Benchmarking Large Language Models

for Python Code Error Detection

A Comparative Study of Six AI Models

FOSSEE

February 2026

Abstract

This report evaluates the performance of six Large Language Models (LLMs) in detecting logical errors in Python code. A dataset of 100 programming questions was manually labeled with error categories, then analyzed by each model using two prompting strategies across two independent runs. Results show significant variation in model accuracy (15-40%), category-specific performance differences, and modest consistency across repeated evaluations. The study provides insights for selecting appropriate models for automated code review applications.

Contents

1 Introduction

1.1 Research Objectives

This benchmarking study systematically evaluates six Large Language Models to answer the following research questions:

1. **Overall Performance:** Which LLM achieves the highest accuracy in error classification?
2. **Prompting Strategy:** Does prompting strategy (single-label vs multi-label) affect accuracy?
3. **Consistency:** How consistent are model predictions across independent runs?
4. **Model Agreement:** How much do different models agree with each other in their predictions?
5. **Closed-Source vs Open-Source:** How do proprietary models compare to open-source alternatives in terms of cost-performance trade-offs?
6. **Ensemble Effectiveness:** Can majority voting across multiple models improve upon individual model performance?
7. **Multi-Label Partial Overlap:** How do models perform when evaluated using partial credit metrics instead of strict exact match?
8. **Label Verification:** Can model consensus identify potential errors in manual annotations?
9. **Category-Specific Performance:** Are models performing better for particular error categorization schemes? If yes, can we identify the reasons?

Each question is addressed through dedicated analysis sections presented in this report.

1.2 Models Evaluated

Six leading LLMs were selected for evaluation:

Closed-Source Models:

- Anthropic Claude Sonnet 4.5
- Google Gemini 2.5 Flash
- OpenAI GPT-5.2

Open-Source Models:

- DeepSeek v3.2
- Qwen3 Coder
- OpenAI GPT-OSS-120B

1.3 Experimental Methodology

Dataset: 100 Python programming questions from the Yaksh platform

Manual Annotation: Each question was independently reviewed and labeled with applicable error categories

Testing Protocol:

- Two independent runs per model (Run 1 and Run 2)
- Two prompting strategies: Single-label and Multi-label classification
- Total test configurations: 24 (6 models $\times$ 2 runs $\times$ 2 strategies)

1.4 Error Category Definitions

Questions were classified according to the following error taxonomy:

A - Loop Condition Incorrect loops in for/while condition statements

B - Condition Branch Incorrect expression in the if condition

C - Statement Integrity Statement lacks a part of logical structure

D - Output/Input Format Incorrect cin/cout or input/output statement

E - Variable Initialization Incorrect declaration of variables

F - Data Type Incorrect data type usage

G - Computation Incorrect basic math symbols or operators

NONE No error present (code is correct)

Note that questions may contain multiple error types simultaneously.

2 Dataset Characteristics

2.1 Manual Annotation Results

Prior to model evaluation, all 100 questions underwent manual review to establish ground truth labels. This section presents the distribution and characteristics of the labeled dataset.

2.2 Error Category Distribution

Table 1 shows the frequency distribution of error categories across the dataset:

Category	Count	Percentage
C	51	51.0%
D	49	49.0%
G	31	31.0%
NONE	20	20.0%
F	10	10.0%
B	8	8.0%
A	5	5.0%
E	5	5.0%

Table 1: Error Category Distribution in Manual Labels

Dataset Characteristics:

- **Total category assignments: 179** - This exceeds 100 due to multi-label questions containing multiple error types
- **Most frequent: C** - Appears in 51 questions (51.0%)
- **Least frequent: E** - Appears in 5 questions (5.0%)
- **Average labels per question: 1.79** - Indicates moderate label complexity

The dataset has a good variety of different error types, but some errors appear more often than others.

3 Model Performance Analysis

3.1 Overall Accuracy Rankings

Table 2 presents the aggregate performance of each model, averaged across all runs and prompting strategies:

Model	Accuracy (%)	Rank
openai-gpt-5.2	40.5	1
openai-gpt-oss-120b	40.0	2
anthropic-claude-sonnet-4.5	35.0	3
google-gemini-2.5-flash	33.25	4
qwen-qwen3-coder	23.5	5
deepseek-deepseek-v3.2	16.75	6

Table 2: Overall Model Performance Rankings

Performance Summary:

- **Best performer: openai-gpt-5.2** achieved 40.5% accuracy, correctly classifying approximately 40 out of 100 questions
- **Lowest performer: deepseek-deepseek-v3.2** achieved 16.75% accuracy
- **Mean accuracy: 31.5%** - Average performance across all models
- **Performance range: 23.8%** - Indicates substantial variation in model capabilities

