**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
   * Models and Metrics
2. **Dataset Description**
   * Source
   * Dataset Details
   * Problem Type
   * Correlation Heatmap
3. **Dataset Preprocessing**
   * Handling Null Values
   * Handling Categorical Features
4. **Feature Scaling**
   * Dataset Splitting
5. **Model training & testing**
6. **Model Comparison Analysis**
7. **Conclusion**

**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:

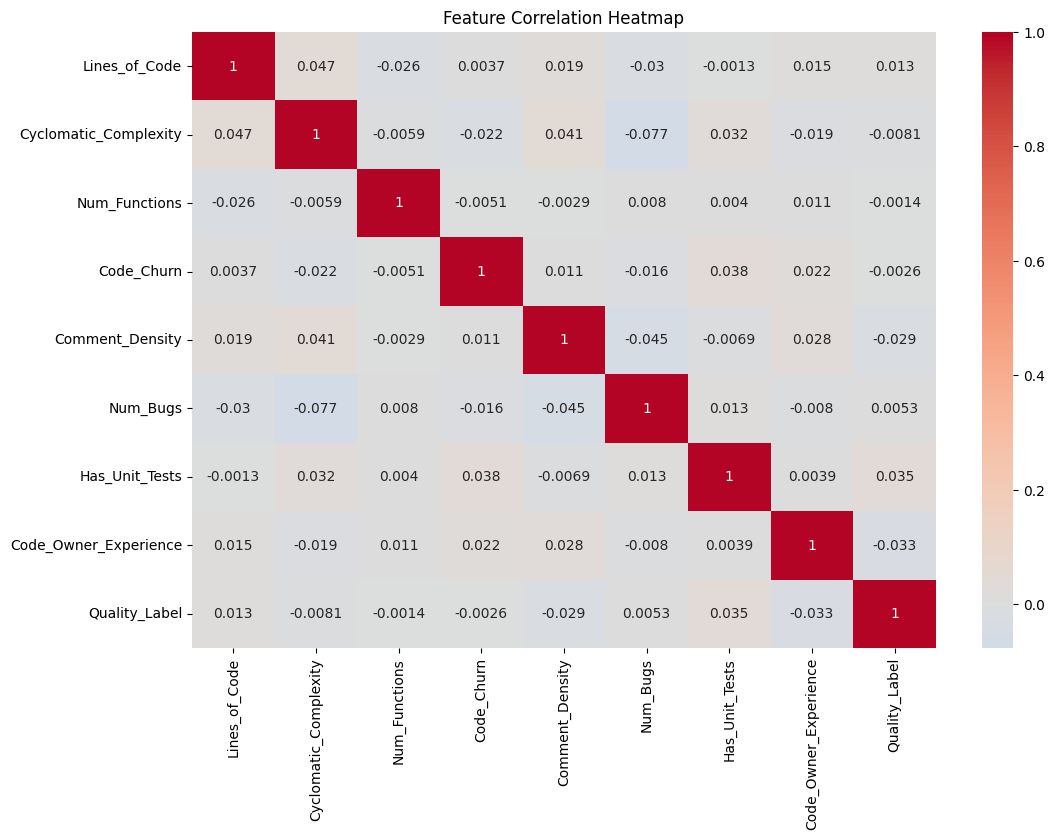
* Accuracy
* Classification Report (Precision, Recall, F1-Score)
* Confusion Matrix
* ROC-AUC Curves

