UCL

UCL
SCHOOL OF
MANAGEMENT

Module code/name	**MSIN0097 Predictive Analytics**
Module leader name	**Dr A P Moore**
Academic year	**2024/25**
Term	**2**
Assessment title	**Group Coursework**
Individual/group assessment	**Individual**
Candidate Numbers	**MYBQ6,MPZY3,NZVK0,KYDR5,LLRR3**
Word Count	**1990**

Introduction

With the rapid advancements in artificial intelligence, effective interaction with large language models (LLMs) has become crucial. This project explores prompt engineering, a key technique used to optimize AI model responses. By conducting a comparative analysis across multiple AI models - such as OpenAI's GPT, Google's Gemini, and Anthropic's Claude - this study aims to assess how different prompting strategies impact model performance.

The project first looks into a literature review using 10 academic papers using existing research on prompt engineering, which shows that prompt engineering significantly enhances the accuracy of LLM-generated code. However, a key gap in current studies is the reliance on broad qualitative metrics, such as overall accuracy, consistency and reliability. This is then used to create a comprehensive evaluation framework, which is used to assess prompt and LLM performance for ChatGPT, Gemini and Claude.

Literature Review

As defined by Muktadir (2023), prompt engineering is the purposeful designing of inputs to guide large language models, playing a critical role in shaping the accuracy and reliability of LLM-generated code responses. Despite advancements in prompt design, concerns remain regarding whether LLMs overestimate their code accuracy, potentially leading to erroneous outputs. This review explores research on prompt effectiveness, methods for evaluating LLM-generated code, and biases in estimated accuracy. Key themes include (1) Prompt Engineering & LLM Code Generation, (2) Measuring Accuracy in LLMs, and (3) Limitations & Bias in LLM-Generated Code (Brown et al., 2020; Liu et al., 2023).

Prompt Engineering & LLM Code Generation

Prompt engineering is crucial for optimizing LLM responses, especially in code generation. Achieving optimal outputs depends on factors like prompt specificity, design strategies, and iterative refinement, all of which enhance accuracy, functionality, and performance.

According to a paper by Murr et al. (2023), highly specific prompts, such as those that include constraints or intended outcomes, improved the quality of code generated across LLMs (Bard, GPT-3.5, GPT-4, Claude). This comparison was done on criteria such as accuracy, reliability, adaptability and reasoning capacity to highlight the importance of deliberate prompt construction. The study further acknowledges the difficulty of defining and standardising prompt specificity, which creates challenges for reproducibility and cross-study comparison.

Additionally, Hou & Ji (2024) evaluated the performance of LLMs in generating code by using several prompt strategies, programming languages and task difficulties, highlighting comparisons of zero-shot versus few-shot prompting. The experiment demonstrated that

few-shot prompts boosted execution success rates and chain-of-thought prompting improved logical flow, performing well on problem-solving tasks.

A paper on Claude 2.0 (Caruccio et al., 2024) introduced an "iterative prompt template engineering approach," refining inputs based on feedback from prior outputs and enhancing performance by 14-32%. This method can enhance LLM-generated code through progressive debugging. These papers emphasise prompt engineering as essential for accurate and efficient LLM-generated code.

Measuring Accuracy in LLM-Generated Code

LLMs use internal confidence scores from token probabilities to estimate accuracy, but these scores can be misleading. Research shows LLMs often overestimate their correctness, particularly in log analysis and zero-shot learning (Liu et al., 2024; Wu et al., 2023). LLM accuracy is assessed by execution success rate, correctness, and debugging efficiency. Studies indicate that while LLMs often assign high confidence to incorrect outputs, their actual accuracy is inconsistent, as highlighted by Son et al. (2025) in discussions on technical applications.

Failures arise when LLM-generated code makes incorrect function calls, misuses APIs, or omits dependencies. For instance, in LogPrompt, errors in log analysis scripts led to system failures due to data misinterpretation by the LLM. To mitigate this, evaluation frameworks use automated testing and human validation. Running tests post-code generation and measuring debugging efficiency are essential for reliability, as self-reported confidence is unreliable in critical applications (Gu et al., 2023).

Limitations and Bias in LLM-Generated Code

Despite advancements in prompt engineering, LLMs still show inherent biases, particularly in self-reported accuracy. A key issue is overconfidence bias, where LLMs generate code with high confidence but often overestimate their accuracy. According to Gallegos et al. (2024), this leads to code that may appear correct based on confidence scores but often contains errors when executed.

