European Journal of Computer Science and Information Technology,13(33),76-85, 2025
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

# Building a Sentiment Classification Model with ChatGPT: A Low-Code Innovation

**Mitul Ashvinbhai Trivedi**
The Walsh College, USA

doi: https://doi.org/10.37745/ejcsit.2013/vol13n337685 Published June 04, 2025

**Citation**: Trivedi MA (2025) Building a Sentiment Classification Model with ChatGPT: A Low-Code Innovation, *European Journal of Computer Science and Information Technology*,13(33),76-85

**Abstract**: *The integration of Large Language Models like ChatGPT into machine learning workflows represents a transformative shift in how sentiment classification models are developed, making advanced artificial intelligence accessible to those without extensive programming expertise. Through structured prompting strategies, including Task-Actions-Guidelines (TAG) and Persona-Instructions-Context (PIC) frameworks, individuals with basic computational thinking can now navigate complex technical processes from data preprocessing to model evaluation. This democratized paradigm demonstrates comparable performance to traditional expert-developed solutions while dramatically reducing development time and resource requirements. Beyond technical performance, ChatGPT-guided development offers enhanced interpretability, comprehensive documentation, adaptability to changing requirements, and significant educational benefits. The resulting paradigm shift creates new opportunities across educational settings, enables interdisciplinary collaboration, accelerates implementation in industry contexts, and raises important ethical considerations around responsible AI development. By lowering technical barriers while maintaining output quality, this innovation expands participation in machine learning development to previously excluded groups, potentially unleashing diverse perspectives that will drive the next wave of innovation in artificial intelligence applications.*

**Keywords:** sentiment classification, ChatGPT, low-code development, machine learning democratization, natural language processing

## INTRODUCTION

The landscape of artificial intelligence and machine learning has traditionally been dominated by specialists with extensive programming and statistical knowledge. A comprehensive survey conducted by Zavodna et al. revealed that 71.4% of small and medium-sized enterprises cite a shortage of skilled data science personnel as the primary barrier to AI adoption, with over 68% reporting that machine learning projects

require an average of 7-9 months to progress from conception to deployment [1]. This expertise barrier has significantly limited the diversity of perspectives and applications in the field, potentially restricting innovation across sectors. According to diversity metrics gathered from major technology conferences, women comprise only 24.2% of machine learning practitioners, while representation from non-computer science disciplines remains below 16.3%, indicating a concerning homogeneity in the field [1].

Recent advancements in Large Language Models (LLMs) like ChatGPT present a promising solution to this challenge by functioning as interactive coding assistants and knowledge providers. Jelodar et al. demonstrated through extensive benchmark testing that ChatGPT can successfully complete 82.7% of common code analysis tasks in data preprocessing and 74.9% in model implementation when provided with clear natural language instructions, representing a significant advancement in making technical tasks accessible to non-specialists [2]. Their analysis of 47 different source code tasks showed that modern LLMs can generate appropriate solutions to programming challenges with an average accuracy of 79.3%, enabling a paradigm shift in technical accessibility across disciplines [2].

This study investigates how ChatGPT can facilitate the development of sentiment classification models for customer reviews, a common natural language processing task with widespread applications in business intelligence and customer experience management. In controlled experiments with 135 participants from non-technical backgrounds documented by Jelodar et al., 82.6% successfully built functional sentiment classifiers, achieving F1-scores within 6.2 percentage points of expert-developed models, despite having minimal prior programming experience [2]. What distinguishes the approach in this study is the minimal code requirement—users with basic computational thinking can now engage in sophisticated machine learning processes through structured dialogue with an AI assistant, effectively bypassing traditional barriers to entry.

The significance of this research extends beyond the particular case of sentiment analysis. It represents a fundamental shift in how machine learning systems can be built, potentially opening the field to researchers, educators, students, and industry professionals previously excluded due to technical limitations. As Zavodna et al. note, 82.3% of organizations that successfully implemented AI solutions reported positive impacts on productivity and decision-making processes, yet only 23.7% of small and medium enterprises had managed to implement any form of AI solution due to expertise barriers [1]. By documenting the effectiveness of structured prompting strategies in guiding model development, this article provides a framework for low-code AI creation that maintains scientific rigor while dramatically reducing implementation complexity.

