodePilot) to make programmers more productive. Some people have gone so far and heralded the end of the programming profession, an unsubstantiated claim.

We investigate the research question to what extent individuals without the necessary technical background can still use such systems to achieve a set task. Our single case study based preliminary evidence suggests that using such systems may lead to a good task completion rate, but without deepening the understanding much on the way.

**Keywords:** Computer-Supported Instruction · AI for Teaching · Pretrained Foundational Models · Artificial Intelligence in Education · Classroom Case Study

## 1 Introduction

Increasingly, employees with a business background are also expected to have a high-level understanding of machine learning concepts. Consequently, the textbook *Introduction to Statistical Learning* [2, 5] is used to train MBA students at Stanford University. Following this trend, the course “Team Project Artificial Intelligence” was offered as an elective from the area of projects in the winter semester 2022/2023 in the Master’s degree program in business studies at the Coburg University of Applied Sciences and Arts.

We borrowed a task from Kaggle.com, namely the task of sentiment analysis of online-feedback about treatment experiences with medics.

Out of 19 participants across all projects of the semester and study program, 6 participants took part in the project. The goal of the project was to impart knowledge for the focus “IT Management”, which enables the students to participate in machine learning projects in their later professional life or to be able to comprehend them. In particular, knowledge from the following areas was imparted.

- Technical competences:
  - Recognition of machine learning use cases.
  - Implementation of machine learning use cases, especially in the phases of business understanding, data understanding, data preparation, modeling and evaluation.
- Methodical competences:
  - Using ChatGPT for programming.
  - Following a machine learning methodology.
- Social competences:
  - Presenting technical and business results to a group.
  - Discussing in a group.
  - Analyzing, evaluating, and refining one’s approach to learning and problem solving.

The project has had another intended impact in addition to building students’ competencies. Namely, the generation of data in a controlled experiment to compare two methods with an intervention group and a control group. Therefore, during the planning of the project, dependent (iterations, experimentation, planning, communication) and independent (business case, input data, algorithm, upskilling, domain knowledge, infrastructure) variables as well as behavioral rules were defined to increase the reliability, validity and credibility of the results.

The teaching of the competencies and the performance of the activities required for this purpose were achieved by means of various formats, each of which was selected in such a way that the previously defined dependent and independent variables were subject to as few disturbing influences as possible. An inquiry at the beginning of the project about the theoretical and practical knowledge showed that the participants (as well as the students of the master’s program in business administration in general) did not have adequate machine learning or programming knowledge that would have enabled the independent and successful implementation of a machine learning project.

To bridge the gap between the programming skills needed for the project and those brought by the participants, the use of ChatGPT has been suggested for activities with programming content, especially for generating and concatenating Python code fragments and fixing bugs in the program code.

In this paper, we investigate the research question “Can students with no knowledge of computer science and no knowledge of a programming language use ChatGPT to generate all the Python code for a machine learning project?”.

## 2 Related Work

After foundational language models based on the transformer architecture by Google [6] with BERT (Bidirectional Encoder Representations from Transformers) [1] received great attention in the scientific community, more and more so-called large language models (LLM) have emerged [7].

Vaithilingam et al. conducted an experiment with 24 participants, 23 of whom had more than two years of programming experience. They investigated, among other things, whether the LLM Githubs Copilot, which is based on OpenAI's Codex, offers more support to experienced programmers than the code completion tool Intellisense. No significant difference was found. However, participants indicated that Copilot gave them a better start in the programming task. Even in the case of incorrect code, at least a helpful direction was given, which could offer help to novice programmers in particular. 12 of the participants found it difficult to understand or change the code generated by Copilot, which was also due to the fact that the generated code is not commented [5]. Kasneci et al. have critically examined the opportunities and risks of LLM in education. They concluded that LLM offer many opportunities to enhance learners' learning experience, but generated information can also have a negative impact on learners' critical thinking and problem-solving skills. To counteract this, LLM should support learning and not replace human authorities [3].

Controlled experiments in software engineering are a common means of establishing cause-and-effect relationships. A survey of 103 articles found that 87 % of the subjects were students [4].

