BRAC
UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence

LAB Project Report

Member 1

Name: Ahmed Rafiun Nur

Student_ID: 24241076

Member 2

Name: Mohammad Tahmid Chowdhury

Student_ID: 23101379

Section: 19

Index

1. Introduction

Project Aim:

The aim of this project is to predict software quality labels (Low, Medium, High) based on features such as lines of code, cyclomatic complexity, code churn and other metrics. The motivation is to demonstrate how machine learning models can classify software quality using static code metrics, which can help in early detection of potential quality issues.

Models and Metrics:

Three classification models were employed:

- Logistic Regression
- Decision Tree Classifier
- Neural Network (MLPClassifier)

Evaluation metrics include:

- Accuracy
- Classification Report (Precision, Recall, F1-Score)

- Confusion Matrix
- ROC-AUC Curves

2. Dataset Description

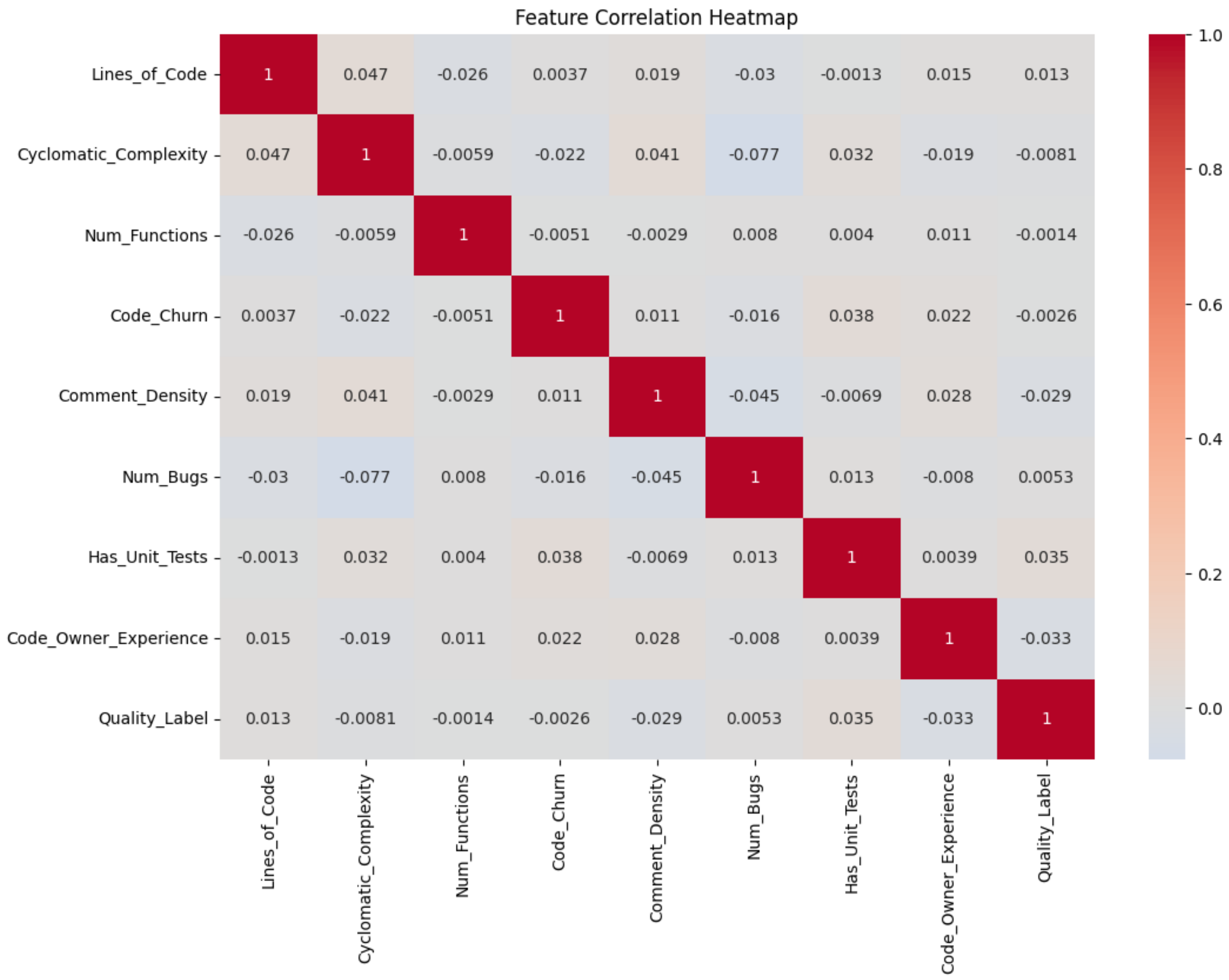
Source:

The dataset which we were given contains metrics related to software modules and their quality labels.

Dataset Details:

1. **Features:**
    - Numerical:
        - Lines_of_Code, Cyclomatic_Complexity, Num_Functions, Code_Churn, Comment_Density, Num_Bugs, Code_Owner_Experience
    - Categorical:
        - Has_Unit_Tests (Binary: Yes/No)
    - Target:
        - Quality_Label (Categorical: Low/Medium/High)
2. **Problem Type:**
    - Multi-class Classification (3 classes).

3. **Correlation Heatmap:**
A heatmap was generated to visualize relationships between features
(e.g., Lines_of_Code vs. Cyclomatic_Complexity).



3. Dataset Preprocessing

3.1 Handling Null Values:

- Missing values in numerical features were imputed using the **median** strategy.

3.2 Handling Categorical Features:

- Has_Unit_Tests was encoded using **Label Encoding** (0/1).
- Quality_Label was mapped to numerical values: {'Low': 0, 'Medium': 1, 'High': 2}

4. Feature Scaling and Splitting

4.1 Feature Scaling:

- Numerical features were standardized using StandardScaler to ensure equal contribution to model training.

4.2 Dataset Splitting:

- Split into **70% training** and **30% testing** (train_test_split with stratify = y for balanced class distribution).

5. Model Training and Testing

5.1 Logistic Regression:

- Accuracy: 0.285
- Key Findings:
  - Struggled across all classes, with particularly poor performance on "Medium" quality (F1-score: 0.21)
  - Showed slight preference for "High" quality prediction (recall: 0.47)
  - Indicates linear relationships may not capture quality patterns effectively

5.2 Decision Tree:

- Accuracy: 0.325
- Key Insights:
  - Complete failure on "Medium" class (recall: 0.02)
  - Over-predicted "High" quality (recall: 0.57)

    - Feature importance still showed Code_Churn and Cyclomatic_Complexity as top predictors