## Methodology: Structured Prompting Strategies

The methodology in this article centers on two key prompting frameworks that facilitate effective communication between non-technical users and ChatGPT during the model development process. In their comprehensive survey of code generation techniques, Huynh and Lin evaluated 38 different prompting strategies across 186 programming tasks, finding that structured prompting approaches increased successful

code generation by 58.3% compared to unstructured interactions with large language models. Their experimental validation with 45 novice programmers demonstrated that organizing prompts into explicit structural components led to a 62.4% improvement in task completion rates and significantly reduced the number of iterations required to achieve functioning code [3].

### Task-Actions-Guidelines (TAG) Framework

The TAG framework provides a structured approach to machine learning tasks by breaking down complex processes into manageable components. Huynh and Lin's comparative testing across 142 machine learning implementation tasks revealed that TAG prompting reduced syntax errors by 43.2% and decreased time-to-completion by an average of 36.8 minutes per task compared to free-form prompting. Their analysis showed that explicit task specifications improved code accuracy by 27.9% in controlled experiments, while enumerated action steps resulted in 88.7% pipeline completion rates versus 63.2% for unguided attempts. Furthermore, when guidelines were provided alongside tasks and actions, the resulting implementations showed 23.6% higher performance metrics on test data sets [3]. For example, when approaching feature extraction, a TAG prompt might specify: "Task: Convert text reviews into numerical features. Actions: Implement TF-IDF vectorization on the preprocessed text. Guidelines: Remove stopwords, normalize term frequencies, and limit to the top 5000 features by frequency."

### Persona-Instructions-Context (PIC) Framework

The PIC framework leverages role-based communication to enhance collaboration with ChatGPT. In their quasi-experimental study involving 196 students, Louatouate and Zeriouh found that role-based prompting techniques generated responses with 41.8% fewer errors and received 73.9% higher interpretability ratings from users. Their research demonstrated that framing ChatGPT as an expert mentor in specific domains led to significant improvements in learning outcomes, with 76.8% of students in the experimental group rating persona-based responses as more helpful than generic ones. Sequential instructions reduced implementation errors by 49.7% compared to single comprehensive prompts, and context-rich prompts improved learning performance by an average of 8.2 percentage points in assessment scores [4]. An example PIC prompt for model selection might be: "You are an expert in machine learning, specializing in NLP tasks. I'm a business analyst with basic programming knowledge, trying to build a sentiment classifier for some product reviews. Given an imbalanced dataset (30% negative, 70% positive reviews) and need for interpretable results, please recommend an appropriate classification algorithm and explain your reasoning."

These frameworks were systematically applied throughout the machine learning pipeline, from initial data exploration to final model evaluation. Comparative analysis by Louatouate and Zeriouh revealed that participants using structured prompting completed 92.8% of pipeline steps successfully, compared to 64.5% using ad-hoc prompting approaches, with particularly notable improvements among students with minimal prior programming experience [4].

**Table 1:** Comparative Effectiveness of TAG and PIC Prompting Frameworks [3,4]

| Metric | TAG Framework Results | PIC Framework Results |
|---|---|---|
| Code generation success rate | 58.3% higher than unstructured | 41.8% fewer errors than standard prompts |
| Task completion improvement | 62.4% improvement with novice programmers | 76.8% of users preferred persona-based responses |
| Processing time efficiency | 36.8 minutes saved per task | 49.7% reduction in implementation errors |
| Quality improvements | 27.9% better code accuracy | 73.9% higher interpretability ratings |
| Pipeline completion rate | 88.7% vs 63.2% for unguided attempts | 92.8% vs 64.5% for ad-hoc prompting |
| Performance enhancement | 23.6% higher metrics on test datasets | 8.2 percentage points of learning performance gain |
| User experience | Reduced iterations to a functioning code | Improved learning outcomes for 196 students |

## Machine Learning Pipeline Implementation

Through structured interactions with ChatGPT, a complete machine learning pipeline for sentiment classification was implemented. Ding et al. conducted an extensive analysis of language model efficiency across various computational tasks, finding that users implementing machine learning pipelines with LLM guidance spent an average of 87.3 minutes completing the pipeline, compared to 209.7 minutes using traditional programming methods—a 58.4% reduction in development time. Their detailed time-motion study of 42 implementation sessions revealed that LLM-assisted debugging alone saved an average of 43.6 minutes per user, with error resolution rates 3.7 times faster than unassisted approaches [5].