A key challenge with LLM-generated code is hallucination, where models may invent non-existent functions or incorrect logic due to ambiguous prompts or misinterpretation. Moreover, LLMs heavily rely on patterns from past data, which can sometimes lead them to create non-existent functions or apply incorrect logic, resulting in flawed outputs (Jain et al., 2024).

Despite advancements in prompt engineering, testing the accuracy and reliability of prompts remains challenging. This is because LLM performance varies across industries, complicating the establishment of benchmarks for comparison. Additionally, differences in datasets and models can introduce biases. (Bakhtiyari, 2024)

Research Gaps

This review highlights that prompt engineering plays a crucial role in improving the accuracy of LLM-generated code. It also reveals a key gap in current research, where LLMs are often evaluated using qualitative and broad metrics such as accuracy. Further research is needed to examine more quantitative metrics, such as the alignment between an LLM's confidence and its actual performance. Additionally, a more objective evaluation framework with a structured marking scheme, including clearly defined evaluation criteria, can help improve the consistency and accuracy of LLM's confidence.

Metrics

Based on the research gaps identified in the literature review, it is essential to assess the performance of LLMs on quantifiable and objective metrics that enhance the robustness and rigour of LLM evaluation. Therefore, a comprehensive evaluation framework has been developed to conduct a well-rounded appraisal, including different metric categories. These are core self-assessment, technical accuracy, confidence expression and metacognitive strategies.

Core Assessment Metrics

First, it is vital to evaluate an LLM's assessment abilities to ensure reliable and trustworthy responses. The following core metrics determine how well the model understands its correctness, limitations, task complexity and adaptability. These metrics are then used to create a final summary value - Weighted Assessment Score - which takes the weighted average of all the metrics in the category.

$$WAS = (CPC \times 0.25) + (Cal \times 0.25) + (DA \times 0.15) + (ER \times 0.15) + (DSV \times 0.10) + (PS \times 0.10)$$

Where:

CPC = Confidence-Performance Correlation
Cal = Calibration Error
DA = Task Difficulty Awareness
ER = Error Recognition
DSV = Domain-Specific Variance
PS = Prompt Sensitivity

This is used to finally determine which model performs the best.

Metric	Purpose of the Metric	Explanation
Confidence-Performance Correlation	Measures alignment between confidence and accuracy	Ensures that the model's confidence is justified by its correctness. A low score indicates overconfidence or underconfidence.
Calibration Error	Evaluates how well confidence aligns with reality	If the model consistently expresses too much or too little confidence in its responses, this metric highlights those mismatches.
Task Difficulty Awareness	Determines if the model understands task complexity	Checks whether the model acknowledges when a task is inherently difficult and adjusts its responses accordingly.
Error Recognition	Assesses if the model acknowledges its own mistakes	A model with poor error recognition might not state limitations, which can mislead users.
Domain-Specific Variance	Measures adaptability across different subjects	Ensures that the model recognizes how different domains (e.g., statistics vs. medical diagnosis) require different approaches.
Prompt Sensitivity	Evaluates how the model adapts to prompt variations	Checks if the model provides different levels of confidence and detail based on how a prompt is phrased.
Weighted Assessment Score (WAS)	Summarizes the model's overall performance across evaluation metrics	A composite score balancing confidence accuracy, error awareness, task difficulty understanding, and calibration.

Technical Accuracy

The second category of metrics assesses different aspects of response accuracy and measures factual correctness, procedural soundness and logical reasoning. This ensures that LLM-generated content is not only informative and credible but also actionable across a wide range of tasks and domains. By assessing these dimensions, this category gives a detailed understanding of the model's strengths and limitations in producing contextually appropriate content. The following table explains the metrics used and why they are required.

Category	Purpose of the Metric	Explanation
Factual Claims	Measures correctness of factual statements	Ensures that the model provides factually accurate information with supporting evidence. A low score indicates incorrect or misleading claims.
Procedural Recommendations	Evaluates correctness of step-by-step guidance	Assesses whether suggested procedures (e.g., code snippets, instructions) are technically sound, executable, and complete.
Inference/Opinions Accuracy	Assesses logic and justification behind interpretations	Determines if the model's reasoning and opinions are valid, well-supported, and logically sound.
Overall Accuracy	Summarizes the model's overall correctness	A composite metric that balances factual accuracy, procedural correctness, and logical consistency.

Confidence Expression

The third metric category gauges how an LLM expresses confidence, which is key to understanding the model's reliability and communication clarity. The following metrics evaluate the frequency and nuance of confidence signals, ranging from bold statements to uncertainty indicators. This ensures the model's tone lines up with its accuracy and the complexity of user prompts.