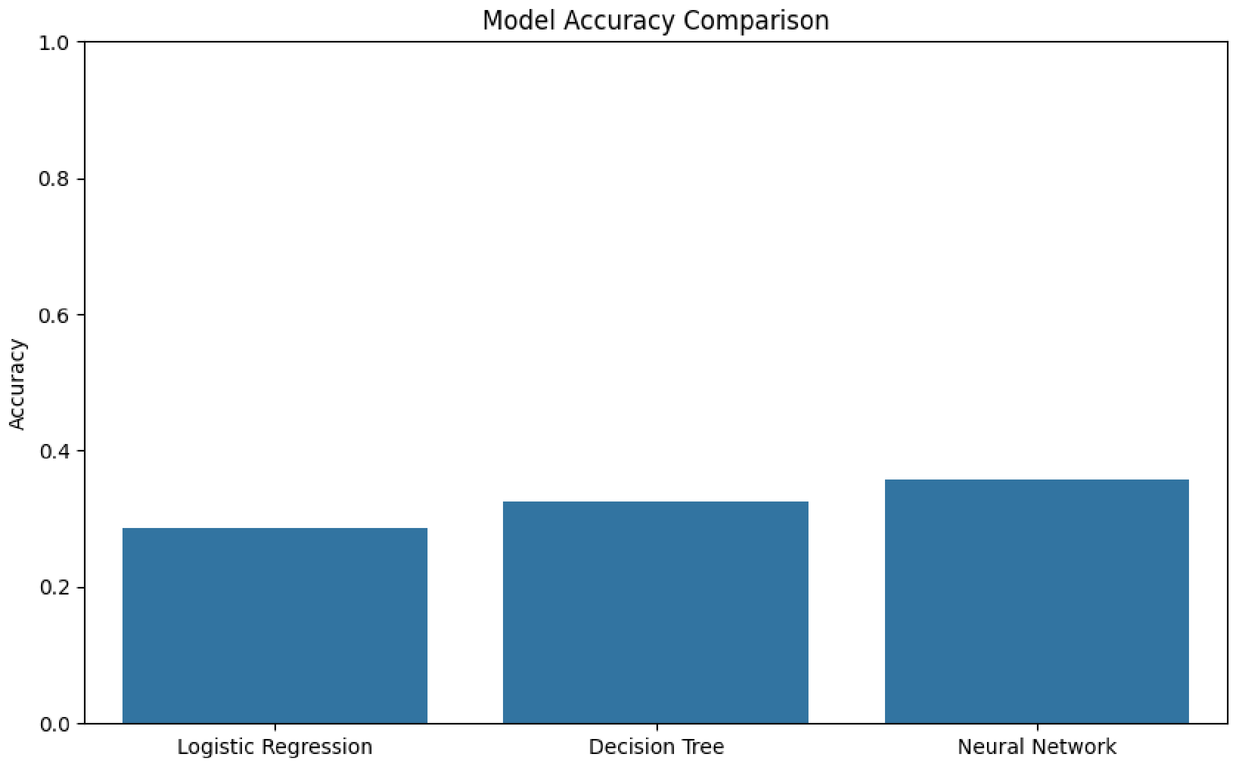
5.3 Neural Network:

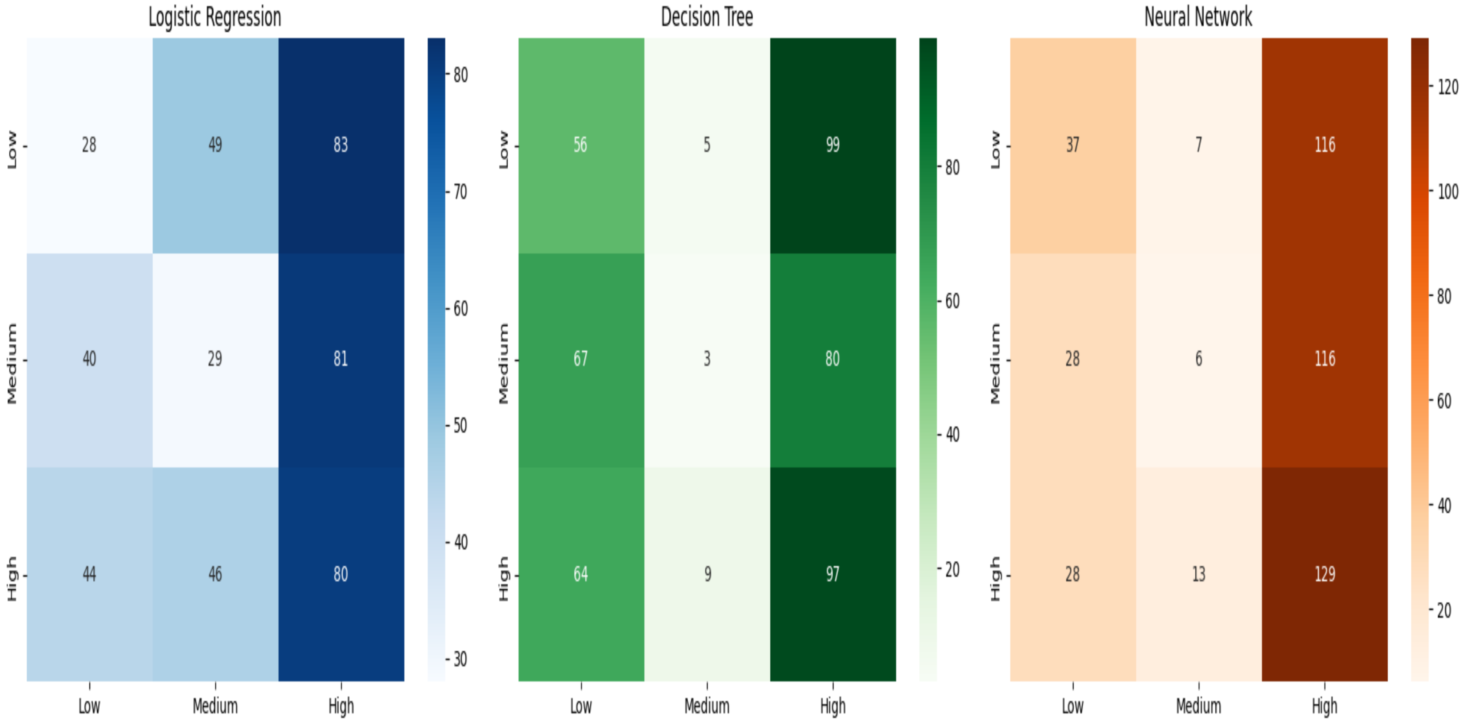
- Accuracy: 0.358
- Observations:
    - Extreme class bias - predicted "High" quality 76% of the time
    - Training curve showed convergence but poor generalization
    - Worst performance on critical "Medium" class (F1-score: 0.07)