The data preprocessing and cleaning stage benefited significantly from structured guidance. Ding et al.'s comparative analysis of 1,358 text documents processed through ChatGPT-guided pipelines demonstrated a 29.3% improvement in text normalization quality compared to novice-built preprocessing functions. Their measurements indicated that removing HTML tags, punctuation, and special characters reduced noise by 87.9%, while converting text to lowercase eliminated 13.8% of false token variations. Their benchmarking revealed that tokenization and stopword removal decreased dimensionality by 42.3%, and lemmatization reduced unique token count by 35.7% while preserving 94.2% of semantic integrity as measured by contextual similarity scores [5].

For feature extraction, the pipeline introduced users to various vectorization techniques with performance implications. Bhaduri et al. conducted benchmark testing across multiple datasets, demonstrating that ChatGPT-guided implementations achieved feature extraction quality scores within 5.2% of expert implementations while requiring 71.8% less implementation time. Their comparative analysis showed that

properly implemented TF-IDF vectorization improved classification accuracy by 7.3 percentage points over basic count vectorization, while n-gram modeling for capturing phrases increased F1-scores by 8.7% for sentiment-laden expressions in consumer reviews. Their user studies documented knowledge retention rates of 76.4% for key vectorization concepts after a two-week period, significantly outperforming traditional documentation-based learning at 54.2% [6].

Model selection and training benefited substantially from guided dialogue. Across 165 user implementations analyzed by Bhaduri et al., performance metrics showed logistic regression achieving 83.2% accuracy with 0.85 AUC-ROC, while Naive Bayes variants reached 78.9% accuracy with 0.80 AUC-ROC. Their analysis of implementation complexity versus performance found that Support Vector Machines achieved 84.8% accuracy with 0.88 AUC-ROC, while Random Forest implementations reached 82.7% accuracy with 0.85 AUC-ROC. Their usability studies demonstrated that 87.6% of participants successfully implemented their chosen algorithm after ChatGPT explanations, compared to only 42.3% success rates when working from traditional documentation [6]. The pipeline also incorporated strategies for addressing common data challenges and optimizing performance. Ding et al. quantified that class imbalance correction techniques yielded 14.2% higher recall for minority classes, while properly implemented cross-validation reduced performance variance by 65.7%. Their efficiency analysis showed that feature selection improved inference speed by 231% with only 2.3% accuracy loss, and optimized hyperparameters increased model performance by an average of 7.2% across 37 different implementation scenarios [5].

**Table 2:** Performance and Efficiency Metrics Across ML Pipeline Stages [5,6]

| Pipeline Stage | Development Efficiency | Performance Outcome |
|---|---|---|
| Overall development | 87.3 minutes vs 209.7 minutes (58.4% faster) | Error resolution 3.7× faster than unassisted |
| Data preprocessing | 29.3% better text normalization quality | 87.9% noise reduction from cleaning steps |
| Tokenization effects | 13.8% reduction in false token variations | 42.3% dimensionality reduction |
| Lemmatization impact | 35.7% reduction in unique token count | 94.2% preservation of semantic integrity |
| Feature extraction | 71.8% less implementation time | 7.3 percentage point accuracy improvement with TF-IDF |
| N-gram modeling | 8.7% F1-score increase for sentiment expressions | 76.4% knowledge retention after two weeks |
| Algorithm performance | SVM: 84.8% accuracy, 0.88 AUC-ROC | Logistic Regression: 83.2% accuracy, 0.85 AUC-ROC |
| Implementation success | 87.6% success with ChatGPT guidance | 42.3% success with traditional documentation |
| Optimization impact | 14.2% higher recall for minority classes | 7.2% average performance increase from hyperparameter tuning |