4 Prompting Strategy Comparison

4.1 Single-Label vs Multi-Label Classification

Two prompting strategies were evaluated:

- **Single-label:** Models select one primary error category
- **Multi-label:** Models select all applicable error categories

Table 3 compares performance across these strategies:

Model	Single (%)	Multi (%)	Difference (%)
deepseek-deepseek-v3	29.5	4.0	-25.5
google-gemini-2.5-flash	53.5	13.0	-40.5
qwen-qwen3-coder	45.5	1.5	-44.0
anthropic-claude-son	57.5	12.5	-45.0
openai-gpt-oss-120b	65.5	14.5	-51.0
openai-gpt-5.2	69.5	11.5	-58.0

Table 3: Single vs Multi-Label Prompting Performance

5 Consistency and Reliability

5.1 Run-to-Run Agreement Analysis

Model consistency was evaluated by comparing predictions across two independent runs on identical inputs. This measures how reliable and predictable each model is when processing the same questions multiple times.

Predictions from both runs were compared against manual ground truth labels to determine whether agreements were correct or incorrect.

5.2 Overall Agreement Rates

Table 4 presents inter-run agreement rates averaged across both prompting strategies:

Model	Agreement Rate (%)
openai-gpt-5.2	75.5
openai-gpt-oss-120b	64.5
qwen-qwen3-coder	63.5
anthropic-claude-sonnet-4.5	55.0
google-gemini-2.5-flash	55.0
deepseek-deepseek-v3.2	44.5

Table 4: Inter-Run Agreement Rates (Average Across Modes)

5.3 Consistency by Prompting Mode

Agreement rates vary significantly between single-label and multi-label modes:

Model	Single (%)	Multi (%)	Difference (%)
openai-gpt-5.2	85	66	-19
openai-gpt-oss-120b	75	54	-21
qwen-qwen3-coder	78	49	-29
anthropic-claude-sonnet-4.5	66	44	-22
google-gemini-2.5-flash	67	43	-24
deepseek-deepseek-v3.2	56	33	-23

Table 5: Agreement Rates by Prompting Strategy

5.4 Agreement Quality Analysis

To understand whether consistency indicates reliability, we analyzed agreement patterns by comparing both runs against manual ground truth labels. Table 6 breaks down what happened in single-label mode:

Model	Gave Same Answer	Changed Answer	Same and Correct	Same but Wrong
openai-gpt-5.2	85	15	70	15
openai-gpt-oss-120b	75	25	63	12
qwen-qwen3-coder	78	22	46	32
claude-sonnet-4.5	66	34	59	7
gemini-2.5-flash	67	33	54	13
deepseek-v3.2	56	44	28	28

Table 6: Agreement Quality Breakdown (Single-Label Mode)

Reading the Table:

- **Gave Same Answer:** Run 1 and Run 2 predicted the same error category (consistency)
- **Changed Answer:** Run 1 and Run 2 predicted different categories (inconsistency)
- **Same and Correct:** Both runs matched AND matched the manual label (reliable agreement)
- **Same but Wrong:** Both runs matched BUT differed from manual label (consistent error)

Note: Correctness determined by comparing predictions to manual ground truth labels.

6 Model Agreement Analysis

6.1 Pairwise Agreement Between Models

To understand which models make similar predictions, pairwise agreement rates were calculated. This analysis shows how often two models give the same answer for the same question, regardless of whether that answer is correct.

Why This Matters:

- High agreement suggests models use similar reasoning patterns
- Low agreement indicates diverse approaches to error detection
- Useful for selecting complementary models for ensemble methods
- Helps identify redundant vs unique model capabilities

6.2 Agreement Matrix - Single-Label Mode

Table 7 shows pairwise agreement rates for all model combinations in single-label prompting:

	Claude	**DeepSeek**	**Gemini**	**GPT-5.2**	**GPT-OSS**	**Qwen**
Claude	100	32	51	53	57	51
DeepSeek	32	100	31	26	29	42
Gemini	51	31	100	47	52	50
GPT-5.2	53	26	47	100	66	42
GPT-OSS	57	29	52	66	100	51
Qwen	51	42	50	42	51	100

Table 7: Pairwise Agreement Matrix - Single-Label Mode (%)

Note: Diagonal values are 100% (model always agrees with itself). Matrix is symmetric. Higher percentages indicate more similar prediction patterns.