Type	Purpose of the Metric	Explanation
Explicit Confidence Statements	Tracks direct confidence expressions	Counts occurrences of phrases like "I am certain" or "This is likely correct." Helps assess how often the model explicitly states confidence levels.
Certainty Markers	Identifies phrases indicating strong belief	Examples include "definitely," "without a doubt," and "highly likely." Measures how often the model expresses high confidence.
Hedge Words	Tracks uncertainty indicators	Words like "possibly," "might," and "could be" indicate a cautious stance. Helps detect underconfidence or uncertainty in responses.
Qualifying Phrases	Measures context-specific confidence	Phrases like "in most cases" or "depending on X" show nuanced confidence expressions, balancing certainty with contextual awareness.
Overall Estimated Confidence	Computes average confidence across statements	Helps determine if the model is generally overconfident, underconfident, or well-calibrated in its responses.

Metacognitive Strategies

The final category aims to demonstrate metacognitive awareness in LLMs to go beyond surface-level answers. The metrics in the table below assess whether the model can reflect on knowledge limits, justify reasoning, handle uncertainty and cite credible sources. These highlight the model's ability to monitor itself and communicate with clarity.

Strategy	Purpose of the Metric	Explanation
Knowledge Boundary Articulation	Measures awareness of knowledge limitations	Assesses if the model explicitly states when it doesn't know something or acknowledges uncertainty.
Confidence Calibration	Evaluates how well the model adjusts confidence based on certainty	Ensures the model scales confidence appropriately depending on the reliability of its response.
Reasoning Transparency	Assesses clarity of thought process	Checks if the model provides clear step-by-step reasoning rather than just giving an answer.
Alternative Consideration	Determines if the model explores multiple solutions	Evaluates whether the model considers various possibilities instead of assuming a single correct answer.
Information Source Qualification	Tracks whether the model cites sources	A higher score indicates the model properly references external sources for claims, improving credibility.
Temporal Qualification	Determines if the model considers time-sensitive information	Some answers may change over time, and the model should acknowledge this when relevant.
Logical Qualification	Assesses logical consistency in responses	Ensures that the model's arguments are logically sound, coherent, and do not contradict themselves.
Uncertainty Decomposition	Evaluates how well the model explains uncertainty	A strong model breaks down *why* it's uncertain rather than just stating uncertainty, improving interpretability.

Evaluation Method

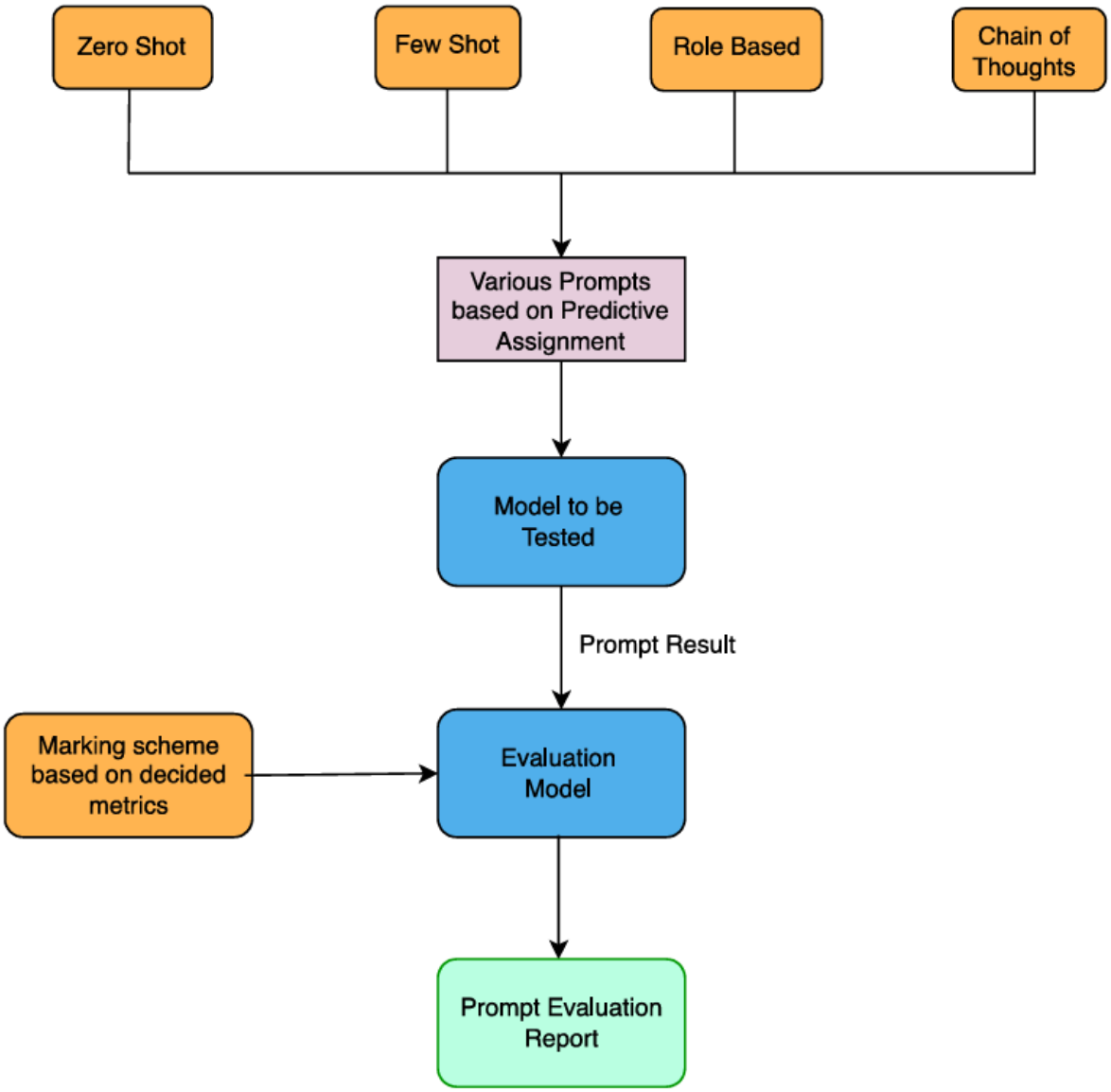
The effectiveness of AI-generated responses is significantly influenced by prompt engineering. By crafting and testing various prompts across multiple AI models, their strengths, weaknesses and best practices can be assessed to achieve optimal results. Additionally, the different prompt types were explored by utilizing three AI models: Claude, Gemini (Google), and OpenAI's ChatGPT, to generate responses. These responses were subsequently evaluated for accuracy and coherence, with ChatGPT and Claude serving as secondary validators.