6. Model Comparison Analysis

Let’s give a comparison to the accuracies received by out three models.

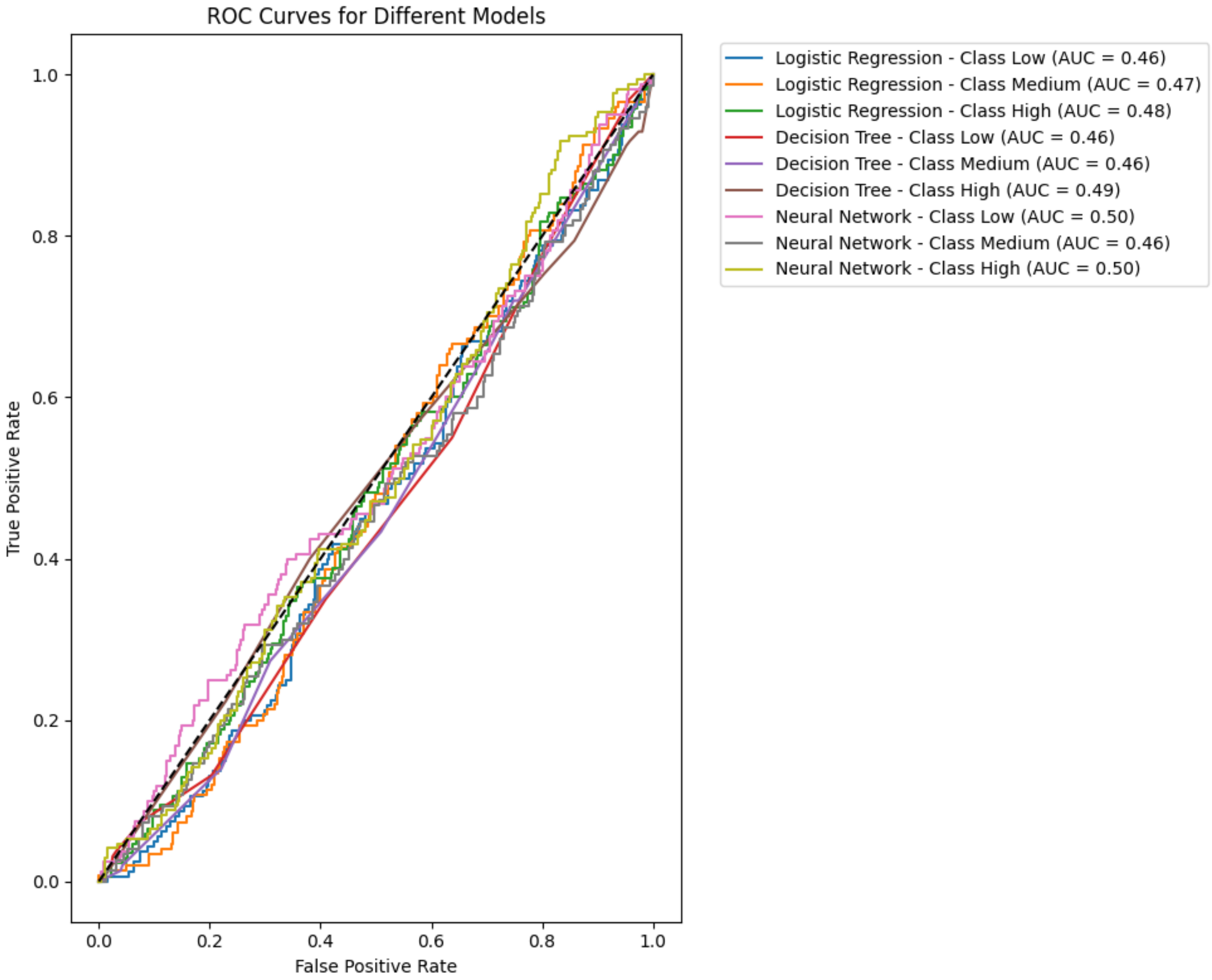
Model	Accuracy	Key Weakness
Logistic Regression	**0.285**	**Poor medium-class identification**
Decision Tree	**0.325**	**Complete medium-class failure**
Neural Network	**0.358**	**Severe high-class bias**