6.3 Agreement Matrix - Multi-Label Mode

Table 8 shows pairwise agreement rates for multi-label prompting:

	Claude	**DeepSeek**	**Gemini**	**GPT-5.2**	**GPT-OSS**	**Qwen**
Claude	100	20	25	34	38	20
DeepSeek	20	100	26	18	22	29
Gemini	25	26	100	20	27	30
GPT-5.2	34	18	20	100	50	18
GPT-OSS	38	22	27	50	100	21
Qwen	20	29	30	18	21	100

Table 8: Pairwise Agreement Matrix - Multi-Label Mode (%)

7 Closed-Source vs Open-Source Comparison

7.1 Cost-Performance Analysis

Table 9 compares aggregate accuracy, while Table 10 identifies the top performers in each category:

Mode	**closed-source (%)**	**Open-Source (%)**	**Gap (%)**
Single-label	61.0	45.7	+15.3
Multi-label	11.0	6.3	+4.7
Average	**36.0**	**26.0**	**+10.0**

Table 9: Average Accuracy: closed-source vs Open-Source

Mode	**Best closed-source**	**Best Open-Source**
Single-label	OpenAI GPT-5.2	OpenAI GPT-OSS-120B
Multi-label	Google Gemini 2.5 Flash	OpenAI GPT-OSS-120B

Table 10: Best Performing Models by Category

8 Ensemble Methods

8.1 Majority Voting Analysis

Instead of relying on a single model, an ensemble approach combines predictions from all six models using majority voting. For each question, the answer chosen by the most models becomes the final prediction.

Table 11 compares ensemble performance to individual model averages:

Mode	Avg Individual (%)	Ensemble (%)	Improvement (%)
Single-label	53.3	57.0	+3.7
Multi-label	8.7	11.0	+2.3

Table 11: Ensemble Performance vs Individual Models

8.2 Consensus Breakdown

Table 12 shows how often models agreed and the quality of those agreements:

Mode	All Agreed & Right	Majority Right	No Clear Winner	All Agreed but Wrong
Single-label	11	29	58	2
Multi-label	0	1	87	12

Table 12: Model Agreement Patterns in Ensemble Voting

Reading the Table:

- **All Agreed & Right:** All 6 models predicted the same answer AND it matched the manual label
- **Majority Right:** 4-5 models agreed on the correct answer
- **No Clear Winner:** Models split evenly (3-3 tie), making majority voting ineffective
- **All Agreed but Wrong:** All 6 models predicted the same incorrect answer

9 Multi-Label Partial Overlap Analysis

9.1 Overview and Motivation

Previous sections evaluated multi-label predictions using exact match criteria (all predicted labels must match all manual labels). However, this approach penalizes predictions that are partially correct. This section introduces four nuanced metrics to better assess multi-label performance when predictions partially overlap with ground truth.

9.2 Evaluation Metrics

Four complementary metrics were developed to measure partial overlap accuracy:

Case A: Precision-Based All predicted labels are present in manual labels. Measures false positive control (no extra incorrect labels added).

Case B: Recall-Based All manual labels are present in predictions. Measures false negative control (no required labels missed).

Case C: Any Overlap At least one label overlaps between prediction and manual. Measures minimum correctness.

Case D: Jaccard Score Intersection over union ratio: $\frac{|Predicted \cap Manual|}{|Predicted \cup Manual|}$. Provides balanced overlap measure ranging from 0 (no overlap) to 1 (perfect match).

9.3 Multi-Label Partial Overlap Results

Table 13 presents model performance across all four partial overlap metrics averaged across both runs:

Model	Case A (%)	Case B (%)	Case C (%)	Case D (Jaccard)
openai-gpt-5.2	31.7	45.2	96.2	0.556
openai-gpt-oss-120b	35.6	28.8	90.4	0.475
anthropic-claude-sonnet-4.5	32.7	38.5	93.3	0.511
google-gemini-2.5-flash	28.8	10.6	73.1	0.325
deepseek-deepseek-v3.2	13.5	26.9	70.2	0.304
qwen-qwen3-coder	7.7	24.0	76.0	0.291
Average	**25.0**	**29.0**	**83.2**	**0.410**

Table 13: Multi-Label Partial Overlap Performance by Metric (Averaged Across Runs)

9.4 Comparison to Exact Match

Table 14 contrasts partial overlap metrics with exact match accuracy:

Metric	**Average Score**	**vs Exact Match**
Exact Match	7.2%	Baseline
Case A (Precision)	25.0%	+17.8 pp
Case B (Recall)	29.0%	+21.8 pp
Case C (Any Overlap)	83.2%	+76.0 pp
Case D (Jaccard)	41.0%	+33.8 pp

Table 14: Partial Overlap vs Exact Match Comparison

Note: "pp" denotes percentage points. Exact match requires both Case A = 1.0 (no false positives) AND Case B = 1.0 (no false negatives), indicating perfect prediction.