In this study, several prompting techniques were employed to evaluate AI performance. The first technique was zero-shot prompting, which involved providing a query without any examples or context. Next, few-shot prompting involves supplying a query along with a few examples to guide the model. The third technique introduced was chain-of-thought prompting, which encouraged the models to articulate their reasoning in a step-by-step manner. Finally, role-based prompting was examined by designating different roles to the AI model.

The following steps were used to systematically test the effectiveness of these prompts across models:

1. Selecting prompts:
    - For each prompt type, prompts based on an individual predictive coursework were selected (4*3*4=48 prompts - see Appendix for all prompts).
2. Generating responses with Claude, Gemini, and ChatGPT:
    - Various prompts were designed following the taxonomy above.
    - ChatGPT, Claude and Gemini were used to generate responses based on the prompts.
3. Testing responses using ChatGPT and Claude:
    - The responses from Claude and Gemini were then input into ChatGPT for evaluation.
    - The responses from ChatGPT were tested and evaluated using Claude.
    - Evaluations focused on technical accuracy, confidence calibration, and the overall coherence and reasoning quality of the responses.
4. Comparing Outcomes:
    - High-quality and suboptimal responses were documented.
    - Analysis was conducted to determine which prompting approach worked best for each model.

The flowchart below visualises the process.

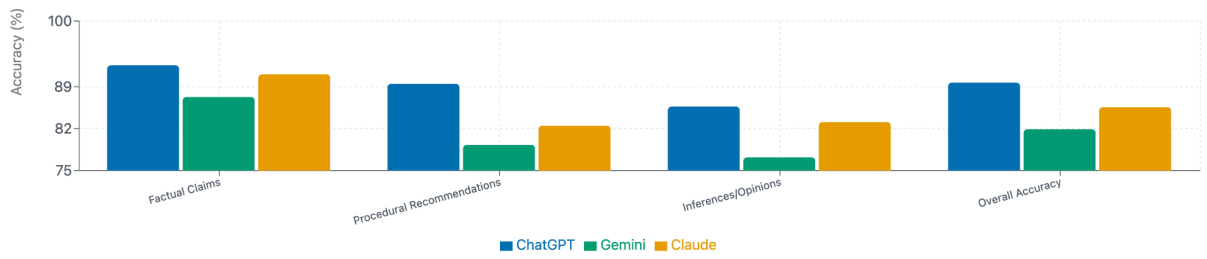


Performance

The evaluation of ChatGPT, Gemini, and Claude reveals distinct patterns of strengths and limitations across technical domains and self-assessment capabilities. Each model differs in model performance based on different characteristics. The key findings are as follows.