According to our research, no papers have been published on the use of LLMs or ChatGPT in particular that investigated how well people without a technical background can programme.

### 3 Method

To answer the research question, various methods were used before, during and after the project.

At the beginning of the project, a survey was conducted to determine the practical experience and theoretical knowledge of each participant. For example, the following closed-ended questions were asked with a free-text option that allowed participants to choose between “yes” and “no” answers, and to be able to explain their answer with an additional answer choice:

- Do you already have practical experience in programming?
- Do you already have theoretical knowledge in Machine Learning Engineering?
- Do you already have practical experience in Machine Learning Engineering?

If one of the questions was answered with “Yes”, then the content of the associated free-text option was analyzed and a point value from 0 (no knowledge or experience) to 3 (more than two relevant lectures attended or everyday professional life) was assigned depending on the proficiency.

Two groups (team blue and team red) were formed, which had as equally distributed experience and knowledge as possible in the three questions on programming and machine learning engineering (team blue 3 points and team red 3 points) and in other questions (team blue 25 points and team red 19 points). The two groups differed in that they each had to use one of two machine learning methodologies to carry out the project, which led to different ways of working.

In order to compensate for the lack of technical knowledge and experience, a joint coaching session was held for both groups in the first third of the project before the use of a programming environment. This provided the theoretical foundations of machine learning and a guideline for using ChatGPT to generate the Python code for the project by means of a live demonstration with ChatGPT and Google's development environment Colab.

During the live demonstration, ChatGPT and Google Colab were used in interaction. In ChatGPT, it was shown where prompts are entered, responses generated and regenerated. In Google Colab, it was shown which elements the interface has and in particular how code blocks are inserted and executed. As an example of the interaction between the two web-based interfaces, simple mathematical calculations and a simple data pipeline were generated with ChatGPT and inserted as code blocks in Google Colab. To illustrate the data pipeline, the framework Pandas was imported, a data frame was filled with a data set (CSV), the content of the data frame was sorted and the content of the data frame was visualised. The individual work steps were carried out by the lecturer and by volunteer participants.

The project was implemented in four teamwork sessions per group, three of which included programming. The individual sessions lasted between 83 min and 229 min. However, the average was 181 minutes.

During the project, data such as ChatGPT histories, schedules, project documentation, Python files, presentations and tables with over 250 team activities were generated by the group participants and the lecturer/experiment leader.

The project switched from Google Colab to Jupyter notebooks and JetBrains's PyCharm. The groups were not allowed to use any other tools, information or communication tools than those provided. The project was completed with presentations and final reports by the groups (Fig. 1).

## 4 Evaluation

The teams interacted with ChatGPT (version 3.5 with German conversations) in different ways. In team blue, one of the three participants was responsible for interacting with ChatGPT and programming in the development environment. In team red, all three participants worked together with ChatGPT most of the time and wrote the programming code for the project. How the prompts were formulated was at the discretion of the participants. Code fragments were asked for with general prompts without project-specific specifications and with specific prompts with detailed specifications that enabled seamless integration into the already existing code.

Although ChatGPT always provided explanations in addition to code, these were rarely read through. There was often a lack of understanding and time to follow the explanations. Instead, the code generated with ChatGPT, including explanatory comments, was copied completely and with hardly any adjustments into a programming environment and executed. In case of errors and unexpected returns, ChatGPT was mostly consulted and in rare cases the lecturer. In some

