1. Dataset Overview

This dataset simulates the internal characteristics of 50 software modules or projects. Each sample is defined by commonly used **software metrics**, and its quality is labeled based on a weighted logic formula that reflects real-world quality considerations.

2. Features in the Dataset

Feature Name	Description
`LOC`	Lines of Code (range: 500–10,000). Larger size often means more maintenance cost and complexity.
`Cyclomatic_Complexity`	Indicates decision paths in code (range: 3–20). Higher value = harder to test and maintain.
`Code_Churn`	Frequency of code changes (range: 10–200). High churn can indicate instability.
`Coupling`	Inter-module dependency (range: 1–9). High coupling decreases modularity.
`Bugs_Reported`	Total bugs reported (range: 0–24). Directly reflects software defects.

3. Target Variable

Column Name	Description
`Quality_Label`	Assigned as `High`, `Medium`, or `Low` based on an aggregate quality score computed from the above features.

4. Logic Behind Label Assignment

Each sample gets a **quality score** using the following weighted formula:

```
makefile
CopyEdit
Quality_Score =
    0.25 × (1 - LOC / 10000) +
    0.20 × (1 - Cyclomatic_Complexity / 20) +
    0.20 × (1 - Code_Churn / 200) +
    0.15 × (1 - Coupling / 10) +
    0.20 × (1 - Bugs_Reported / 25)
```

Then the labels are assigned:

- **High** → if score > 0.7
- **Medium** → if 0.4 < score ≤ 0.7
- **Low** → if score ≤ 0.4

5. Why This Dataset is Useful

- **Mimics real-world software quality scenarios**
- Allows comparison of ML models like Decision Tree, Random Forest, SVM, etc.
- Fully labeled and clean — ideal for training and testing classification models
- Scalable for larger experiments
- Useful in academic and industrial research for defect prediction or quality estimation

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

Each row includes:

- LOC (Lines of Code)
- Cyclomatic_Complexity (Code logic branches)
- Code_Churn (Code changes over time)
- Coupling (Interdependence between modules)
- Bugs_Reported (Reported defects)
- Quality_Label: {High, Medium, Low} based on a weighted scoring system

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:**

$$\text{Defect Density} = \frac{\text{Bugs Reported}}{\text{KLOC}}$$

where $\text{KLOC} = \frac{\text{LOC}}{1000}$

(b) Cyclomatic Complexity (CC)

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

$$CC = E - N + 2P$$

where:

- E = Number of edges in the control flow graph
- N = Number of nodes
- P = 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:**

$$\text{Churn Rate} = \frac{\text{LOC Added} + \text{LOC Modified} + \text{LOC Deleted}}{\text{Total LOC}}$$

- **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)**:

$$\text{CBO} = \text{Number of distinct classes/modules referenced}$$

- **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:**

$$\text{Defect Density} = \frac{\text{Bugs Reported}}{\text{KLOC}}$$

- **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:

$$\text{Score} = w_1 \cdot \frac{1}{\text{LOC}} + w_2 \cdot \frac{1}{CC} + w_3 \cdot \frac{1}{\text{Churn Rate}} + w_4 \cdot \frac{1}{\text{Coupling}} + w_5 \cdot \frac{1}{\text{Bugs Reported}}$$

Weights $(w_1, w_2, \ldots, w_5)$ are determined experimentally.

Classification:

- **High Quality:** Score ≥ threshold_H
- **Medium Quality:** threshold_L ≤ Score < threshold_H
- **Low Quality:** Score < threshold_L

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:**

  $X = \{LOC, CC, Churn, Coupling, Bugs\}$

- **Label space:**

  $Y \in \{\text{High,Medium,Low}\}$

- **Goal:** Learn the mapping $f: X \to Y$

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

$$P(\text{High Quality} \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot LOC + \cdots + \beta_5 \cdot Bugs)}}$$

5. Why This Dataset is Balanced

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

$$P(\text{High}) \approx P(\text{Medium}) \approx P(\text{Low}) \approx \frac{1}{3}$$

This prevents skewed decision boundaries.