**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

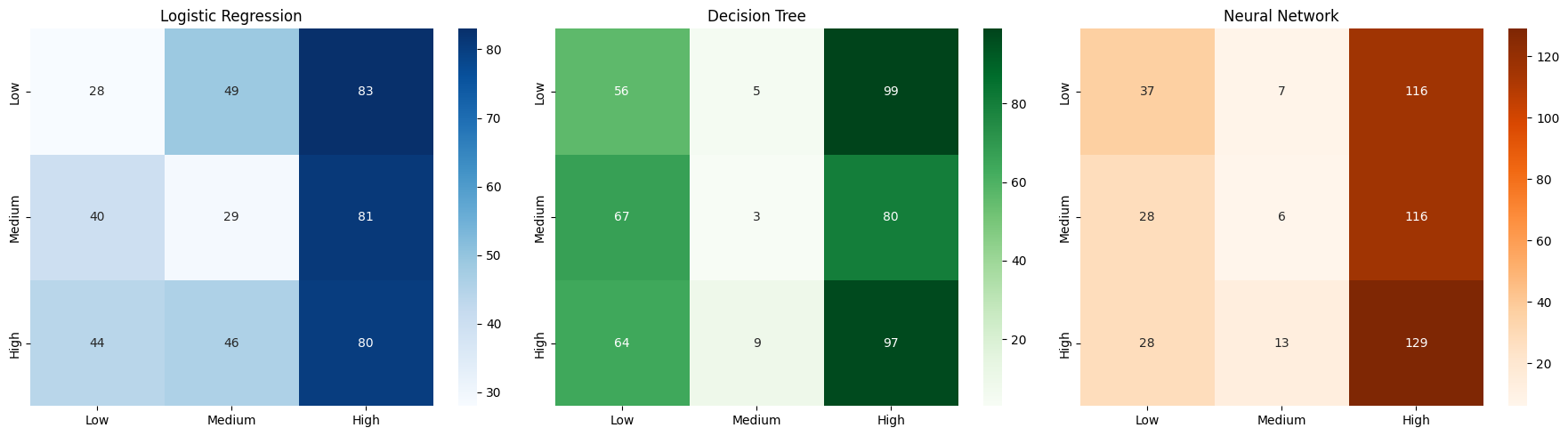
**5.3 Neural Network:**

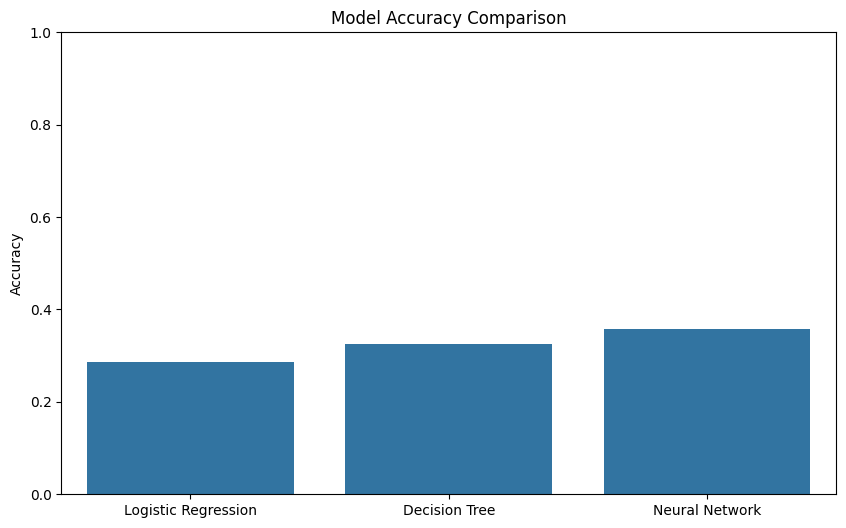
* Accuracy: 0.358
* Observations:
  + Extreme class bias - predicted "High" quality 76% of the time
  + Training curve showed convergence but poor generalization
  + Worst performance on critical "Medium" class (F1-score: 0.07)

**6. Model Comparison Analysis**

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

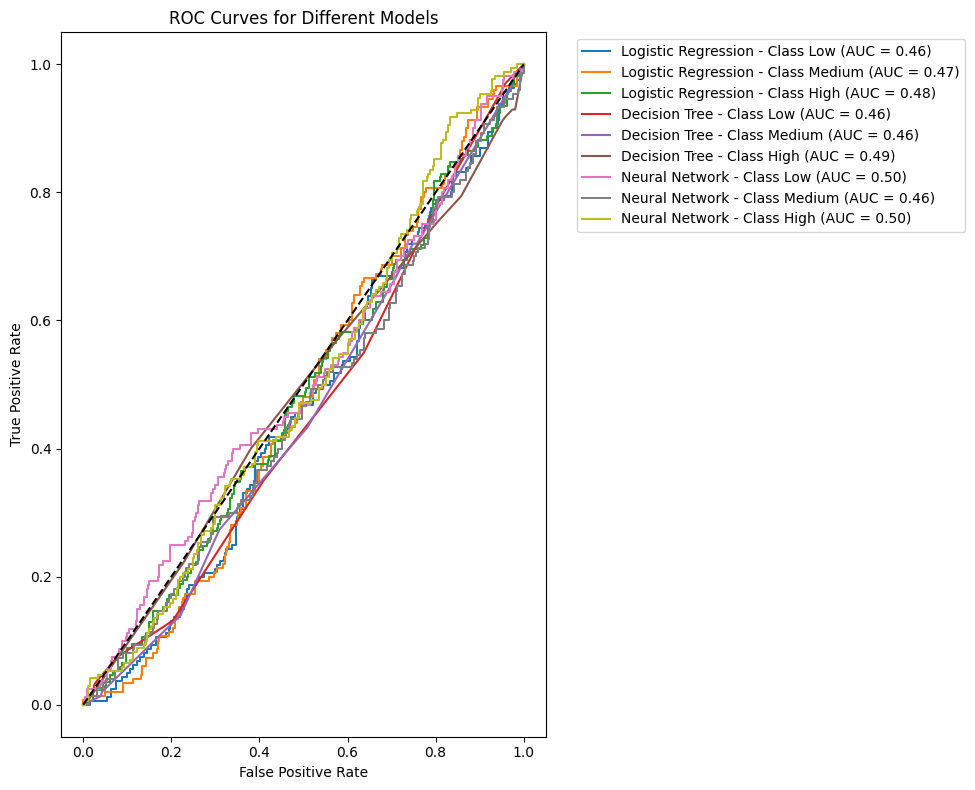
| **Model** | **Accuracy** | **Key Weakness** |
| --- | --- | --- |
| **Logistic Regression** | **0.285** | **Poor medium-class identification** |
| **Decision Tree** | **0.325** | **Complete medium-class failure** |
| **Neural Network** | **0.358** | **Severe high-class bias** |

****

****

**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