Inspiring Excellence

Name: Nazmuz Sakib Zarif
Student ID: 222992433
Name: Meherab Hossain
Student ID: 22299071
Course: CSE422
Section: 03
Submission Date: 10.05.2025

TABLE OF CONTENTS

Introduction

In this project we aim to build a model that can classify the quality of a software based on various code-related features. This model will help us automate the assessment of software quality and allow us to identify which parts of our coding are having a greater impact on the performance of the code. It saves time helping us identify problems quicker by using the power of machine learning models for classification.

Dataset Description

Features:

8 input features and 1 target
The dataset we are using contains 9 columns with the last column being our target value, there are 9 features for us to use in classifying the dataset into one of the three target values.

Type

Classification
As we are classifying into three possible outcomes (high, medium and low). It is a classification problem

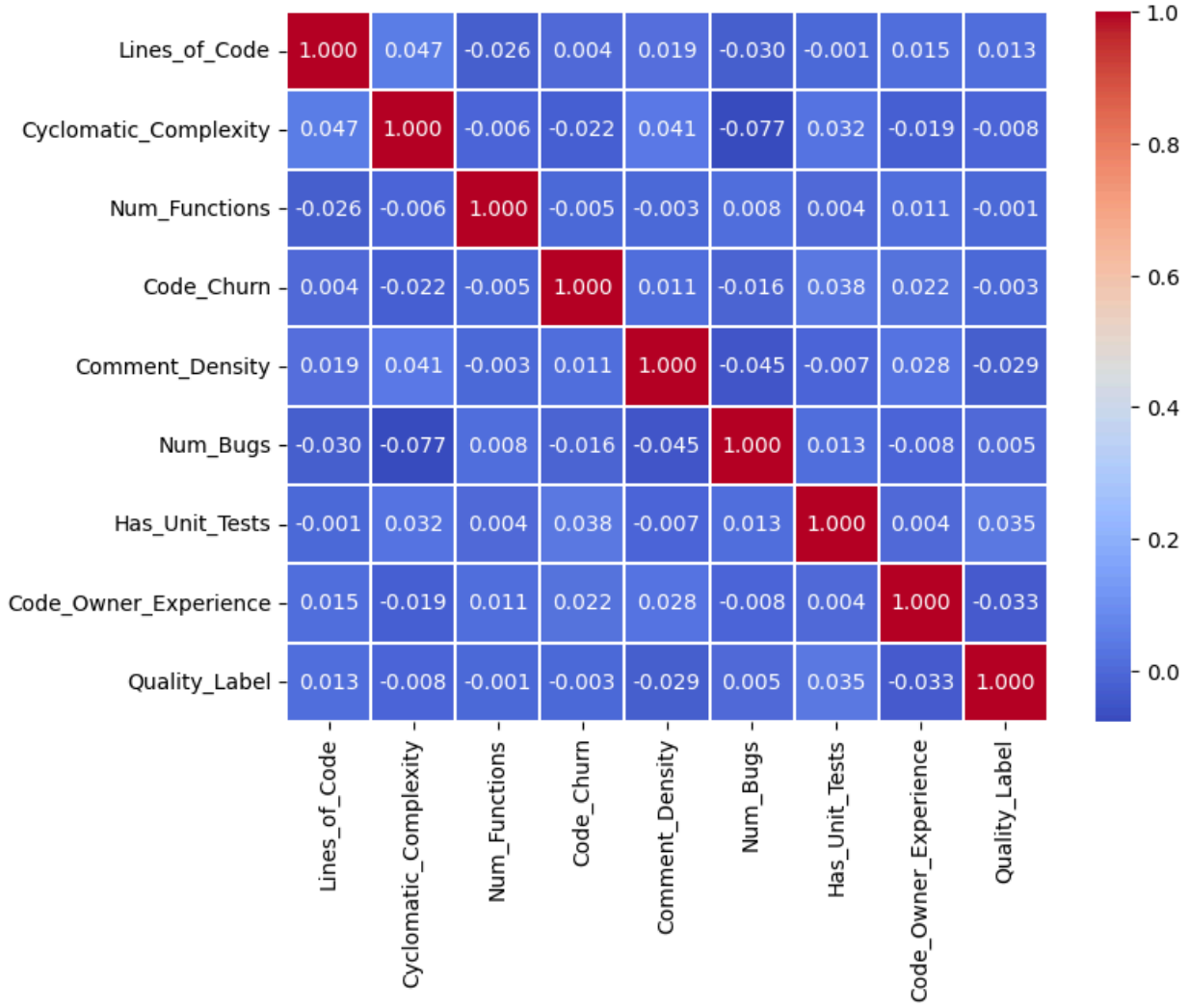
Datapoints

1600 instances
In total there are 1600 rows in the dataset excluding the top row which are all individual datavalues