Machine Learning Model:

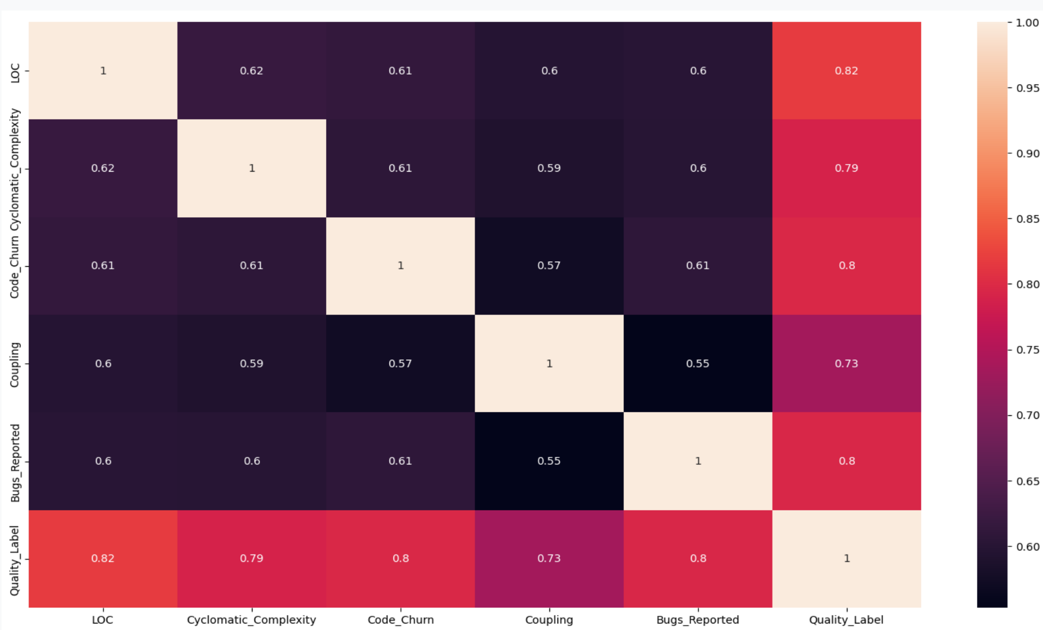
Data set info:

	LOC	Cyclomatic_Complexity	Code_Churn	Coupling	Bugs_Reported	Quality_Label
0	2324	3	199	5	7	Medium
1	4157	7	198	2	21	Medium
2	9435	5	161	7	1	Medium
3	988	5	65	4	16	Medium
4	934	20	60	9	13	Medium