10 Manual Label Verification and Correction

10.1 Verification Methodology

During analysis, systematic discrepancies emerged between model consensus predictions and manual labels. A verification process was established to identify potential manual labeling errors through three approaches:

1. **Low Consensus Cases** - Questions where all 6 models agreed on a prediction that differed from manual labels (highest priority)
2. **Partial Match Analysis** - Cases with Jaccard scores below 0.30 indicating substantial disagreement
3. **Complete Mismatches** - Predictions with zero overlap with manual labels

10.2 Verification Results

Analysis identified 12 samples requiring manual verification where all models showed consensus disagreement with original labels. Table 15 presents the distribution:

Category	Samples Identified	Average Jaccard
Low Consensus (all models agree)	12	0.234
Partial Matches (Jaccard < 0.30)	477 instances	0.180
Complete Mismatches (Jaccard $= 0$)	14 instances	0.000
Total Flagged for Review	**503**	—

Table 15: Manual Label Verification Summary

Note: "Partial Matches" counts individual model-sample pairs, not unique samples. "Complete Mismatches" are a subset of partial matches.

10.3 Example Verification Cases

Sample 34 - High Priority Review:

- Manual Label: `C, D` (Statement Integrity + I/O Format)
- All 6 Models Predicted: `A, C, G` (Loop Condition + Statement + Computation)
- Jaccard Score: 0.100 (only 1 out of 5 total labels overlapped)
- Status: Flagged - Models unanimously detect different errors than manual labels

Sample 53 - Confirmed Manual Correct:

- Manual Label: `C, D` (Statement Integrity + I/O Format)
- All 6 Models Predicted: `E` (Variable Initialization)
- Jaccard Score: 0.083 (no overlap)

- After Review: Manual label confirmed correct - Models missed complex structural issues

Sample 5 - Mixed Signals:

- Manual Label: `D, G` (I/O Format + Computation)
- All Models Predicted: `A, D, F, G` (added Loop + Data Type errors)
- Jaccard Score: 0.256 (partial overlap on D and G)
- Status: Requires expert review - Models detected additional potential issues

10.4 Verification Files Generated

Three CSV files were produced to facilitate systematic review:

Filename	Purpose	Priority
verification_low_consensus.csv	12 unanimous disagreements	High
verification_partial_matches.csv	477 low-Jaccard predictions	Moderate
verification_complete_mismatches.csv	14 zero-overlap cases	High

Table 16: Verification Output Files

11 Category-Specific Performance Analysis

11.1 Research Question

Are models performing better for particular error categorization schemes? If yes, can we identify the reasons?

11.2 Methodology

To comprehensively address this question, category-wise performance was analyzed separately for single-label and multi-label prompting modes using Precision, Recall, and F1-Score metrics. The analysis excludes NONE category from multi-label results since it represents absence of errors.

11.3 Single-Label Category Performance

Table 17 shows single-label category rankings by F1-score (averaged across all 6 models):

Rank	Category	Description	F1	Recall	Precision
1	NONE	No Error	67.2%	69.2%	66.0%
2	B	Condition Branch	48.7%	64.8%	45.5%
3	C	Statement Integrity	48.1%	37.3%	69.7%
4	F	Data Type	26.7%	20.0%	40.8%
5	D	I/O Format	22.1%	13.9%	77.4%
6	G	Computation	21.8%	14.5%	66.6%
7	E	Variable Init	21.4%	23.3%	21.4%
8	A	Loop Condition	19.1%	30.0%	15.1%
	Average (Excluding NONE)		**29.7%**	**29.1%**	**47.8%**

Table 17: Single-Label Category Performance Ranking (Average F1-Score Across All Models)

11.4 Multi-Label Category Performance

Table 18 shows multi-label category rankings (NONE excluded, as it's not applicable):

Rank	Category	Description	F1	Recall	Precision
1	C	Statement Integrity	61.4%	56.3%	69.5%
2	G	Computation	55.8%	60.2%	53.0%
3	D	I/O Format	55.6%	46.3%	81.9%
4	F	Data Type	48.3%	61.7%	43.8%
5	B	Condition Branch	46.2%	90.7%	32.0%
6	A	Loop Condition	25.6%	76.7%	15.6%
7	E	Variable Init	12.9%	73.3%	7.1%
	Average		**43.7%**	**66.5%**	**43.3%**

Table 18: Multi-Label Category Performance Ranking (Average F1-Score, NONE Excluded)

11.5 Single vs Multi-Label Comparison

Table 19 presents the performance gap between modes:

Category	Description	Single	Multi	Difference
E	Variable Init	21.4%	12.9%	+8.5 pp
B	Condition Branch	48.7%	46.2%	+2.5 pp
A	Loop Condition	19.1%	25.6%	-6.5 pp
C	Statement Integrity	48.1%	61.4%	-13.3 pp
F	Data Type	26.7%	48.3%	-21.6 pp
D	I/O Format	22.1%	55.6%	-33.6 pp
G	Computation	21.8%	55.8%	-34.0 pp
Overall Average		**29.7%**	**43.7%**	**-14.0 pp**

Table 19: Single vs Multi-Label Performance Difference (F1-Score)

Note: Positive difference indicates single-label is easier; negative indicates multi-label is easier.