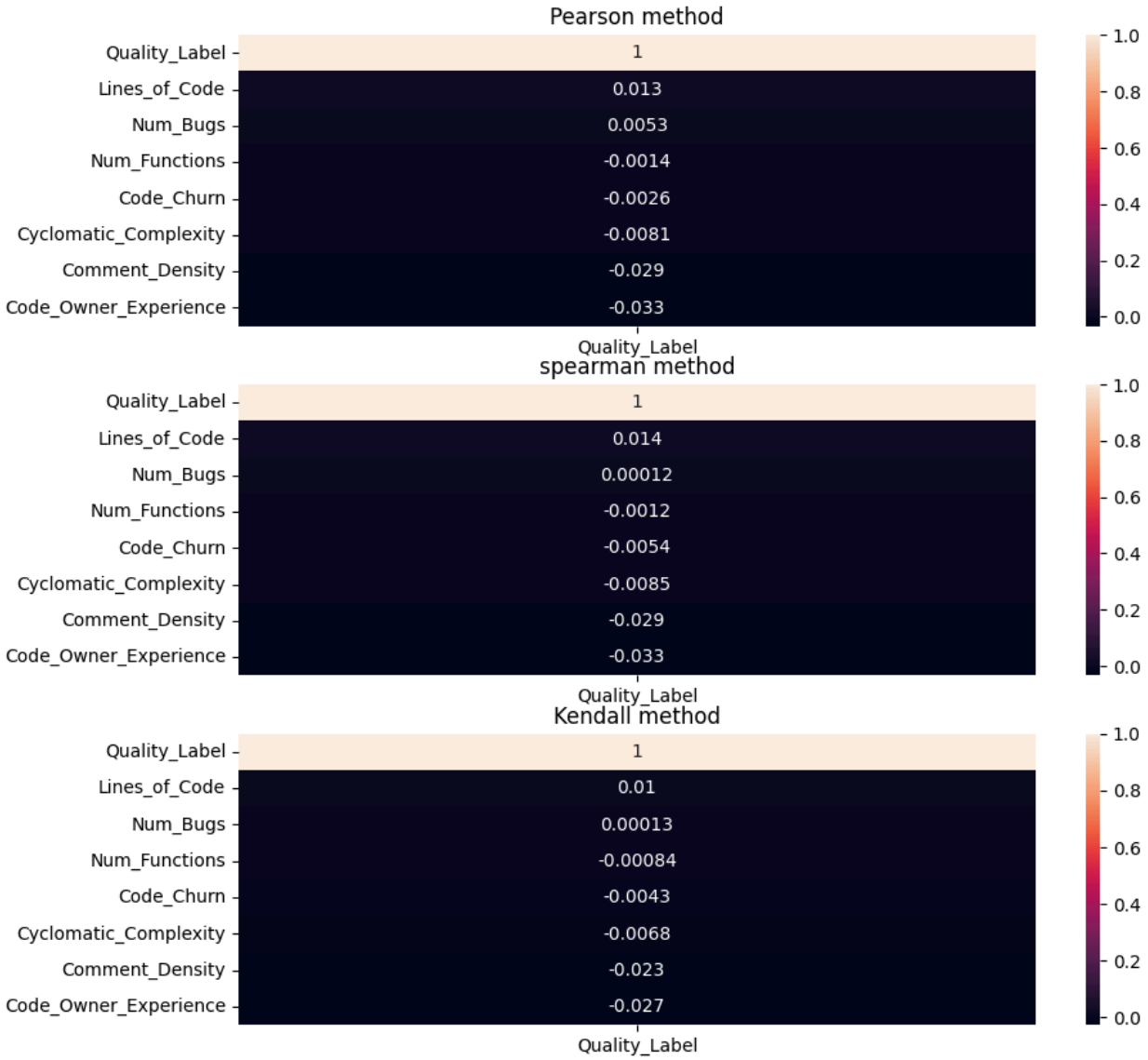
Feature Type

7 Quantitative and 1 Qualitative feature
The first seven features are numbers representing a quantity and the last feature is a yes or no column representing a binary quality

Correlation



From the Heatmap we can see how each of the features correlates to the target and each other. This helps us understand how each feature is affected by another feature changing and overall how the target is affected as well.



These are three different correlation methods that better help us understand which numerical features correlate with the target the most. From all three we can see Code_Owner_Experience has the highest correlation among the Numerical Variables and thus has the highest effect on the target variable.