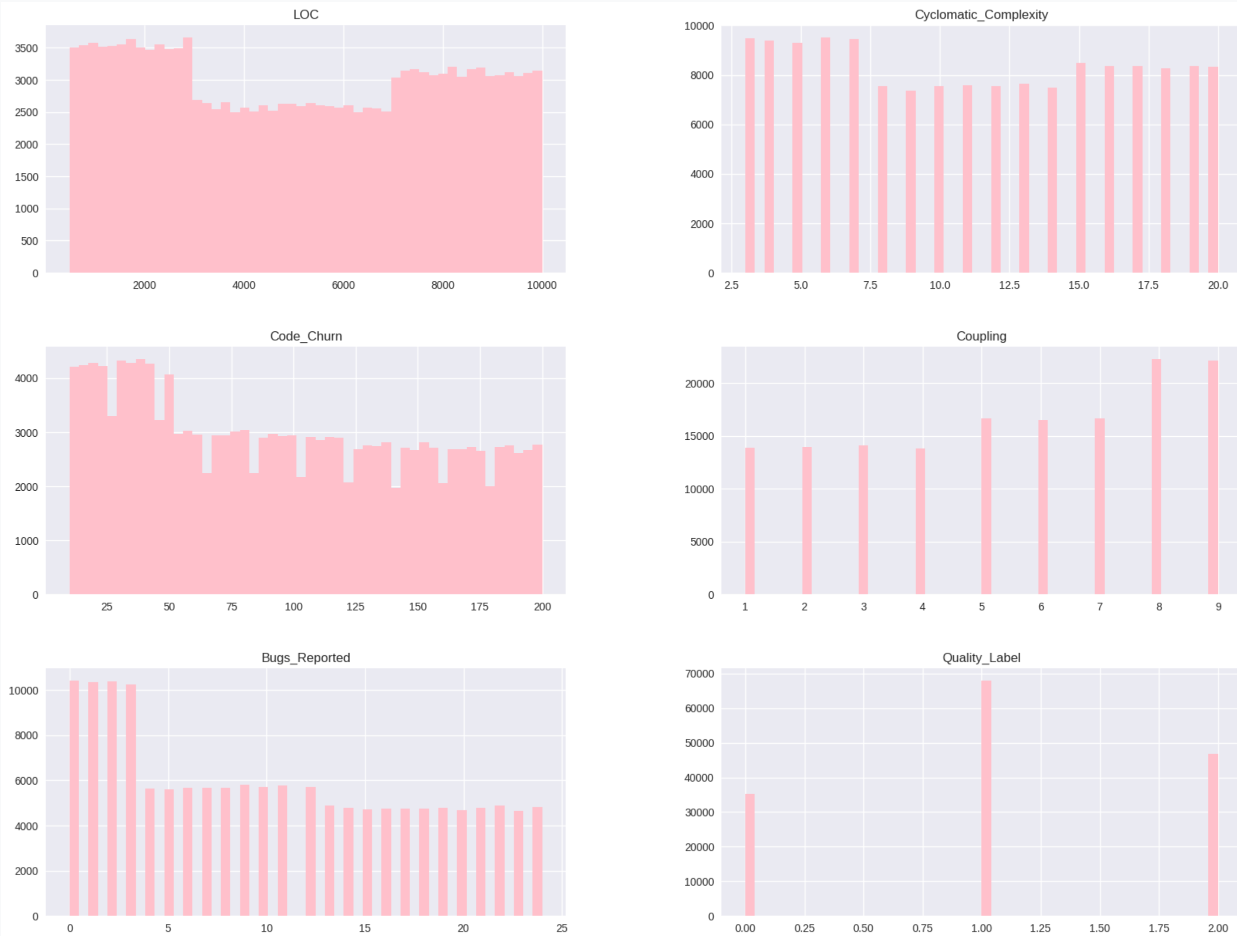
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 6 columns):
 #   Column                 Non-Null Count   Dtype 
---  ------                 --------------   ----- 
 0   LOC                    150000 non-null  int64 
 1   Cyclomatic_Complexity  150000 non-null  int64 
 2   Code_Churn             150000 non-null  int64 
 3   Coupling               150000 non-null  int64 
 4   Bugs_Reported          150000 non-null  int64 
 5   Quality_Label          150000 non-null  object
dtypes: int64(5), object(1)
memory usage: 6.9+ MB
```