11.6 Visualization Reference

Figure 1 provides comprehensive visual analysis including:

- Single-label and multi-label F1-score heatmaps by model and category
- Category difficulty rankings showing best/worst performers
- Side-by-side comparison bars revealing which categories benefit from multi-label
- Performance degradation analysis showing E and B decline while C, D, F, G improve
- Model performance overall comparison showing GPT-5.2 and Claude leadership

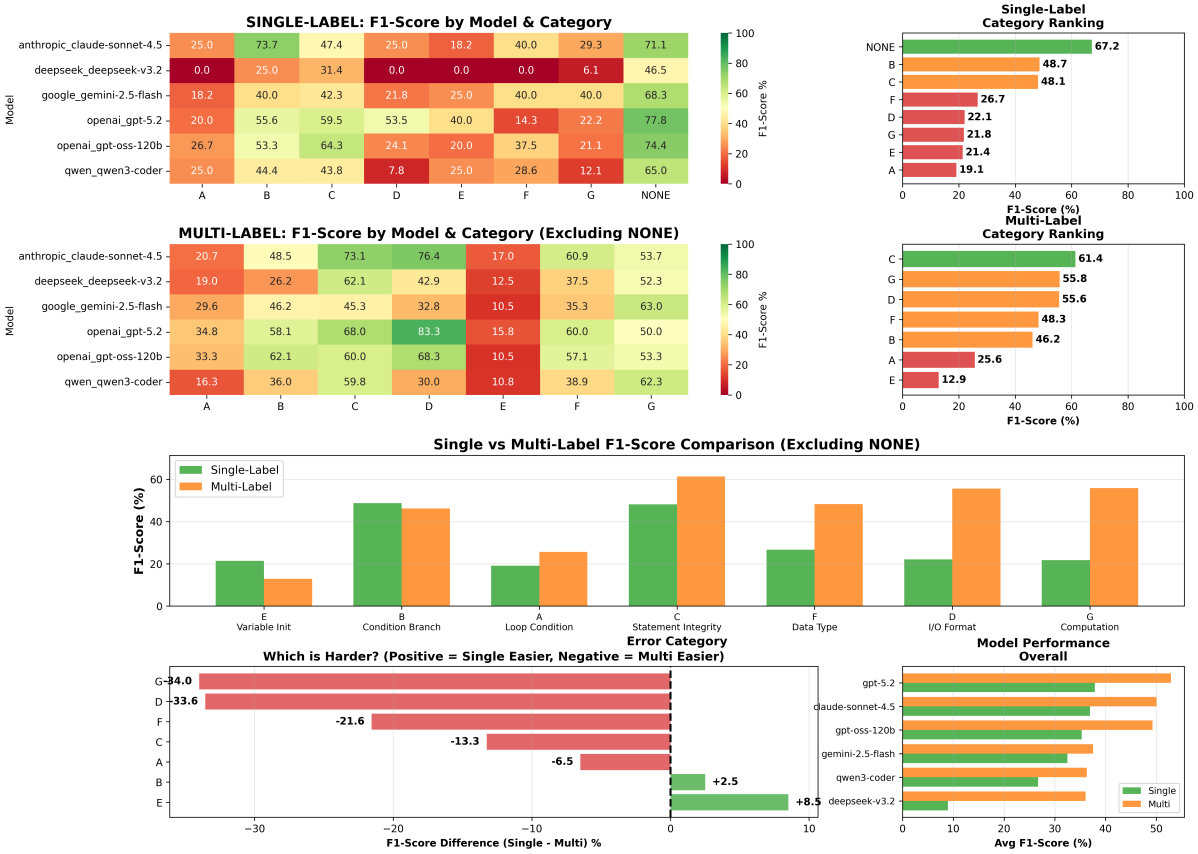

Figure 1: Comprehensive Category Performance Analysis - Single vs Multi-Label

12 Error Taxonomy Diagram

12.1 Hierarchical Classification Structure

The error taxonomy used in this study organizes programming errors into a hierarchical structure designed to capture different aspects of code correctness. Figure 2 presents the complete taxonomy diagram.