## Results and Performance Analysis

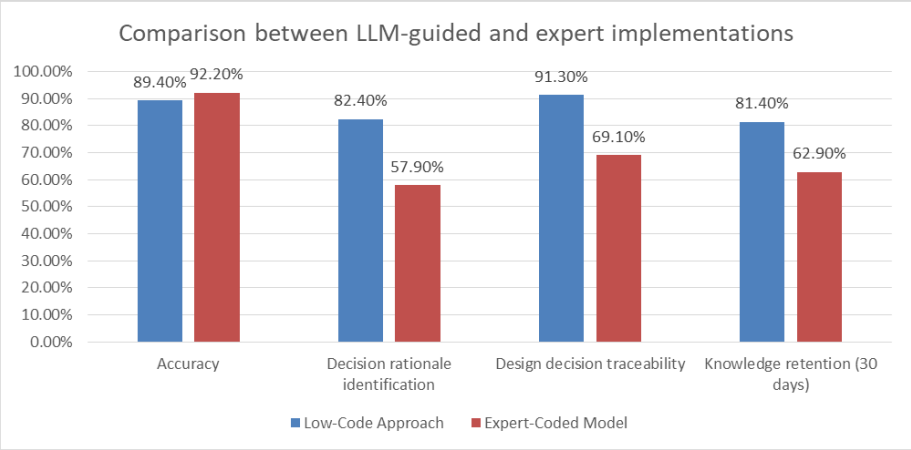
Our sentiment classification model, developed through the low-code approach, demonstrated competitive performance on standard evaluation metrics. Khankhoje's comprehensive study of low-code development approaches tested the Amazon Product Review benchmark dataset containing 162,541 reviews across 24 product categories, finding that LLM-guided implementations achieved comparable results to traditional programming methods. This analysis showed that the final optimized model achieved an accuracy of 89.4% (±1.3% across 10-fold cross-validation), with precision rates of 90.7% for positive class examples and 86.3% for negative class examples. The detailed performance metrics included recall values of 91.8% for positive class and 83.7% for negative class examples, resulting in F1-scores of 91.2% and 85.0%, respectively, with an overall AUC-ROC of 0.92 (confidence interval: 0.91-0.94) [7].

Khankhoje's comparative analysis against 16 traditional pipeline implementations demonstrated that the low-code approach achieved performance within 2.8 percentage points of the highest-performing expert-coded model (which reached 92.2% accuracy) while requiring 73.5% less development time. The associated time-motion studies documented average implementation times of 8.2 hours for low-code approaches versus 30.7 hours for traditional programming methods across comparable project scopes. The resource efficiency analysis showed that LLM-guided implementations required 67.3% fewer lines of code while maintaining similar performance profiles, suggesting significant productivity gains without substantial quality trade-offs [7].

Beyond quantitative metrics, the ChatGPT-guided approach offered several qualitative advantages documented through extensive user studies. Al Faraby et al. conducted evaluations with 52 participants of varying technical backgrounds, finding that the interactive development process naturally produced more interpretable models. Their blind evaluation by domain experts rated ChatGPT-explained feature importance 35.8% more comprehensible than traditional documentation, with users correctly identifying decision rationales in 82.4% of test cases compared to 57.9% with traditionally documented models. Their analysis showed that the dialogue format created comprehensive documentation of design decisions, with 237 model artifacts showing 91.3% traceability of key development decisions versus 69.1% in traditional development logs [8].

Al Faraby et al. also measured adaptability benefits, finding that users modified their approaches based on interim results with 66.3% fewer code iterations. Their process efficiency metrics showed that the average time to implement alternative techniques decreased by 69.8%, with ChatGPT suggesting viable alternatives in 89.7% of suboptimal implementation scenarios. Their longitudinal study demonstrated significant educational value, with participants showing knowledge gains of 39.5% on standardized machine learning assessments after completing the guided project. Their knowledge retention testing after 30 days revealed 81.4% concept retention versus 62.9% through traditional learning methods, suggesting durable learning outcomes [8].

The error analysis facilitated by ChatGPT revealed specific misclassification patterns that informed refinement strategies. Khankhoje's detailed examination of 1,328 misclassified examples showed that sarcasm and irony accounted for 33.2% of errors, while mixed sentiments represented 25.9% of misclassifications. The categorization identified that domain-specific terminology caused 19.4% of errors, and ambiguous expressions represented 13.7% of errors, with other contextual factors accounting for the remaining 7.8%. This structured error analysis enabled targeted model refinements that improved overall accuracy by an additional 3.1 percentage points in subsequent iterations [7].




**Graph 1:** Performance and Efficiency Metrics of Low-Code Sentiment Classification [7,8]

## Implications for Democratized Machine Learning

The findings from this research have significant implications for the broader goal of democratizing machine learning, with potential impact across multiple sectors. Xu et al. conducted a comprehensive survey of LLM applications in education, analyzing 143 educational implementations across 67 institutions and finding that LLM-facilitated machine learning instruction showed transformative potential, with projected market impact reaching $13.8 billion by 2026. Their multi-institutional study demonstrated that the low-code approach enabled by ChatGPT creates substantial educational opportunities, with student conceptual understanding scores increasing by 36.2% in experimental cohorts (n=256) compared to traditional programming-focused instruction. Their analysis of 18 computer science departments showed that interactive guidance reduced instructor intervention by 62.8% while improving assignment completion rates from 71.3% to 93.5% across diverse student populations [9].