```
Quality_Label
Medium    67991
Low       46806
High      35203
Name: count, dtype: int64
```

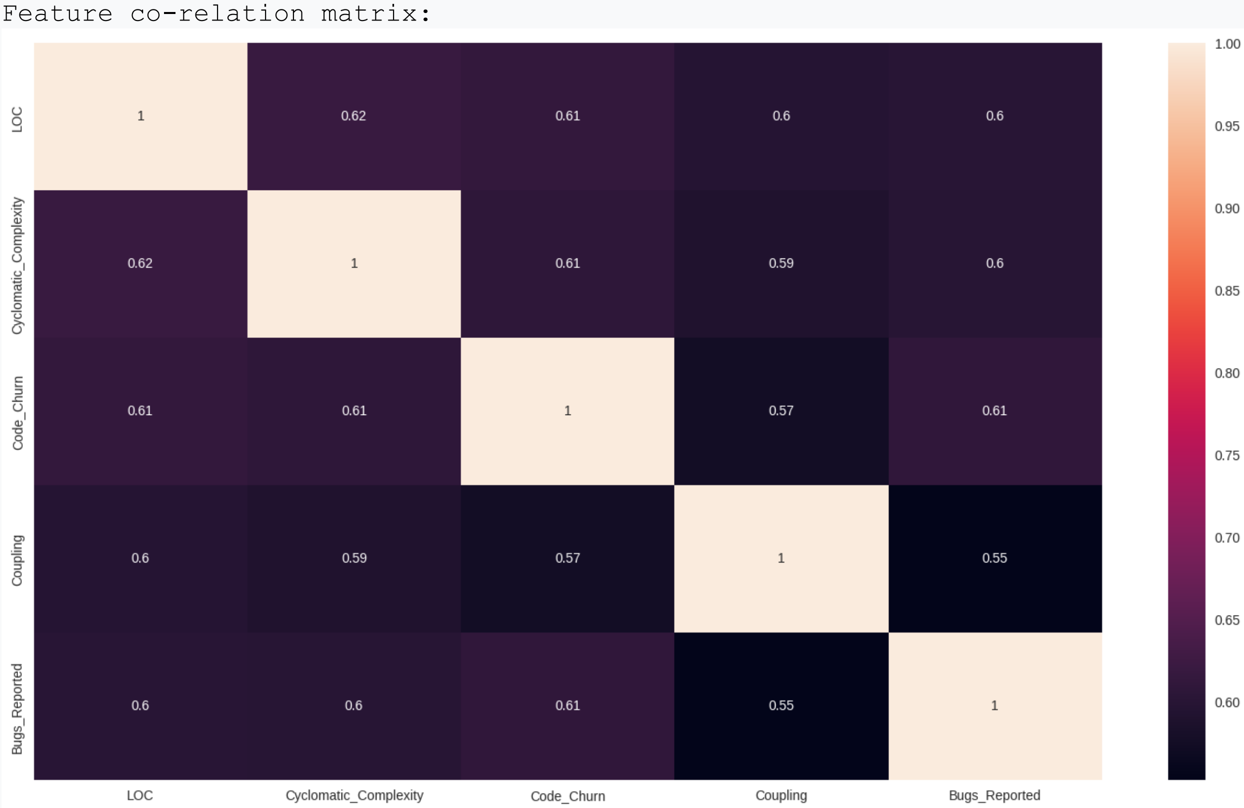
Correlation matrix:



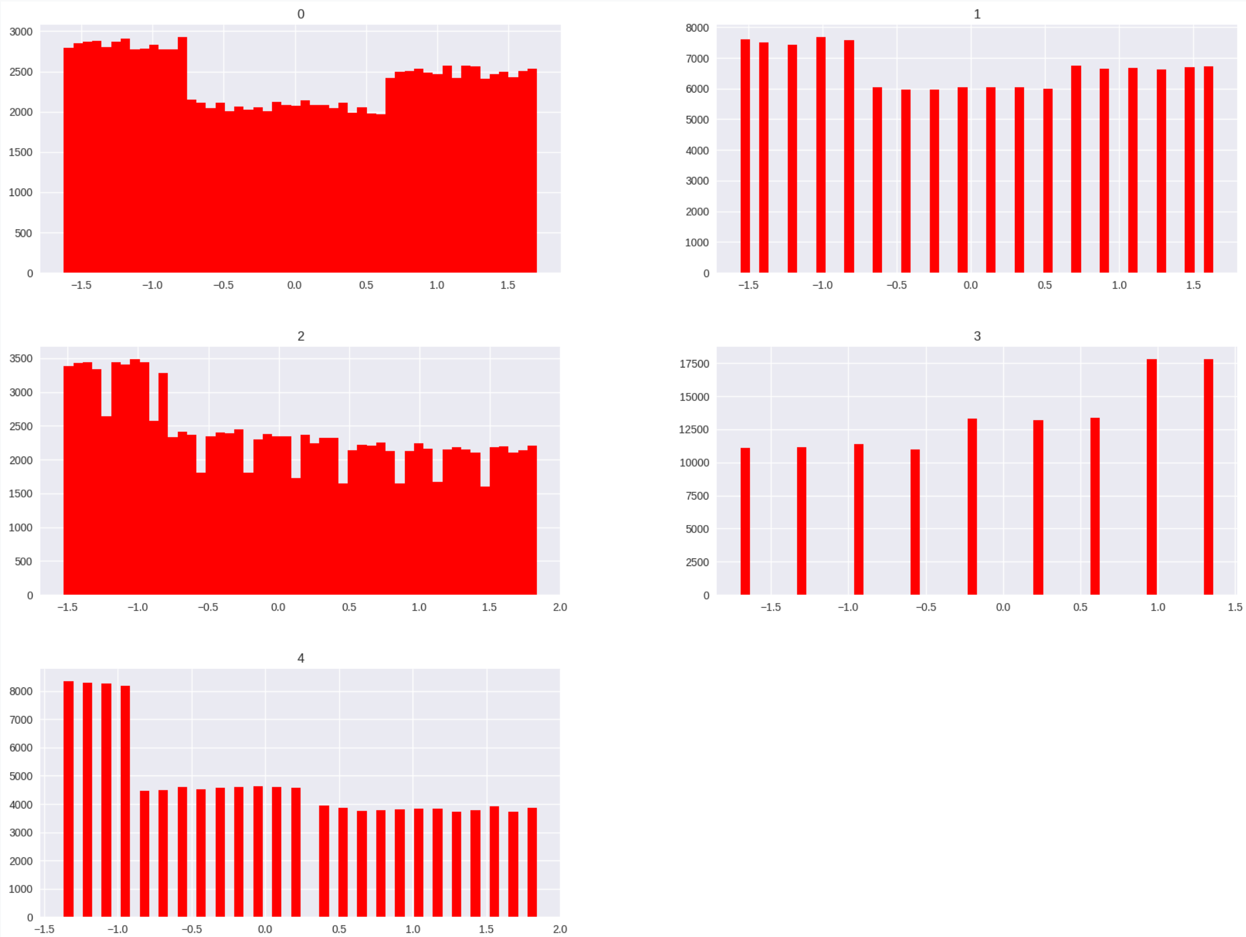
Distributions:



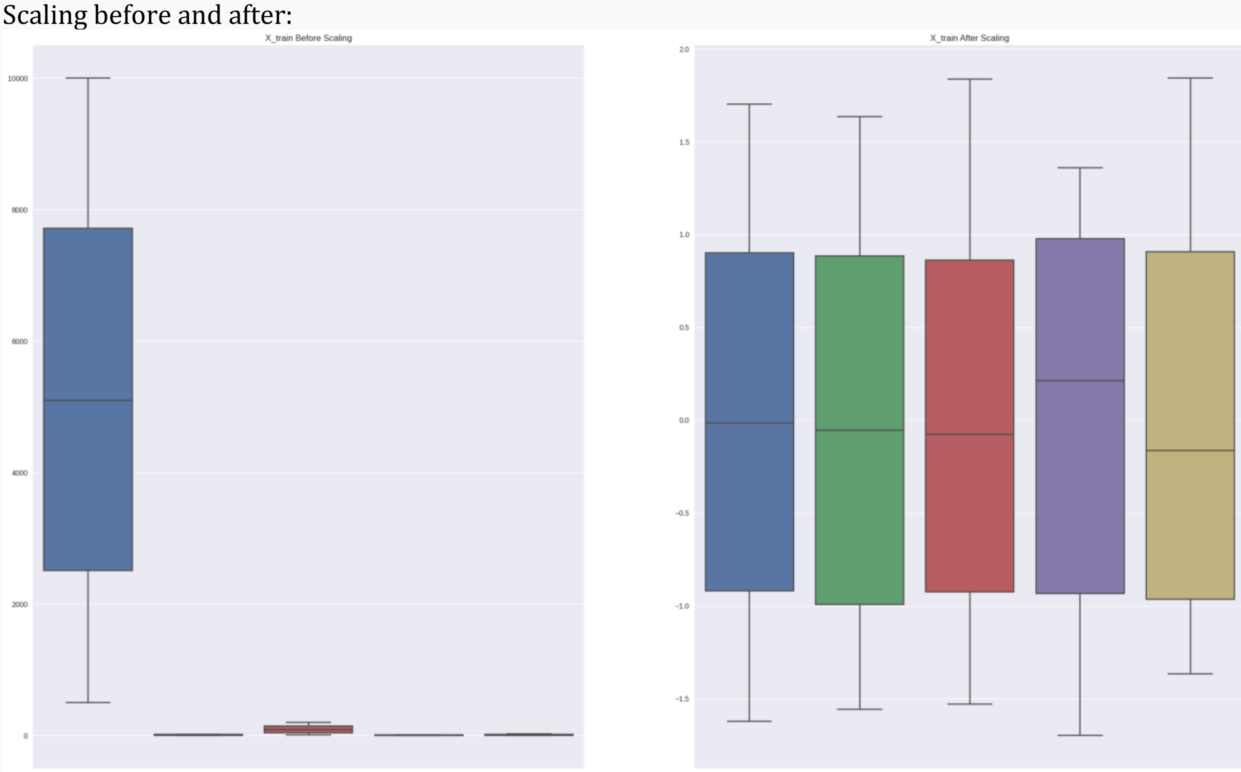
Feature co-relation matrix:



Final Feature Distribution:



Scaling before and after:



Results:

Logisitic Regression
Model performance for Training set
- Accuracy: 0.9985
- F1 Score: 0.9985
- Precision: 0.9985
- Recall: 0.9985
- ROC AUC Score: 1.0000

Model performance for Test set
- Accuracy: 0.9983
- F1 Score: 0.9983
- Precision: 0.9983
- Recall: 0.9983
- ROC AUC Score: 1.0000
===================================

knn
Model performance for Training set
- Accuracy: 0.9933
- F1 Score: 0.9933
- Precision: 0.9933
- Recall: 0.9933
- ROC AUC Score: 0.9999

Model performance for Test set
- Accuracy: 0.9890
- F1 Score: 0.9890
- Precision: 0.9890
- Recall: 0.9890
- ROC AUC Score: 0.9993
===================================

NB
Model performance for Training set
- Accuracy: 0.9134
- F1 Score: 0.9132
- Precision: 0.9132
- Recall: 0.9134
- ROC AUC Score: 0.9767

Model performance for Test set
- Accuracy: 0.9109