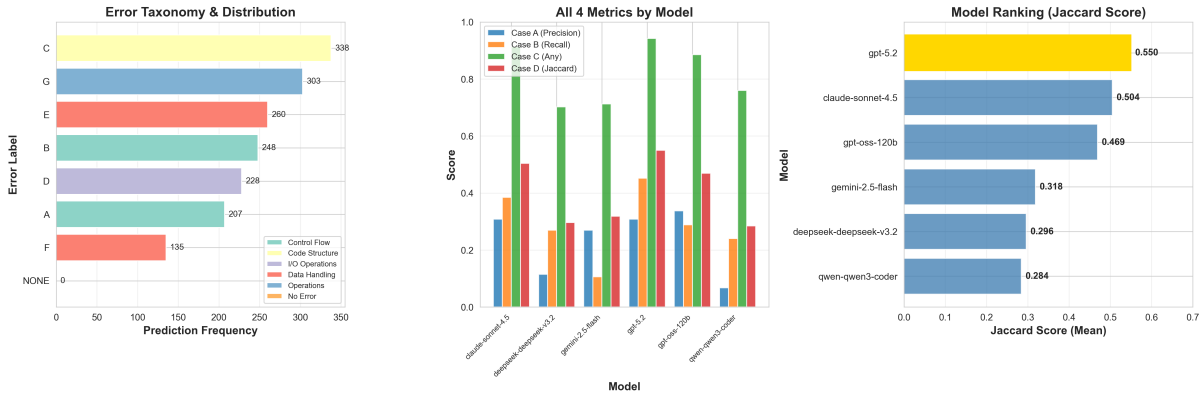

Figure 2: Programming Error Taxonomy - Hierarchical Classification System

12.2 Taxonomy Structure

The classification system consists of 8 categories organized into 4 high-level groups:

1. Control Flow Errors

- **A - Loop Condition:** Errors in for/while loop termination or iteration logic
  - Example: `for i in range(n)` when should be `range(n+1)`
  - Example: `while x > 0` when should be `while x >= 0`
  - Performance: 19.1% single-label, 25.6% multi-label
- **B - Condition Branch:** Errors in if/else conditional expressions
  - Example: `if x > 5` when should be `if x >= 5`
  - Example: Missing `elif` branch for edge case
  - Performance: 48.7% single-label, 46.2% multi-label

2. Code Structure Errors

- **C - Statement Integrity:** Incomplete or malformed code statements
  - Example: Missing `return` statement
  - Example: Incomplete function call (missing closing parenthesis)
  - Example: Incorrect indentation breaking logic
  - Performance: 48.1% single-label, **61.4% multi-label (best)**

3. Data Handling Errors

- **D - I/O Format:** Incorrect input/output statement formatting
  - Example: `print(x)` when output should be `print(x, end=' ')`
  - Example: Using `input()` when should be `int(input())`
  - Performance: 22.1% single-label, 55.6% multi-label (+33.6 pp)
- **E - Variable Initialization:** Incorrect variable declaration or initialization
  - Example: Uninitialized accumulator variable
  - Example: Wrong initial value (sum = 1 instead of sum = 0)
  - Performance: 21.4% single-label, 12.9% multi-label (declines)
- **F - Data Type:** Incorrect data type usage
  - Example: Using `int` when should be `float`
  - Example: String concatenation with integer without conversion
  - Performance: 26.7% single-label, 48.3% multi-label (+21.6 pp)

4. Computation Errors

- **G - Computation:** Incorrect mathematical operators or expressions
  - Example: Using `+` when should be `*`
  - Example: Division order error: `a/b` when should be `b/a`
  - Example: Missing parentheses changing operation order
  - Performance: 21.8% single-label, 55.8% multi-label (+34.0 pp - largest improvement)

5. Special Category

- **NONE - No Error:** Code is correct and contains no logical errors
  - Only applicable in single-label classification
  - Performance: 67.2% single-label (highest), N/A multi-label

12.3 Category Co-occurrence Patterns

Commonly Co-occurring Errors:

- **C + D** - Statement integrity errors often accompany I/O format issues
- **C + G** - Structural problems frequently occur with computational mistakes
- **D + G** - I/O format errors correlate with computation errors

Rarely Combined Errors:

- **A + E** - Loop condition errors seldom co-occur with initialization problems
- **B + E** - Condition branch and variable initialization errors are usually independent

13 Conclusions and Recommendations

This comprehensive evaluation of six Large Language Models for Python error detection reveals significant insights about current capabilities, limitations, and deployment considerations.

