A. Structural Features

Ye code ke structure se related hote hain:

1. **Total number of words** – space se split karke count karna.
2. **Total number of characters** – including whitespaces ya excluding, dono alag features ban sakte hain.
3. **Longest word length** – code me jo sabse lamba token/word hai uska length.
4. **Average line length** – total characters / total lines.
5. **Number of lines** – total line count.
6. **Indentation style** – spaces vs tabs ratio.
7. **Average indentation depth** – har line ka indentation level ka average.
8. **Blank line ratio** – blank lines / total lines.

B. Token-based Features

Yaha hume lexical tokens ka analysis karna hai:

9. **Token frequency distribution** – keywords, identifiers, literals, operators, symbols ka count.
10. **Ratio of keywords to identifiers** – AI code me zyada structured keyword usage hota hai.
11. **Average token length** – har token ka average size.
12. **Identifier length distribution** – variable/method names kitne lamba/chhota hai.
13. **CamelCase vs snake_case ratio** – AI code me uniform style hota hai.
14. **Unique identifier count** – kitne unique variables/functions banaye gaye hain.

C. Complexity Features

AI aur human code ka complexity style alag hota hai:

15. **Cyclomatic complexity** – number of independent paths in the code (McCabe's complexity).
16. **Nesting depth** – loops/if ke andar kitna nested structure hai.
17. **Average branching factor** – kitne `if/else/switch` branches ka average.
18. **Number of comments** – AI me comments ka distribution alag hota hai.
19. **Comment density** – comments ka percentage code ke size ke hisaab se.

D. Statistical & Stylistic Features

20. **Whitespace ratio** – spaces / total characters.
21. **Punctuation frequency** – `; , . () {}` ka usage pattern.
22. **Operator density** – operators per line.
23. **Literal density** – numeric & string literals ka ratio.
24. **Repetition score** – kitne patterns repeat ho rahe hain (AI me repetition zyada hota hai).
25. **Entropy** – randomness of characters/tokens (AI code me usually low entropy hota hai).

E. Language-specific Features

26. **API call variety** – kitne alag library/API calls use hue.
27. **Import/Include diversity** – AI me imports ka pattern consistent hota hai.
28. **Unused variables/functions ratio** – AI me kam hota hai, human me zyada.
29. **Error handling ratio** – `try/catch` ka percentage.
30. **Naming meaningfulness score** – NLP se check karna ki variable names meaningful hain ya random.

I. Lexical & Token-Level Features

Features extracted from the **raw text or token sequence** of code.

Feature Name	Definition	Why It Matters	Measurement / Example
Total Tokens	Number of lexical tokens after tokenization	AI often produces consistent token lengths, human code varies	Tokenize using language grammar (e.g., `Pygments`) and count
Total Characters	Total characters in the code	Basic size metric	`len(code)`
Average Token Length	Mean length of tokens	AI tends to have shorter identifiers	`sum(len(t) for t in tokens) / len(tokens)`
Longest Token Length	Maximum length of any token	Humans sometimes use very descriptive identifiers	`max(len(t) for t in tokens)`
Keyword-to-Identifier Ratio	Count of language keywords ÷ count of variable/function names	AI often uses more keywords than necessary	Count from language keyword set
Special Character Frequency	Frequency of `{}`, `;`, `()`, `.`	Style indicator	Count occurrences per 100 tokens
Digit Ratio in Identifiers	Fraction of identifiers containing digits	AI may avoid or overuse digits in names (`var1`, `temp2`)	`(digit_identifiers / total_identifiers)`
Case Style Distribution	camelCase, snake_case, PascalCase ratios	AI sometimes prefers snake_case due to training data	Regex check
Identifier Naming Entropy	Shannon entropy of identifier names	Lower entropy = more repetitive naming	Use entropy formula on identifier strings
Comment-to-Code Ratio	Comments ÷ code lines	AI may produce overly generic comments or none	Count `//`, `#`, `/* ... */` occurrences
Comment Density Variance	Variance in where comments appear	Humans place comments in context; AI places them uniformly	Compare line numbers of comments

II. Formatting & Layout Features

Features about **how the code looks**.

Feature Name	Definition	Why It Matters	Measurement / Example
Indentation Depth Mean	Average indentation level	AI keeps it uniform, humans vary	Count spaces/tabs per line
Indentation Depth Variance	Variability in indentation	Low variance → possibly AI	Compute variance
Average Line Length	Mean characters per line	AI often produces uniform line lengths	`sum(len(line) for line in lines)/len(lines)`
Line Length Variance	Variability in line length	Humans have higher variance	`variance([len(l) for l in lines])`
Blank Line Ratio	Blank lines ÷ total lines	AI may insert blanks systematically	Count empty lines
Brace Placement Style Ratio	Ratio of { on same line vs next line	Can indicate coding style bias from AI training data	Pattern matching
Trailing Whitespace Frequency	% of lines ending with spaces/tabs	Humans more likely to leave accidental spaces	Count

III. Syntactic (AST-Based) Features

Derived from **parsing** the code into an Abstract Syntax Tree (AST).

Feature Name	Definition	Why It Matters	Measurement / Example
AST Depth	Max depth of parse tree	AI may produce simpler or overly deep nesting	Parse with `tree-sitter` or `ast`
Average Nesting Depth	Mean depth of loops/conditionals	Humans nest deeper for corner cases	Walk AST
Cyclomatic Complexity	Number of independent execution paths	AI sometimes writes more direct code	Use `radon` or manual formula
Number of Functions	Total function definitions	AI may modularize too much or too little	Count AST nodes of type `FunctionDef`
Function Length Mean & Variance	Avg & variance of LOC per function	Humans vary more in function size	Count lines per function
Number of Parameters per Function	Mean params per function	AI tends to stick to 2–3	Extract from AST
Loop Type Ratio	For/While ratio	Language usage preference from training data	Count loop types

Conditional Density	Conditionals ÷ total lines	AI may use more straightforward conditions	Count `if`, `switch`
Try/Except or Try/Catch Usage	Error handling style	AI tends to use generic blocks	AST search

IV. Semantic / Behavioral Features

Language-aware, **meaning-related** metrics.

Feature Name	Definition	Why It Matters	Measurement / Example
Library Usage Diversity	Unique library imports ÷ total imports	Humans use domain-specific libs	Count unique import statements
API Call Frequency	Calls to external APIs	AI uses standard APIs more often	Search function calls not in local scope
Algorithm Pattern Match	Whether common textbook algorithms appear	AI tends to pick canonical versions	Pattern matching on known algorithm shapes
Error Handling Granularity	Whether exceptions are specific	Humans specify exception types	Look for generic vs specific catch clauses
Hardcoded Values Ratio	Hardcoded numbers/strings per LOC	AI may use magic numbers	Search for numeric literals
Redundancy Score	Similar code blocks repeated	AI sometimes repeats snippets	Token-based similarity check

V. Statistical & Entropy-Based Features

Measure **predictability/randomness**.

Feature Name	Definition	Why It Matters	Measurement / Example
Token Entropy	Shannon entropy over token sequence	AI has lower token entropy	Apply entropy formula on token list
Character Entropy	Same but for characters	Lower entropy = more predictable	Apply entropy on string
Burstiness	Variation in length between adjacent lines	Human code less uniform	Calculate standard deviation of line length diffs
Repetition Ratio	Frequency of repeated n-grams	AI more repetitive	Count repeated token sequences
AST Subtree Reuse Rate	Repeated syntax tree patterns	AI may reuse identical AST branches	Compare AST hashes