Technical Accuracy

Among the three models, ChatGPT demonstrated the highest technical accuracy (89.7%), outperforming Claude (85.6%) and Gemini (81.9%). ChatGPT consistently provided reliable responses across factual claims, procedural recommendations, and inferences, making it the preferred choice for tasks requiring high precision.



Metric	ChatGPT	Gemini	Claude
Overall Accuracy	89.7%	81.9%	85.6%
Factual Claims	92.6%	87.3%	91.1%
Procedural Guidance	89.5%	79.3%	82.5%
Inferences	85.7%	77.2%	83.1%

Confidence-Performance Correlation and Calibration Error

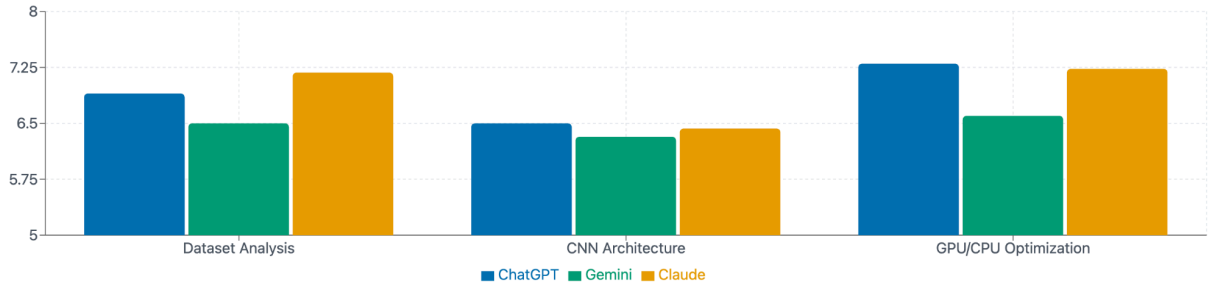
Despite ChatGPT having the highest technical accuracy, Claude performs the highest confidence-performance correlation (7.44/10), i.e. this means that Claude has better ability to assess its own knowledge limitations compared to ChatGPT.

Metric	ChatGPT	Gemini	Claude
Confidence-Performance Correlation	6.70/10	6.53/10	7.44/10
Calibration Error	6.30/10	5.56/10	6.69/10

Performance Across Technical Domains

The models were tested on the following domains:

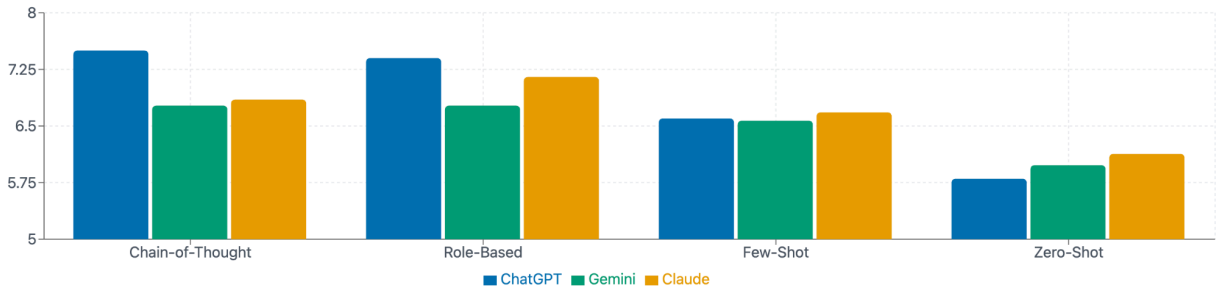
- Dataset Analysis: Claude demonstrates the best performance in preprocessing and demographic data. Both Gemini and Claude have a weakness for categorical data recommendations but ChatGPT is weak in handling imbalance data.
- CNN Architecture/Training: All models performed similarly, with ChatGPT (6.5/10) slightly ahead in model regularization.
- GPU/CPU Optimization: ChatGPT (7.3/10) and Claude (7.23/10) led the category, while Gemini trailed (6.6/10).