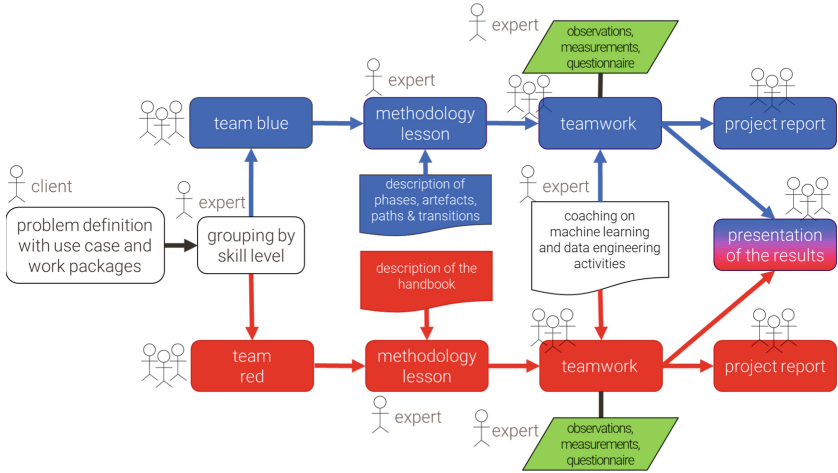


Fig. 1. Controlled Study Design

cases, functions from libraries were mentioned that showed incompatibilities. This is due to the fact that the ChatGPT version used was only trained with data up to 2021.

From a technical point of view, it was clear to the participants what the code fragments should output. Therefore, prompts could be given correctly and the results of the executed code could be interpreted in the development environment.

Interaction behaviour with ChatGPT of the team red: The team generated the programme code for the project with 97 prompts. With specifications such as “In future, please take our data name df into account in the code. Please reissue the code”, the group influenced the subsequent response behaviour of ChatGPT so that the generated responses fitted better into the already generated and adopted Python code, which reduced the required understanding about the programming. In one case, the team formulated a prompt without a question. The response was “I’m not sure what exactly you want to customise. If you mean a specific customisation in the DataFrame, please give me more information about it.” When the code inserted and executed in the development environment produced unexpected or erroneous results, ChatGPT was interacted with again, for example with the following prompt “what does this mean? Empty DataFrame Columns: [index, rating, comment] Index: []”. ChatGPT was also questioned in several stages without inserting code into the development environment in between. Through this, a learning success/progress is recognisable, because the students saw independently that the generated code would not bring the desired result. For example, they were first asked “How do I add a new column?” and then “Add a new column with the title class in df2”. Another peculiarity was that when ChatGPT entered the same error code twice (without using regenerate), it gave an off-target answer the first time and a customised and on-target

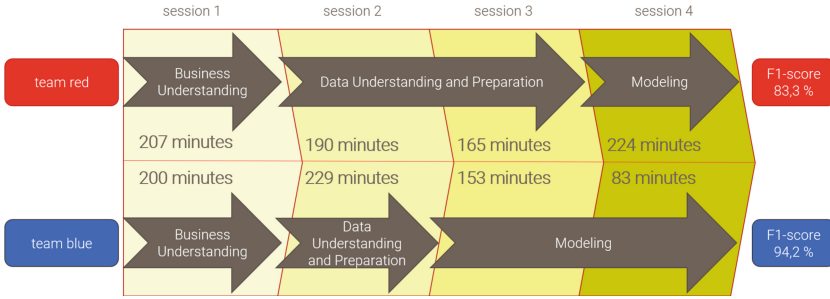


Fig. 2. Teamwork Sessions in Comparison

answer the second time. ChatGPT first replied “Sorry for the misunderstanding. To avoid the SettingWithCopyWarning, you can adjust the code as follows: ...” and then with “Sorry again for the misunderstanding. To avoid the SettingWithCopyWarning, you can correct the code as follows:...”. ChatGPT did not always provide the best answers. For example, the bag-of-words method was recommended to convert textual data (online feedback) into numerical features. Nowadays, modern methods exist that would not have lost contextual information or semantic relationships during the conversion, which could have resulted in better classification performance. The participants were not able to understand the generated code in detail and, at least in part, showed little understanding of it. For example, the code for logistic regression was asked twice. The only change in the second prompt was to change the distribution from 90 % training data and 10 % test data to 80 % training data and 20 % test data, which could have been done directly in the development environment by making the smallest changes. Elsewhere, ChatGPT was queried for code to count the frequency of certain words. However, the words searched for were then changed in several iterations directly in the development environment.

Interaction behaviour with ChatGPT of the team blue: The team generated the code for the project using 53 prompts. There was a better understanding of how to adapt the generated code snippets to the previous programming in order to produce executable code. In the group, it took much longer to give ChatGPT specific guidelines that directed the response behaviour so that the generated code needed less customisation and fit better with the given identifiers. Prompts like this “how do I plot an expression in Python that I would otherwise render with print?” show that there was at least a basic understanding of programming. The documentation given in this group on the methodology gave more specific recommendations for action, which is why technical terms such as lemmatisation, stemming or class imbalance reduction were sometimes entered in prompts.