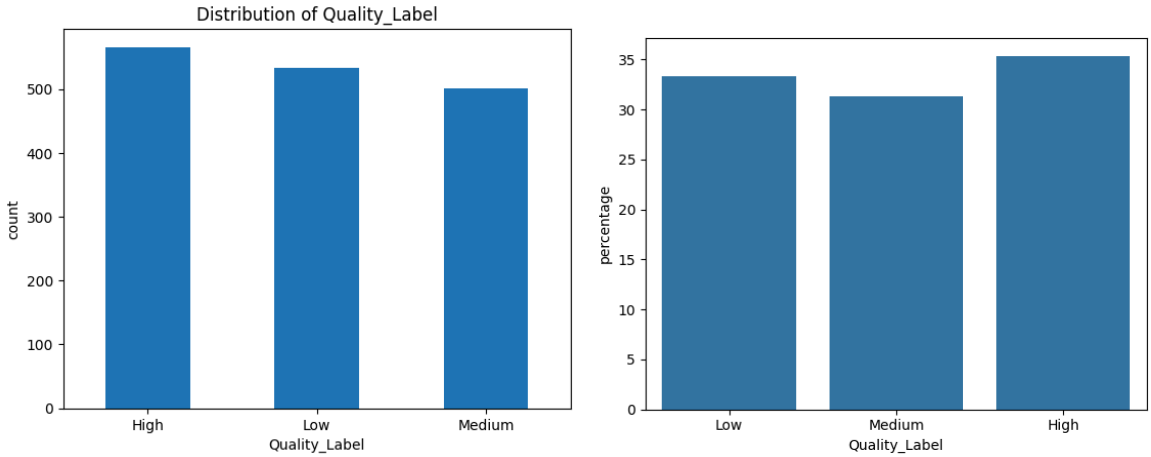
However almost all of the features are not highly correlated with the target overall and so little effect on the target.

	Has_Unit_Tests	Quality_Label
Has_Unit_Tests	0.998436	0.056199
Quality_Label	0.056199	0.999687

There is only one categorical variable, Has_Unit_Tests which also has a significant effect on the target as seen from the Heatmap. Using Cramer's V it has a correlation of 0.056199 which is greater than most numerical features and thus affects the target significantly.

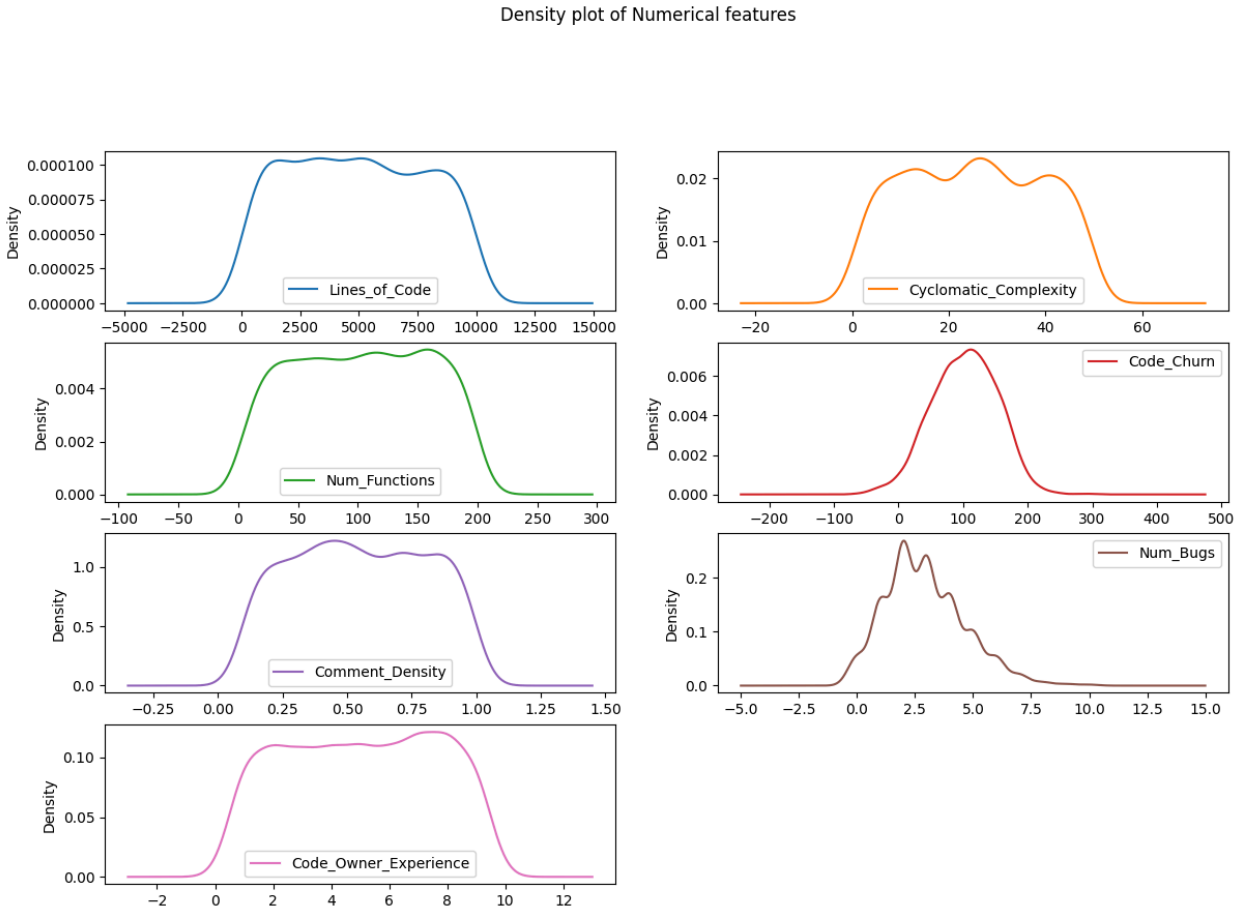
Here we can clearly see that Has_Unit_Test, Lines_of_Codes, Comment_Denstity and Code_Owner_Experince has the most significant impact on the outcome and so these will be used to train our models.