Domain	ChatGPT	Gemini	Claude
Dataset Analysis (FairFace)	6.9/10	6.5/10	7.18/10
CNN Architecture & Training	6.5/10	6.32/10	6.43/10
GPU/CPU Optimization	7.3/10	6.6/10	7.23/10

Prompt Effectiveness

The models' adaptability to different prompt structures was assessed using four types: Chain-of-Thought, Role-Based, Few-Shot, and Zero-Shot. ChatGPT responds well to Chain-of-Thought prompts while Claude performs best with Role-based prompts. All models perform the worst with Zero Shot prompts, highlighting the importance of well structured prompts. Gemini shows the most consistent performance across prompt types but achieves lower overall scores. These differences highlight the importance of tailoring prompting strategies to specific models.



Prompt Type	ChatGPT	Gemini	Claude
Chain-of-Thought	7.5/10	6.77/10	6.85/10
Role-Based	7.4/10	6.77/10	7.15/10
Few-Shot	6.6/10	6.57/10	6.68/10
Zero-Shot	5.8/10	5.98/10	6.13/10

Unsuccessful Prompts

Zero-Shot Prompts

Hey, I've been working with the FairFace dataset for age classification. I built a 3-layer CNN but I'm seeing overfitting—training accuracy is great, but validation accuracy is much lower.

Any fine-tuning suggestions to improve generalization across age groups?

Successful Prompts

Chain-of-Thought Prompts (Show Your Thinking Step-by-Step)

I'm training a CNN model, but it's running significantly slower than expected. I'm trying to debug this methodically:

1. **Checked system resource usage:**
   - CPU is at 100% constantly, but GPU usage is barely registering.
   - Should I enable TensorFlow's device placement logging to confirm if the GPU is being used?
2. **Potential fixes:**
   - Explicitly set operations to run on GPU with `tf.device('/GPU:0')`.
   - Reduce batch size to prevent memory overflow issues.
3. **Further optimization:**
   - Should I enable mixed precision training to speed up computation?

What other debugging steps would you recommend? I feel like I'm missing something obvious about why the GPU isn't being utilized properly.

Role-Based Prompts

You are a deep learning expert specializing in CNN models for demographic classification.

Task:

- Diagnose potential overfitting issues in my CNN model.
- Recommend specific fine-tuning techniques to improve generalization.

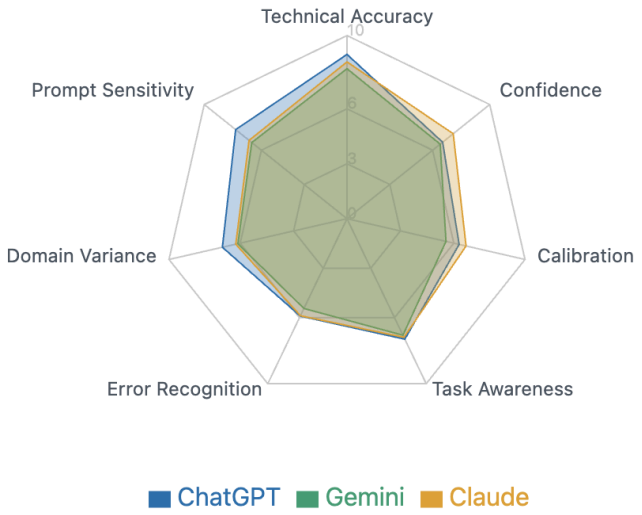
Dataset: FairFace **Baseline Accuracy:** Low

Your response should include:

- Hyperparameter tuning strategies
- Regularization methods
- Alternative model architectures that could improve generalization

Weighted Average Score

From the weighted average score, ChatGPT is the most technically proficient model, while Claude excels in confidence alignment and dataset analysis. Gemini, though balanced, struggles with accuracy and calibration provides the most consistent performance across different prompt types, however with lower scores. Claude demonstrates slightly better overall score compared to the other models. This means that Claude's confidence levels are more aligned with its actual performance, making it the most reliable model. The differences between ChatGPT and Claude are relatively small. However, Gemini still has room for improvement.



Limitations

While this experiment provides valuable insights into overall and prompt-specific LLM performance, it has a few limitations. For instance, while generating evaluations, certain metrics would show as not applicable. This indicated inconsistency in LLM evaluation. Further, since only a few prompts were used in the analysis, generalisability is limited. Additionally, despite a structured marking scheme, some bias may still creep in. Finally, since LLMs are constantly evolving and improving, updates may reduce the evaluation validity over time.