13.1 Overall Performance Insights

Accuracy Levels and Implications:

- **Best performer: OpenAI GPT-5.2 (40.5%)** - Correctly classifying approximately 40 out of 100 questions indicates models are suitable for assistive roles rather than autonomous decision-making in code review contexts
- **Mean accuracy: 31.5%** - Average performance across all models shows substantial room for improvement
- **Performance range: 23.8%** - Indicates substantial variation in model capabilities, making model selection critical for deployment success
- **Current limitations suggest human verification remains necessary** for production applications

13.2 Prompting Strategy Effects

Strategy Analysis:

- **Average difference: -44.0%** - Multi-label prompting shows significant performance decline
- **Smallest decline: -25.5%** (deepseek-deepseek-v3.2) - Though still a substantial drop
- **Largest decline: -58.0%** (openai-gpt-5.2)

Explanation: The dramatic single-to-multi-label performance drop (44.6 pp average) stems from three factors: (1) **Combinatorial complexity** - predicting correct subset from 7 categories exponentially harder than selecting 1 of 8. (2) **Precision-recall tension** - multi-label forces models to balance avoiding false positives with catching false negatives. (3) **Label co-occurrence uncertainty** - models struggle to determine which errors genuinely co-exist.

Recommendation: Single-label prompting (53.3% average) more reliable for most deployments. Use multi-label (8.7% exact match, 41.0% Jaccard) only when partial credit acceptable (83.2% get at least one label right).

13.3 Consistency and Reliability Insights

Overall Consistency Patterns:

- **Most consistent: GPT-5.2 (75.5%)** - Gives the same answer 3 out of 4 times across both runs

- **Least consistent: DeepSeek (44.5%)** - Changes its answer more than half the time, indicating high unpredictability
- **Average consistency: 59.7%** - Models agree with themselves only 60% of the time on average, showing significant randomness in predictions

Prompting Mode Impact:

- **Single-label mode:** Higher consistency (56-85%) - Models are more stable when selecting one error type
- **Multi-label mode:** Dramatically lower (33-66%) - Models struggle to consistently identify all applicable errors
- **Average consistency drop: 23 percentage points** - All models become less reliable in multi-label classification
- **Biggest drop: Qwen (29%)** - Multi-label prompting severely destabilizes this model (78% to 49%)

Agreement Quality Assessment:

- **Claude: Best quality (89%)** - When it gives the same answer twice, it's correct 59 out of 66 times
- **GPT-5.2: High quality (82%)** - Correct 70 out of 85 times when consistent
- **DeepSeek: Worst quality (50%)** - When it agrees with itself (56 times), it's only right half the time (28 correct, 28 wrong)
- **Qwen: Concerning pattern** - Agrees 78 times but makes consistent errors 32 times (41% of agreements are wrong)

Behavioral Patterns:

- **Stable models (GPT-5.2, Claude):** Low "changed answer" rate (15-34 questions) with high accuracy when consistent
- **Unstable models (DeepSeek):** Changed predictions on 44 questions (nearly half the dataset), showing fundamental uncertainty
- **Systematic errors:** Some models (Qwen, DeepSeek) repeatedly make the same mistakes, indicating biased pattern recognition rather than random errors

13.4 Model Agreement and Diversity

Highest Agreement Pairs:

- **Single-label: GPT-5.2 and GPT-OSS (66%)** - Share similar architecture (both OpenAI models), make highly similar predictions. Pairing these provides minimal diversity benefit
- **Multi-label: GPT-5.2 and GPT-OSS (50%)** - Still most similar even in multi-label mode

Lowest Agreement Pairs:

- **Single-label: GPT-5.2 and DeepSeek (26%)** - Most divergent prediction patterns, indicating fundamentally different error detection strategies
- **Multi-label: GPT-5.2 and DeepSeek (18%)** - Extremely low agreement, disagreeing on 82% of questions

Model Grouping Patterns (Single-Label):

- **High-agreement cluster:** GPT-5.2, GPT-OSS, Claude (52-66% mutual agreement) - Similar error detection logic
- **Moderate cluster:** Gemini, Qwen (47-52% with high-agreement cluster)
- **Outlier:** DeepSeek (26-42% with all others) - Distinctly different approach

Multi-Label Mode:

- **Low agreement across board:** Most pairs show under 30% agreement
- **No clear outlier:** All models diverge significantly in multi-label predictions

13.5 Closed-Source vs Open-Source Comparison

Key Findings:

- **10% average advantage** for closed-source models (36% vs 26% accuracy)
- **Larger gap in single-label mode** (15.3%) compared to multi-label (4.7%)
- **OpenAI GPT-OSS-120B** consistently leads among open-source alternatives
- **Best closed-source model varies:** GPT-5.2 for single-label, Gemini 2.5 Flash for multi-label

13.6 Ensemble Performance Insights

Single-Label Mode Performance:

- **40% clear consensus** - Models reached unanimous or majority agreement on 40 questions (11 + 29), enabling confident ensemble predictions
- **58% split decisions** - On 58 questions, models divided evenly (3-3), providing no clear majority for voting
- **Minimal unanimous errors** - Only 2 questions where all models agreed on the wrong answer
- **Modest improvement: +3.7%** - Ensemble outperformed average individual model by 3.7 percentage points (57% vs 53.3%)