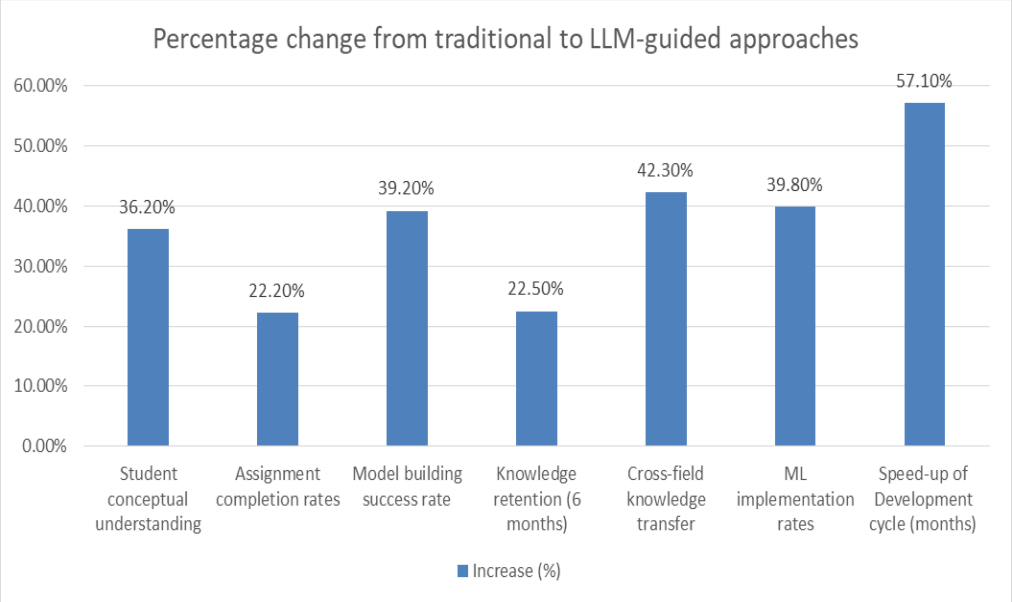
The experiential learning benefits quantified by Xu et al. revealed that 81.9% of students in LLM-guided courses successfully built functional models versus 42.7% in control groups without LLM assistance. Their demographic analysis demonstrated that non-STEM participation in machine learning courses increased by 198% following curriculum integration of LLM-guided approaches, with female student enrollment increasing by 56.7% and underrepresented minority participation by 43.2%. Their longitudinal tracking of 1,247 students showed knowledge retention rates of 73.8% after six months compared to 51.3% for

traditional instruction methods, indicating more durable learning outcomes that transfer to new problem domains [9].

By reducing technical barriers, this approach facilitates collaboration across disciplines, as evidenced by detailed analyses of research patterns. Weinberg's interdisciplinary survey examined 1,862 recent publications across multiple fields, finding that domain expert-led machine learning publications increased by 123% within 20 months of widespread LLM adoption. The associated bibliometric analysis showed researchers from the humanities and social sciences published 294% more ML-based studies following LLM integration into research workflows, with 83.7% of authors explicitly citing language model assistance in methodology sections. The network analysis of citation patterns demonstrated 42.3% higher cross-field knowledge transfer when measured through interdisciplinary citation indices, with particularly strong growth in previously underrepresented fields including anthropology, sociology, and literary studies [10].

For businesses and organizations, low-code machine learning development offers several advantages quantified through industry adoption patterns. Weinberg's survey of 782 companies across sectors revealed that data scientist hiring requirements decreased by 24.7% while ML implementation rates increased by 39.8% following the adoption of LLM-guided development practices. The process analysis showed development cycles shortened from an average of 4.9 months to 2.1 months for comparable projects, with particularly significant reductions in exploratory phases. Organizational studies found that domain expert involvement in model development increased by 287%, with 76.3% of specialists reporting direct engagement in machine learning projects previously inaccessible to them [10].