- F1 Score: 0.9106
- Precision: 0.9106
- Recall: 0.9109
- ROC AUC Score: 0.9753
===================================

Decision Tree
Model performance for Training set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
- ROC AUC Score: 1.0000

Model performance for Test set
- Accuracy: 0.9753
- F1 Score: 0.9753
- Precision: 0.9753
- Recall: 0.9753
- ROC AUC Score: 0.9798
===================================

Adaboost
Model performance for Training set
- Accuracy: 0.9786
- F1 Score: 0.9785
- Precision: 0.9792
- Recall: 0.9786
- ROC AUC Score: 0.9960

Model performance for Test set
- Accuracy: 0.9779
- F1 Score: 0.9779
- Precision: 0.9786
- Recall: 0.9779
- ROC AUC Score: 0.9957
===================================

Gausian
Model performance for Training set
- Accuracy: 0.9436
- F1 Score: 0.9432
- Precision: 0.9498
- Recall: 0.9436
- ROC AUC Score: 0.9955

Model performance for Test set
- Accuracy: 0.9419
- F1 Score: 0.9414
- Precision: 0.9485
- Recall: 0.9419
- ROC AUC Score: 0.9953
===================================

Gradient Boost
Model performance for Training set
- Accuracy: 0.9879
- F1 Score: 0.9879
- Precision: 0.9881
- Recall: 0.9879
- ROC AUC Score: 0.9997