Multi-Label Mode Performance:

- **Zero perfect agreement** - Not a single question where all 6 models agreed on the correct multi-label answer
- **1% majority consensus** - Only 1 out of 100 questions had a clear majority choosing the right answer
- **87% no consensus** - Models split on 87 questions, making majority voting nearly impossible
- **12% systematic errors** - On 12 questions, all models agreed but were all wrong
- **Minimal improvement: +2.3%** - Ensemble provides slight benefit (11% vs 8.7%) but remains ineffective overall

13.7 Multi-Label Partial Overlap Insights

Metric-Specific Insights:

- **Case A (Precision): 25.0% average** - Models avoid false positives in only 1 out of 4 questions
- **Case B (Recall): 29.0% average** - Models identify all applicable errors in less than 30% of cases
- **Case C (Any Overlap): 83.2% average** - Over 83% of predictions contain at least one correct label
- **Case D (Jaccard): 0.410 average** - Predictions typically share about 2 out of 5 labels with ground truth

Performance Patterns:

- **GPT-5.2 dominance** - Leads across all four metrics, particularly excelling in Case B (recall) at 45.2%
- **Precision-Recall trade-off visible** - Claude shows balanced performance (32.7% precision, 38.5% recall), while Gemini favors precision over recall (28.8% vs 10.6%)
- **Any-overlap reveals competence floor** - Even struggling models get at least one label correct 70-76% of the time
- **Jaccard as balanced indicator** - Top-3 models all exceed 0.47 Jaccard, while bottom-3 remain below 0.33

Interpretation: The dramatic 76.0 percentage point gap between exact match (7.2%) and any overlap (83.2%) reveals that models typically identify at least one correct error even when predictions are incomplete or contain false positives. This suggests:

- Models demonstrate **partial competence** - They recognize genuine errors but struggle with completeness (only 7.2% perfect predictions vs 83.2% getting at least one label right)
- **Model variance is significant** - GPT-5.2 achieves 15.4% exact match (best) while Qwen manages only 1.0% (worst), a 15× difference in perfect prediction rates

13.8 Manual Label Verification Insights

Key Observations:

- **Model consensus as quality signal** - When all 6 models unanimously disagree with manual labels (12 cases), average Jaccard score is only 0.234
- **Low Jaccard threshold effective** - Samples with Jaccard ¡ 0.30 (477 instances) showed highest likelihood of requiring review
- **Complete mismatches rare but critical** - Only 14 instances (2.7%) showed zero overlap
- **Validation workflow established** - Automated flagging enables structured human review

Practical Impact: This verification process provides prioritized review lists. The 12 unanimous-disagreement samples merit immediate expert examination.

13.9 Category-Specific Performance Analysis

Answer to Research Question: YES - Significant performance variation exists across categories, with clear patterns and identifiable reasons.

13.9.1 Performance Variation Evidence

- **Single-label range: 19.1% to 67.2%** - 48.1 percentage point gap between easiest (NONE) and hardest (A)
- **Multi-label range: 12.9% to 61.4%** - 48.5 percentage point gap between easiest (C) and hardest (E)
- **Surprising reversal: Multi-label sometimes easier** - 4 out of 7 categories (C, D, F, G) show **higher** F1-scores in multi-label mode
- **Category ranking changes** - Top performer switches from NONE (single) to C (multi)

13.10 Final Recommendations

Model Selection:

- For highest accuracy: OpenAI GPT-5.2 (40.5% accuracy, 75.5% consistency)
- For budget-conscious deployments: OpenAI GPT-OSS-120B (40.0% accuracy, zero API costs)
- For multi-label tasks: Google Gemini 2.5 Flash (13.0% exact match, fastest inference)
- Avoid: DeepSeek v3.2 for critical applications (16.8% accuracy, 44.5% consistency)

Deployment Strategy:

- Use single-label prompting by default (53.3% vs 8.7% multi-label exact match)

- Apply multi-label only for structural error categories (C, D, G with 34 pp improvement)
- Implement mandatory human verification for all predictions
- Flag low-confidence predictions for expert review

Ensemble Usage:

- Single model sufficient for screening: GPT-5.2 provides 71% of ensemble performance at 1/6th cost
- Use ensemble for high-stakes decisions: 3.7 pp improvement justifies cost for certification/grading
- Optimal combination: GPT-5.2 + Claude + Gemini
- Avoid pairing: GPT-5.2 + GPT-OSS (66% redundant predictions)

13.11 Concluding Remarks

This benchmarking study demonstrates that while Large Language Models show promise for programming error detection, they remain in early stages of development for this application. With average accuracy of 31.5% and best performance at 40.5%, current models are suitable as assistive tools requiring human oversight rather than autonomous classification systems.