Imbalanced Dataset

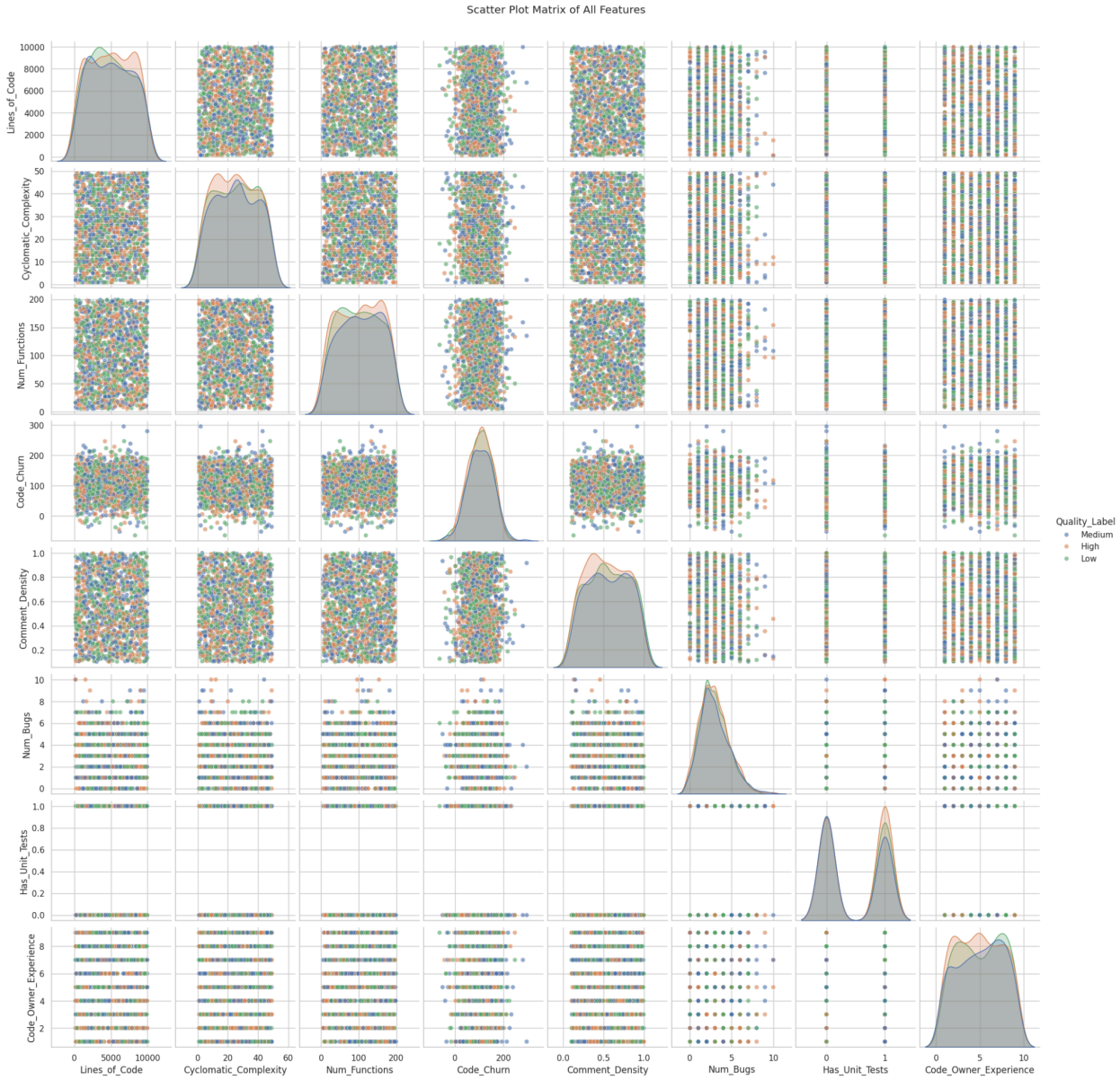


The unique classes do not have an equal number of instances but there are relatively close together, High:566, Medium:501, Low:533.
There is an imbalance of only 4.525% or 61 instances between the max and min percentage of instances which is relatively low and should result in a reliable result but It can be further improved by undersampling High instances and undersampling Low instances. So the number of instances are more close together.

EDA Analysis



Overall the density plot shows that most numerical features are not too skewed and can be used without transformation. Even the most skewed Number of Bugs feature has a skewness of only -0.667812 which may benefit from transformations but can be used without as well. Further analysis can be seen in the Notebook attached with the paper. Using these analysis, we are able to understand the dataset better and making decisions accordingly.



The scatter plot of the dataset clearly shows there are no clear linear separators between the target classes for any of the features hence the accuracy of the linear models are expected to be low compared to training with other datasets.

Dataset Pre-processing

Null/Missing values

Problem: Using database.isnull().sum() we can clearly see that there are 80 NaN values for Lines of cordes, Comment Density and Code_churn. These NaN values are not allowed in some models such as Logistic Regression and Neural Networks.