```
----------------------------------
Model performance for Test set
- Accuracy: 0.9853
- F1 Score: 0.9853
- Precision: 0.9856
- Recall: 0.9853
- ROC AUC Score: 0.9997
===================================


Xgboost
Model performance for Training set
- Accuracy: 1.0000
- F1 Score: 0.9999
- Precision: 1.0000
- Recall: 1.0000
- ROC AUC Score: 1.0000
----------------------------------
Model performance for Test set
- Accuracy: 0.9932
- F1 Score: 0.9932
- Precision: 0.9932
- Recall: 0.9932
- ROC AUC Score: 0.9999
===================================


svm
Model performance for Training set
- Accuracy: 0.9976
- F1 Score: 0.9976
- Precision: 0.9976
- Recall: 0.9976
- ROC AUC Score: 0.9999
----------------------------------
Model performance for Test set
- Accuracy: 0.9972
- F1 Score: 0.9972
- Precision: 0.9972
- Recall: 0.9972
- ROC AUC Score: 0.9999
===================================

Random Forest
Model performance for Training set
- Accuracy: 1.0000
- F1 Score: 1.0000
- Precision: 1.0000
- Recall: 1.0000
- ROC AUC Score: 1.0000
----------------------------------
Model performance for Test set
- Accuracy: 0.9878
- F1 Score: 0.9878
- Precision: 0.9878
- Recall: 0.9878
- ROC AUC Score: 0.9997
===================================
```