Software Metrics or features:

- A software metric is a measure of software characteristics that are quantifiable or countable. Software metrics are important for many reasons, including measuring software **performance** etc. Some of the important metrics are: Static code measures, McCabe metrics, Halstead features.

Halstead's Features:

A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands. **Halstead's metrics** are included in a number of current commercial tools that count software lines of code.

Halstead argued that code that is hard to read is more likely to be fault prone. Halstead estimates reading complexity by counting the number of concepts in a module; e.g. number of unique operators.

The Halstead falls into three groups: the base measures, the derived measures, and lines of code measures.

-- Base measures:
-- mu1 = number of unique operators
-- mu2 = number of unique operands
-- N1 = total occurrences of operators
-- N2 = total occurrences of operands
-- length = N = N1 + N2
-- vocabulary = mu = mu1 + mu2
-- Constants set for each function:
-- mu1' = 2 = potential operator count (just the function name and the "return" operator)
-- mu2' = potential operand count. (the number of arguments to the module)

For example, the expression "return max(w+x,x+y)" has "N1=4" operators "return, max, +,+)", "N2=4" operands (w,x,x,y), "mu1=3" unique operators (return, max,+), and "mu2=3" unique operands (w,x,y).

-- Derived measures:
-- P = volume = V = N * log2(mu) (the number of mental comparisons needed to write a program of length N)
-- V* = volume on minimal implementation
= (2 + mu2')*log2(2 + mu2')
-- L = program length = V*/N
-- D = difficulty = 1/L
-- L' = 1/D
-- I = intelligence = L'*V'
-- E = effort to write program = V/L
-- T = time to write program = E/18 seconds

McCabe Metrics: suggested that code with complicated pathways are more error-prone. His metrics therefore reflect the pathways within a code module.

Cyclomatic Complexity, or "v(G)", measures the number of "linearly independent paths". A set of paths is said to be linearly independent if no path in the set is a linear combination of any other paths in the set through a program's "flowgraph". A flowgraph is a directed graph where each node corresponds to a program statement, and each arc indicates the flow of control from one statement to another. "v(G)" is calculated by "v(G) = e - n + 2" where "G" is a program's flowgraph, "e" is the number of arcs in the flowgraph, and "n" is the number of nodes in the flowgraph. The standard McCabes rules ("v(G)">10), are used to identify fault-prone module.

Complexity Number	Meaning
1-10	Structured and well written code High Testability Cost and Effort is less
10-20	Complex Code Medium Testability Cost and effort is Medium
20-40	Very complex Code Low Testability Cost and Effort are high
>40	Not at all testable Very high Cost and Effort

Software Maintainability Index:

The Software Maintainability Index (MI) is a single-value indicator for the maintainability of a software system.

The Maintainability Index is computed by combining four traditional metrics: It is a weighted composition of the average Halstead Volume per module, the Cyclomatic Complexity, the number of lines of code (LOC) and the comment ratio of the system.

$$171 - 5.2 \cdot \ln(avgHV) - 0.23 \cdot \text{avgCC}(g') - 16.2 \cdot \ln(avgLOC) + 50 \cdot \sin(\text{sqrt}(2.4 \cdot perCM))$$

HV:	Halstead Volume	CC:	Cyclomatic Complexity
LOC:	lines of code	perCM:	% Comment Lines