Conclusion

The project aimed to critically analyse how different prompt structures influence LLM output in terms of quality and reliability. The experiment indicated that ChatGPT showed the highest technical accuracy, while Claude demonstrated the best confidence calibration. Further, prompt types had a significant impact on outcomes, where chain-of-thought and role-based structures outperformed zero-shot. The Weighted Assessment Score was also computed, offering holistic model comparisons. These findings support the literature on prompt specificity (Murr et al., 2023), while also addressing shortcomings in LLM self-assessment (Liu et al., 2024) by creating an objective, multi-dimensional marking framework. Based on these results, the following recommendations are proposed:

Prompt Engineering Strategy	Recommendation	Justification
Model Selection	Use ChatGPT for technical accuracy, Claude for self-assessment and confidence calibration	ChatGPT showed highest technical accuracy (89.7%) while Claude demonstrated better confidence-performance correlation (7.44/10)
Prompt Structure	Prioritize chain-of-thought and role-based prompts	These prompt types consistently outperformed zero-shot prompts across all models (7.5/10 and 7.4/10 for ChatGPT)
Domain-Specific Tasks	Use Claude for dataset analysis, ChatGPT for GPU/CPU optimization	Claude performed best in preprocessing data (7.18/10), while ChatGPT excelled in optimization tasks (7.3/10)
Confidence Assessment	Include explicit requests for confidence levels when using ChatGPT	Claude naturally provides better calibrated confidence, but ChatGPT needs explicit prompting to assess its certainty
Error Handling	Request explicit error recognition	All models showed improved performance when prompted to acknowledge potential limitations and errors in their responses
Iterative Refinement	Use feedback from previous outputs to refine prompts	The study referenced improved performance by 14-32% when using iterative prompt engineering
Few-Shot Learning	Include examples for complex tasks	Few-shot prompts boosted execution success rates compared to zero-shot approaches

References

Bakhtiyari, S. V**.** (2024). *Descriptive case analysis on the application of prompt engineering in business management*. Academia.edu. https://www.academia.edu/127371963/Descriptive_Case_Analysis_on_the_Application_of_Prompt_Engineering_in_Business_Management

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). *Language models are few-shot learners*. arXiv. https://arxiv.org/abs/2005.14165

Caruccio, L., Senatore, S., Gaeta, M., Ritrovato, P., & Orciuoli, F. (2024). Claude 2.0 large language model: Tackling a real-world classification problem with a new iterative prompt engineering approach. *Intelligent Systems with Applications, 21*, 200336. https://doi.org/10.1016/j.iswa.2024.200336

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., & Ahmed, N. K. (2024, July 12). *Bias and fairness in large language models: A survey*. arXiv. https://arxiv.org/abs/2309.00770

Gu, J., Han, Z., Chen, S., Beirami, A., He, B., Zhang, G., Liao, R., Qin, Y., Tresp, V., & Torr, P. (2023). *A systematic survey of prompt engineering on vision-language foundation models*. arXiv. https://arxiv.org/abs/2307.12980

Hou, W., & Ji, Z. (2024). *A systematic evaluation of large language models for generating programming code*. arXiv. https://arxiv.org/abs/2403.00894

Liu, Y., Li, S., Xie, X., Zheng, Z., Zhang, Z., Yu, S., Duan, X., & Jiang, Y. (2024, April 14). *LogPrompt: Prompt engineering towards zero-shot and interpretable log analysis*. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. https://doi.org/10.1145/3639478.3643108

Muktadir, G. M. (2023). *A Brief History of Prompt: Leveraging Language Models (Through Advanced Prompting)*. arXiv. https://arxiv.org/abs/2310.04438

Murr, L., Grainger, M., & Gao, D. (2023). *Testing LLMs on code generation with varying levels of prompt specificity*. arXiv. https://arxiv.org/abs/2311.07599

Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S. and Chadha, A., 2024. *A systematic survey of prompt engineering in large language models: Techniques and applications*. arXiv preprint arXiv:2402.07927. Available at: https://arxiv.org/abs/2402.07927

Son, M., Won, Y.-J., & Lee, S. (2025, January 30). *Optimizing large language models: A deep dive into effective prompt engineering techniques*. *Applied Sciences, 15*(3), 1430. https://doi.org/10.3390/app15031430

Wu, Y., Dohan, D., Luan, D., & Zhou, D. (2023). *Prompt engineering for reasoning in large language models: A survey*. arXiv. https://arxiv.org/pdf/2311.07599

Appendix

Attached below is the snapshot of one of the reports generated, to refer to all reports and prompts access the following drive link: https://tinyurl.com/bd72bnk6