Both groups managed to develop a classifier that can distinguish positive from negative comments by loading a CSV file into a DataFrame, building an understanding of the data, qualitatively enhancing it and implementing a model using the logistic regression algorithm. All without machine learning skills. The cre-

ated classifiers differ in their F1-scores (team red 83.3 % and team blue 94.2 %) and the time needed for implementation (Fig. 2).

## 5 Discussion

The current findings should be interpreted with several limitations in mind. Typically, machine learning projects are carried out by people with training in programming or machine learning. In such a more realistic scenario, the use of ChatGPT would have been more effective. The setting, incorporated into a controlled experiment, did not allow the participants to use sources other than ChatGPT, the methodology documents and the lecturers for programming. In other circumstances, the ChatGPT outputs could have been supported with internet research, for example, which might have produced better results.

We would like to stimulate a discussion that sheds light on the relevance and effectiveness of using ChatGPT for programming in a business studies course.

## 6 Summary, Conclusion and Future Work

We described a case study in which two small teams of business students were given the task of building a sentiment analysis model for German medical practitioner feedback from patients.

In conclusion, our findings support the hypothesis that subjects with little to no relevant training in programming or machine learning can still complete tasks that were traditionally required to require programming skills. However, limitations have been identified that may distinguish the results from those of an experienced machine learning engineering team. In many cases, even simple code fragments were not generated in a functional way in the context of the project, or only after several attempts by ChatGPT. The approaches provided by ChatGPT were helpful for people with programming knowledge even in complex situations, but the students without the necessary know-how could not assess which simple adjustments would be necessary in the code snippet to create executable programme code. According to the authors, at least a basic programming education is necessary to interact effectively and efficiently with ChatGPT for the purpose of creating software.

Feedback indicates that the students' own perception is not that they had a learning experience that deepened their understanding; rather, it was a "gap filling" experience: the foundational language model took the seat that should be filled by a knowledgeable team member.

Elsewhere, we describe how much programming knowledge these models really contain, given that they were conceived to model human language, not programming language(s). In future work, it would be intriguing to explore how Chatbot-based foundational language models could be integrated in tutorial systems that aim at teaching programming principles and programming (language) skills. The need for this is that the demand for programming talent is unmet, not replaced by language models.

**Acknowledgements.** The authors would like to thank the six participants in the experiment, without whom this article would not have been possible, and the anonymous reviewers for their valuable feedback. Supported by BMBF Grants 16DHBKI089, 16DHBKI090 and 16DHBKI091; we would also like to thank the Free State of Bavaria for funding this research under the Hightech Agenda Bavaria R&D programme.

## References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). <https://doi.org/10.18653/v1/N19-1423>, <https://aclanthology.org/N19-1423>
2. James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J.: An Introduction to Statistical Learning—with Applications in Python. Springer Texts in Statistics, Springer Nature, Cham, Switzerland (2023). <https://doi.org/10.1007/978-3-031-38747-0>
3. Kasneci, E., et al.: ChatGPT for good? On opportunities and challenges of large language models for education. *Learn. Individ. Differ.* **103**, 102274 (2023) <https://doi.org/10.1016/j.lindif.2023.102274>, <https://www.sciencedirect.com/science/article/pii/S1041608023000195>
4. Sjoeborg, D., et al.: A survey of controlled experiments in software engineering. *IEEE Trans. Software Eng.* **31**(9), 733–753 (2005). <https://doi.org/10.1109/TSE.2005.97>
5. Vaithilingam, P., Zhang, T., Glassman, E.L.: Expectation vs. experience: evaluating the usability of code generation tools powered by large language models. In: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems. CHI EA 22, Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3491101.3519665>, <https://doi.org/10.1145/3491101.3519665>
6. Vaswani, A., et al.: Attention is all you need (2017). <https://arxiv.org/pdf/1706.03762.pdf>
7. Zhao, W.X., et al.: A survey of large language models (2023)