Solution: To deal with these values we can either use database.dropna() to drop all the rows with NaN values in features or replace the NaN values with median. Implementing both we are using the median approach as it allows us to have a greater number of instances which may help train the model.

Categorical values

Problem: Categorical variables have string classes which cannot be interpreted by the machine learning algorithms. So to make them accessible to the models we have to encode them to discrete numbers so the models are able to classify them effectively.
Solution: For the Has_Unit_Tests variable there are two classes, Yes and No which we can easily use binary encode to map to 1s and 0s. For the Target variable Quality_Label, there are three classes and so we take the help of label encoding to encode the target.

Feature Scaling

Problem: As each variable has different means and standard deviations, the model may misinterpret the data and the result may be biased.
Solution: To avoid this we use the StandardScaler() function to normalize the variables so they do not interfere with the model training and we get an unbiased result.

Dataset splitting

As our Target variable is categorical and multiclass we have to ensure the data splitting is stratified so that the class distribution is equal in the train and test sets. We split the data into 70% training and 30% testing to ensure a good training and evaluation of the model. We use the function train_test_split(X_scaled, y, test_size=0.3, stratify=y, random_state=42) to achieve this. The random state=42 ensures that every time we run the code the same random instances are split into the train and test set to ensure we get the same result unless we change the random_state.