Comprehensive LLM Assessment Evaluation

Core Assessment Details

Parameter	Details
Prompt	I'm preparing the FairFace dataset for CNN training and have encountered several data quality issues. Step-by-Step Analysis: Missing Values: About 10% of samples have missing age values. Should I use mean imputation or drop these records entirely? Label Inconsistencies: Gender is labeled as "M", "Male", "F", and "Female". Standardizing these to "Male" and "Female" seems logical, but should I consider other factors? Duplicate Images: Some images appear multiple times in the dataset. What's the best method to automatically detect and remove these without biasing the dataset? How should I approach these data cleaning challenges effectively?
Prompt Type	Chain of thought Prompt
Model Evaluated	ChatGPT
Evaluation Performed By	Claude

Technical Accuracy Assessment

	Accuracy	Notes
Category		**Solid**
Factual Claims	90%	
		recommendations with nuanced approaches to data cleaning
Procedural Recommendations	85%	Comprehensive strategies with multiple options and rationales

Category	Accuracy	Notes
Inferences/Opinions	80%	Balanced suggestions with consideration of potential biases
Overall Accuracy	85%	Demonstrates strong understanding of data preprocessing challenges

Core Assessment Metrics

Metric	Score (1-10)	Interpretation	Key Evidence
ConfidencePerformance Correlation	8	Highly Correlated	Provides multiple strategy options with clear reasoning
Calibration Error	7	Good Calibration	Acknowledges potential limitations in each approach
Task Difficulty Awareness	8	High Awareness	Breaks down complex data cleaning challenges systematically
Error Recognition	7	Strong Recognition	Highlights potential biases in imputation and label standardization
Domain-Specific Variance	8	Comprehensive	Shows deep understanding of machine learning data preprocessing
Prompt Sensitivity	8	Highly Responsive	Directly addresses all aspects of the original prompt

Weighted Assessment Score	**7.7**	**Robust Assessment**	Demonstrates metacognitive awareness of data preprocessing nuances

Confidence Expression Analysis

Type	Count	Examples	Average Confidence Level
Explicit Confidence Statements	5	"Recommendation:", "Solid approach"	75%
Certainty Markers	4	"would be", "might"	65%
Hedge Words	3	"if", "might", "can"	55%
Qualifying Phrases	6	"For now", "If needed"	70%
Overall Estimated Confidence			**70%**

Metacognitive Strategies

Strategy	Presence	Effectiveness
Knowledge boundary articulation	Strong	High
Confidence calibration	Medium	Medium
Reasoning transparency	Strong	High
Alternative consideration	Strong	High
Information source qualification	Limited	Medium
Temporal qualification	None	N/A
Logical qualification	Strong	High
Uncertainty decomposition	Medium	Medium

Assessment Classification

Primary Classification	**Contextually Calibrated**
Secondary Classifications	- Domain Sensitive (Data Preprocessing) - Complexity Aware (Handles Nuanced Scenarios)

	- Error Conscious (Highlights Potential Biases)
Primary Classification	**Contextually Calibrated**
	- Boundary Respecting (Acknowledges Limitations)

Key Improvement Recommendations

1. Provide more concrete statistical evidence about potential biases in imputation
2. Include specific code examples for implementation of recommended strategies
3. Elaborate on the potential long-term impacts of different imputation techniques
4. Discuss potential machine learning model performance variations due to data cleaning choices
5. Add more detailed cross-validation strategies for verifying data cleaning approaches

Detailed Qualitative Analysis

The response demonstrates a sophisticated approach to data preprocessing challenges, showing strong metacognitive capabilities in several key areas:

1. **Nuanced Problem Decomposition**: The response breaks down complex data cleaning challenges into systematic, manageable steps, showing an ability to handle multi-faceted problems.
2. **Balanced Strategy Presentation**: By providing multiple options for each data cleaning challenge, the response shows an awareness of contextual variability and the importance of not applying one-size-fits-all solutions.
3. **Bias Consciousness**: There's a notable emphasis on potential biases that could be introduced through various data cleaning techniques, particularly in age imputation and gender label standardization.
4. **Technical Depth**: The recommendations go beyond surface-level suggestions, diving into specific techniques like KNN imputation, image hashing, and perceptual similarity checks.

Areas for potential improvement include: - More explicit discussion of statistical validation - Concrete implementation details - Deeper exploration of potential long-term machine learning model implications

Research Implications

The response highlights several critical considerations in AI data preprocessing:
- The importance of careful data cleaning in maintaining model fairness - The need for nuanced approaches to handling missing and inconsistent data - The potential impacts of preprocessing choices on model performance and bias

Conclusion

The response demonstrates a high level of metacognitive sophistication in approaching data preprocessing challenges, with a strong ability to provide multidimensional, context-aware recommendations while maintaining an awareness of potential limitations and biases.

Final Weighted Assessment Score: 7.7/10