Alright — let’s fully break this down from the **mathematical & scientific perspective** so you can clearly explain the *"why"* and *"how"* behind this dataset and its features.  
I’ll go step-by-step: **dataset design, feature mathematics, label logic, and theoretical background**.

## **1. Scientific Purpose of the Dataset**

This dataset is intended for **predictive software quality analysis** — the goal is to forecast whether a codebase (or module) will have **High, Medium, or Low quality** based on measurable software metrics.

The **mathematics** behind this dataset comes from **software engineering metrics theory**, mainly:

* **McCabe’s Cyclomatic Complexity theory** for logic branches
* **Code Churn theory** for code evolution & maintainability
* **Coupling measurement theory** for dependency risks
* **Defect density & bug tracking mathematics** for quality assessment
* **Weighted scoring systems** for classification into quality categories

## **2. Mathematical Meaning of Each Feature**

### **(a) LOC – Lines of Code**

* **Definition:** Total number of physical lines of source code in a module/class.
* **Mathematical Relation to Quality:**  
  Higher LOC often → Higher complexity & maintenance cost (but too low LOC might mean underdeveloped functionality).
* **Typical formula for productivity impact:**

where

### **(b) Cyclomatic Complexity (CC)**

* **Definition:** McCabe's metric for measuring independent paths in a program’s control flow graph.
* **Formula:**

where:

* = Number of edges in the control flow graph
* = Number of nodes
* = Number of connected components (usually 1 for a single program)
* **Interpretation:**
  + CC ≤ 10 → Simple
  + 10 < CC ≤ 20 → Moderate risk
  + CC > 20 → Complex, high maintenance risk

### **(c) Code Churn**

* **Definition:** Number of lines of code added, modified, or deleted over a given time period.
* **Formula:**
* **Impact on Quality:**  
  High churn → unstable codebase → higher bug introduction probability.

### **(d) Coupling**

* **Definition:** Degree of interdependence between software modules/classes.
* **Mathematical Representation:**  
  Often measured as **CBO (Coupling Between Objects)**:
* **Impact:**  
  Higher coupling → More ripple effects from changes → Lower maintainability & higher defect risk.

### **(e) Bugs\_Reported**

* **Definition:** Number of confirmed defects found during testing or production.
* **Defect Density Formula:**
* **Impact:**  
  More bugs → Lower perceived quality.

## **3. Label Logic – Weighted Scoring**

The **Quality\_Label** is assigned via a **weighted scoring function** that integrates all the above metrics.

Example formula:

Weights () are determined experimentally.

Classification:

* **High Quality:** Score ≥ threshold
* **Medium Quality:** threshold ≤ Score < threshold
* **Low Quality:** Score < threshold

## **4. Mathematical Science Behind the Model**

When you feed this dataset to a machine learning model (e.g., Decision Tree, Random Forest, Logistic Regression), the mathematics involves:

* **Feature space:**
* **Label space:**
* **Goal:** Learn the mapping

For example, **Logistic Regression** would model:

## **5. Why This Dataset is Balanced**

A balanced dataset ensures each quality class (High/Medium/Low) has **equal representation**, reducing bias in classification. Mathematically:

This prevents skewed decision boundaries.

If you want, I can now **write a full “Dataset Review & Mathematical Foundation” section** for your project/research paper with **proper academic tone**, ready to paste.  
It will include **feature theory, equations, and justification** exactly like above, but in a formal, journal-style structure.

Do you want me to prepare that? It will make your dataset explanation sound *research-grade*.