Model training & testing

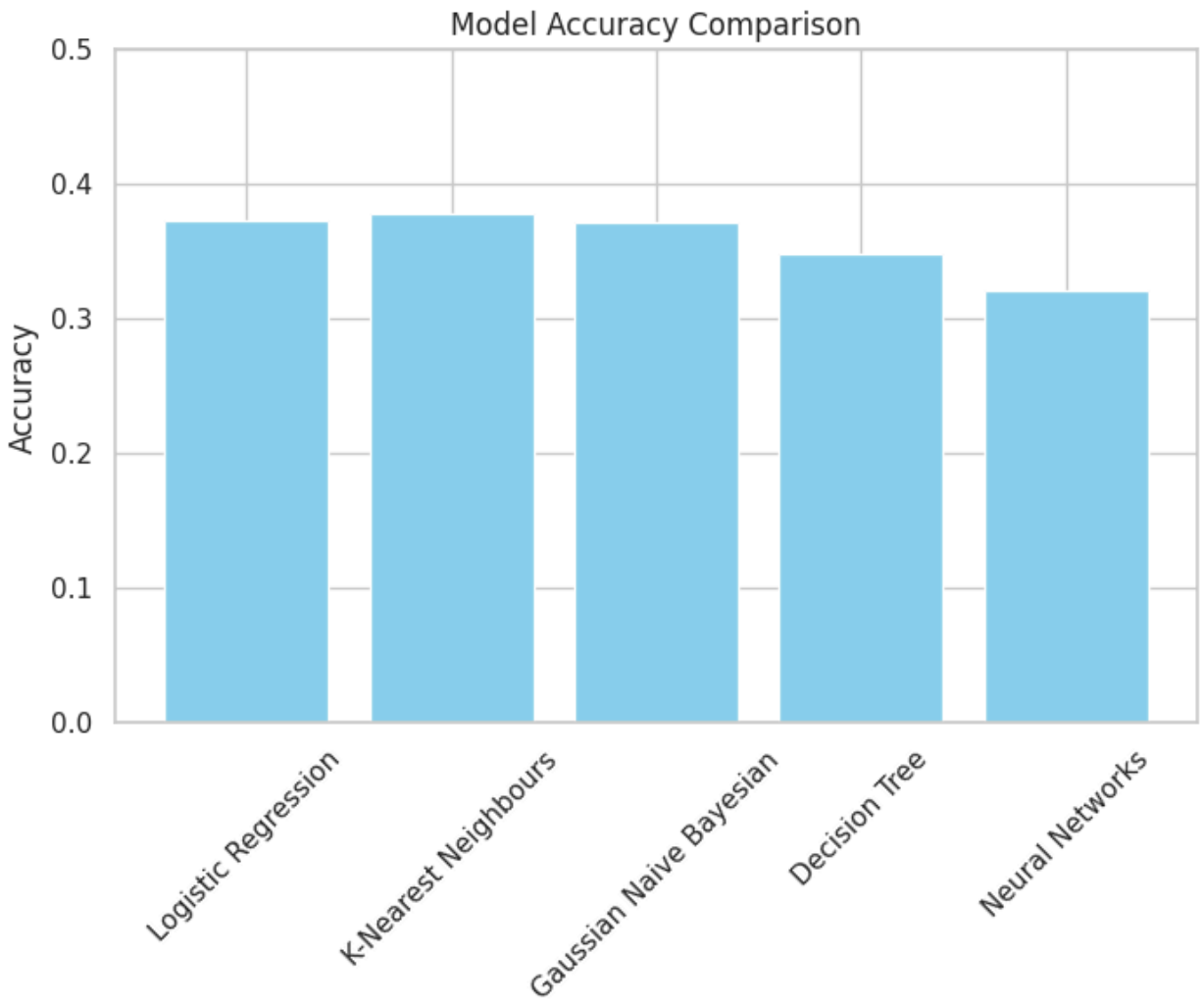
As we have a classification problem, we are gonna be using the following models to predict our target,

- K-Nearest Neighbors.
- Decision Tree.
- Logistic Regression.
- Gaussian Naive Bayesian.
- Neural Network

Applying these 5 models together will help us evaluate the best model for our dataset and how each model interprets the data differently.

Model selection/Comparison analysis

Prediction Accuracy



The bar chart clearly shows that K-Nearest Neighbors has the highest Accuracy among all the 5 models and is on first sight the most accurate model.

Precision, recall comparison

Classification Report (Logistic Regression):

	precision	recall	f1-score	support
High	0.39	0.61	0.47	170
Low	0.35	0.31	0.33	160
Medium	0.37	0.17	0.24	150
accuracy			0.37	480
macro avg	0.37	0.36	0.35	480
weighted avg	0.37	0.37	0.35	480

Classification Report (K-Nearest Neighbours):

	precision	recall	f1-score	support
High	0.41	0.53	0.46	170
Low	0.33	0.35	0.34	160
Medium	0.39	0.23	0.29	150
accuracy			0.38	480
macro avg	0.38	0.37	0.36	480
weighted avg	0.38	0.38	0.37	480

Classification Report (Gaussian Naive Bayesian):

	precision	recall	f1-score	support
High	0.40	0.59	0.48	170
Low	0.35	0.34	0.35	160
Medium	0.32	0.15	0.21	150
accuracy			0.37	480
macro avg	0.36	0.36	0.34	480
weighted avg	0.36	0.37	0.35	480

Classification Report (Decision Tree):

	precision	recall	f1-score	support
High	0.41	0.42	0.41	170
Low	0.33	0.33	0.33	160
Medium	0.30	0.29	0.29	150