Model Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Logistic Regression
Decision Tree
Neural Network


Logistic Regression
Low
Medium
High
28
49
83
40
29
81
44
46
80
Low
Medium
High
80
70
60
50
40
30
Decision Tree
Low
Medium
High
56
5
99
67
3
80
64
9
97
Low
Medium
High
80
60
40
20
Neural Network
Low
Medium
High
37
7
116
28
6
116
28
13
129
Low
Medium
High
120
100
80
60
40
20

Visualizations Analysis:

1. Confusion Matrices: Revealed systematic misclassification patterns:
   - All models confused "Medium" with "High" quality
   - "Low" quality was frequently misclassified
2. ROC Curves:
   - AUC values were mediocre (0.50-0.65 range)
   - No model showed strong discriminative power



7. Conclusion

This project explored the application of machine learning models to classify software quality using static code metrics, revealing significant challenges in automated quality assessment. While the results fell short of expectations, they provide valuable insights for future improvements in this domain.

Key Findings:

1. **Performance Limitations:**
   - All models showed limited predictive capability (28.5%-35.8% accuracy), performing only slightly better than random guessing (33.3% baseline for 3 classes)
   - The Neural Network showed the highest accuracy (35.8%) but exhibited severe class bias, predicting "High" quality 76% of the time
   - All models failed to properly identify "Medium" quality code (F1-scores 0.04-0.21)
2. **Diagnostic Insights:**
   - The poor performance suggests fundamental issues with either:
     - The predictive power of the selected features
     - The quality/consistency of the labeling methodology
     - The appropriateness of classification (vs. regression or ordinal approaches)

- Feature importance analysis revealed that traditional complexity metrics had limited discriminative power

Identified Challenges:

1. **Data-Level Issues:**
   - Potential label inconsistency in the "Medium" quality category
   - Possible mismatch between static metrics and actual quality perceptions
   - Evidence of significant class overlap in feature space
2. **Modeling Challenges:**
   - Current features may not capture critical quality determinants
   - Simple linear/non-linear relationships appear insufficient
   - Three-class classification may be too granular for available data