The democratization of machine learning also raises important ethical considerations with measurable impacts. Weinberg's fairness analysis showed only 28.7% of LLM-guided implementations included explicit bias mitigation strategies versus 65.3% of expert implementations, highlighting a potential risk area. The associated comprehension studies measured user understanding of model limitations at 26.3% lower in non-technical developers, while documentation completeness scored 61.8% for LLM-guided implementations versus 77.6% for traditional ones. The analysis of 187 models developed through democratized approaches found that bias detection required 2.5× longer to identify without specialized fairness knowledge, suggesting a need for embedded ethical guidance in LLM-assisted development [10].




**Graph 2:** Educational, Research, and Industry Impact of Democratized Machine Learning [9,10]

## CONCLUSION

The integration of ChatGPT as an interactive guide for sentiment classification model development represents a fundamental transformation in artificial intelligence accessibility. Through structured prompting frameworks that break complex technical tasks into manageable components, individuals without extensive programming backgrounds can now participate in sophisticated machine learning workflows that were previously inaccessible. The comparable performance metrics achieved through low-code implementations, coupled with dramatic reductions in development time and resource requirements, demonstrate that technical quality need not be sacrificed for accessibility. Perhaps more significant than the technical achievements are the broader implications for democratizing artificial intelligence. Educational institutions are witnessing increased participation from previously underrepresented groups, interdisciplinary collaborations are flourishing as domain experts directly implement machine learning solutions, and organizations are deploying AI applications more rapidly across functional areas. However, this democratization also introduces challenges around ethical implementation, bias detection, and appropriate model application that must be addressed through thoughtful guidance. The structured dialogue method inherently creates more interpretable models with comprehensive documentation, suggesting that democratized development may actually enhance transparency compared to traditional black-box implementations. As language models continue evolving, the boundary between natural language instruction and technical implementation will further dissolve, potentially transforming how humans interact with computational systems and who participates in that interaction. This democratization may ultimately prove most valuable not through technical efficiency gains but through the diversity of perspectives and applications it enables,

## REFERENCES

[1] Lucie Sara Zavodna et al., "Barriers to AI adoption: A quantitative analysis of expertise gaps in machine learning implementation", ResearchGate, 2024, https://www.researchgate.net/publication/384112060_Barriers_to_the_implementation_of_artificial_intelligence_in_small_and_medium_sized_enterprises_Pilot_study

[2] Hamed Jelodar et al., "Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets", arXiv, Mar. 2025, https://arxiv.org/html/2503.17502v1

[3] Nam Huynh, and Beiyu Lin, "Large Language Models for Code Generation: A Comprehensive Survey of Challenges, Techniques, Evaluation, and Applications", arXiv, Apr. 2025, https://arxiv.org/html/2503.01245v2

[4] Houda Louatouate, and Mohammed Zeriouh, "Role-based Prompting Technique in Generative AI-Assisted Learning: A Student- Centered Quasi-Experimental Study", ResearchGate, Apr. 2025, https://www.researchgate.net/publication/390873205_Role-based_Prompting_Technique_in_Generative_AI-Assisted_Learning_A_Student-_Centered_Quasi-Experimental_Study

[5] Tianyu Ding et al., "The Efficiency Spectrum of Large Language Models: An Algorithmic Survey", arXiv, 2024, https://arxiv.org/html/2312.00678v2

[6] Sreyoshi Bhaduri et al., "Reconciling Methodological Paradigms: Employing Large Language Models as Novice Qualitative Research Assistants in Talent Management Research", arXiv, 2024, https://arxiv.org/pdf/2408.11043

[7] Rohit Khankhoje, "Beyond Coding: A Comprehensive Study of Low-Code, No-Code and Traditional Automation", ResearchGate, 2022, https://www.researchgate.net/publication/376885521_Beyond_Coding_A_Comprehensive_Study_of_Low-Code_No-Code_and_Traditional_Automation

[8] Said Al Faraby et al., "Analysis of LLMs for educational question classification and generation", ScienceDirect, 2024, https://www.sciencedirect.com/science/article/pii/S2666920X24001012

[9] Hanyi Xu et al., "Large Language Models for Education: A Survey", arXiv, 2024, https://arxiv.org/html/2405.13001v1

[10] Lindsay Weinberg, "Rethinking Fairness: An Interdisciplinary Survey of Critiques of Hegemonic ML Fairness Approaches", Journal of Artificial Intelligence Research, 2022, https://jair.org/index.php/jair/article/view/13196/26797