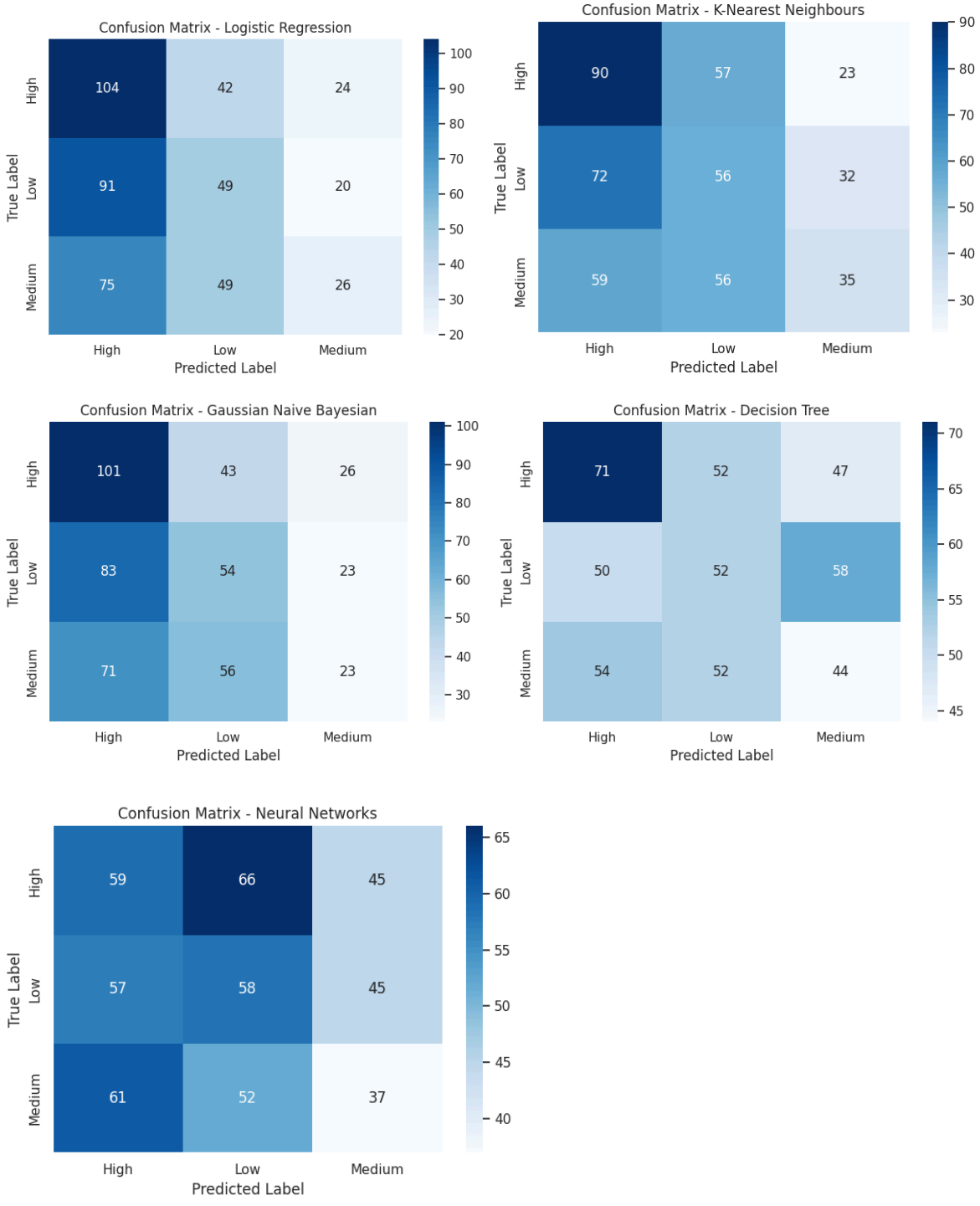
accuracy			0.35	480
macro avg	0.34	0.35	0.35	480
weighted avg	0.35	0.35	0.35	480

Classification Report (Neural Networks):

	precision	recall	f1-score	support
High	0.33	0.35	0.34	170
Low	0.33	0.36	0.35	160
Medium	0.29	0.25	0.27	150
accuracy			0.32	480
macro avg	0.32	0.32	0.32	480
weighted avg	0.32	0.32	0.32	480

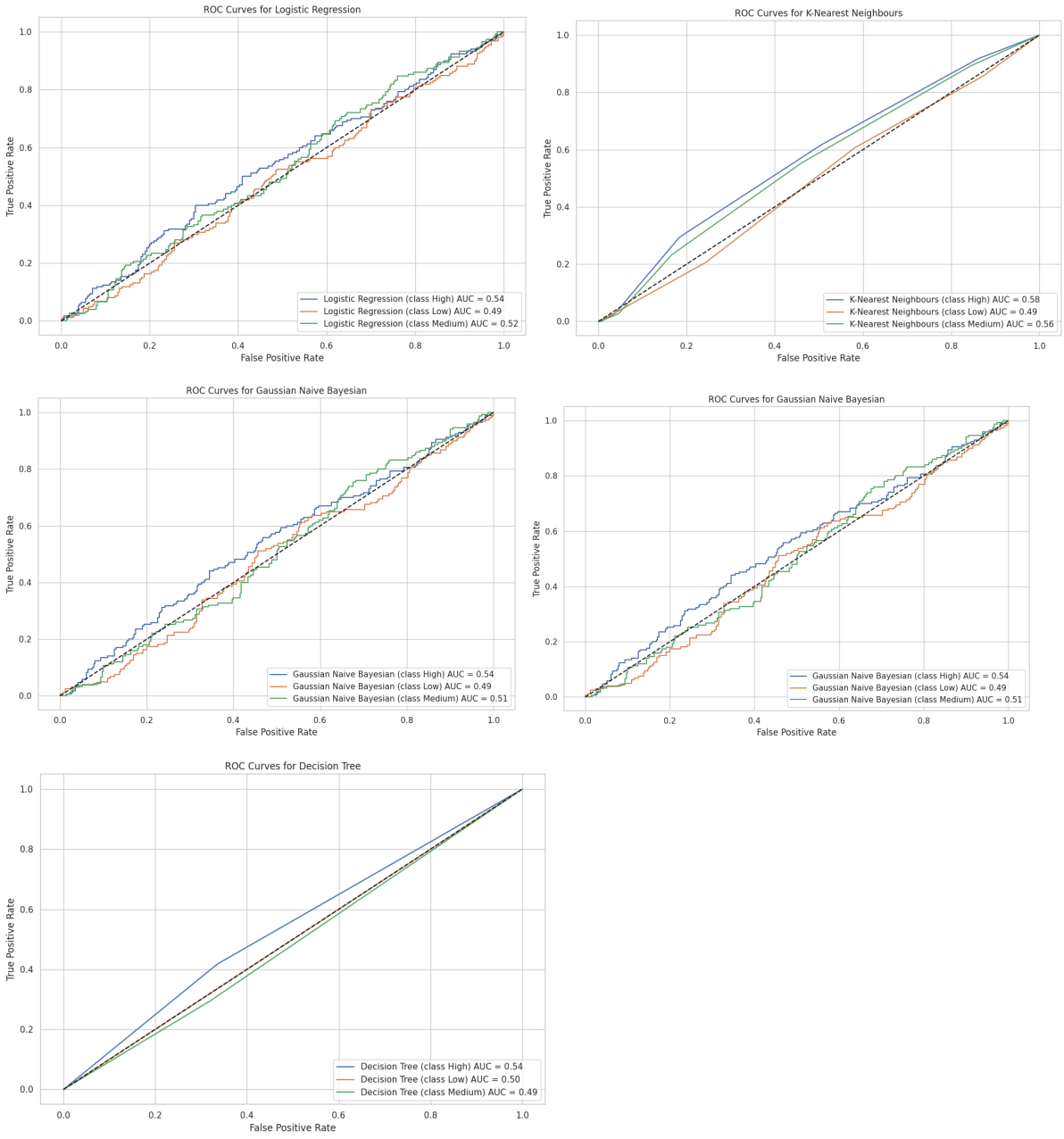
Comparing the Precision, Recall and F1-score, K-Nearest Neighbor still edges out barely with the Logistic Regression model as a close second.

Confusion Matrix



However looking at the confusion matrix, it is clear that K-Nearest Neighbor and Logistic regression is both biased towards the High classification with most classifications being High. In this sense the Neural Network and Decision Tree have the most balanced confusion matrix but can be said to be near random guessing.

AUC score, ROC curve



Overall AUC (One vs Rest)
Logistic Regression: 0.517
K-Nearest Neighbours: .542
Gaussian Naive Bayesian: 0.511
Decision Tree: 0.510
Neural Networks: 0.511

We can see the highest AUC score is also achieved by K-Nearest Neighbours as well.

Conclusion

So even though the K-Nearest Neighbour algorithm is biased towards high it can still split ‘High’ and ‘Low’ classes more effectively than others and so is the best model for our dataset. Almost all the models struggle with dealing with the dataset as there are no clear linear splits between the data as seen from the scatter diagram. Logistic Regression and Naive Bayes closely follow the KNN algorithm but the Neural Network Algorithm can be said to be the most evenly distributed algorithm as the ROC curve for each class is greater than or equal to 0.5 but even so it cannot separate the classes well enough. The Decision Tree can be said to have the most balanced confusion matrix as all the predictions can be said to be consistent between the classes as observed from the confusion matrix. Overall this dataset does not have any easy identifiable splits in the features and so almost all the algorithms struggle to give a good result. It was particularly challenging to select the right features to use for the classification but in the end using the regression analysis we were able to select the best 4 features for us to use. This experiment should help develop a basic knowledge on the dataset and can be used as a stepping stone to further research and improve the classification of software around the world.