-------------------------Title----------------------------

1.Abstract

Code smells are problematic design patterns in code that indicate deeper issues in structure and flow of code. Accurate identification and detection of code smells in code help to improve code maintainability and reduce future problems of code extensibility. Detecting code smells is a challenging task when dealing with code smell datasets because of presence of imbalance between minority (code smell) and majority class (no-code smell) which can be tackled by applying oversampling techniques. We are using industry-relevant MLCQ dataset which has four severity-levels of code-smell and four types of code smell (god-class, feature-envy, data-class, long-method) and there is imbalance between different severity levels. Machine Learning offers promising approach in code smell detection which is hindered by class imbalance, many oversampling techniques are available to address this issue. This study empirically evaluates the effectiveness of various oversampling methods and compare results using performance metrics like precession, recall, F1-score, accuracy. We have used LDA (Linear discriminant analysis) for data dimensionality reduction.

2.Introduction

Quality of code is responsible for maintainability, reliability and long-term usage of code. Code smells can cause problems for developers in future. Code smells such as (god-class, feature-envy, data-class, long-method) can impact quality of code and software in a negative fashion and decrease understandability of code.

Various machine learning techniques can help in early detection of code smells but inherent class imbalance present in code smell datasets can result in biased results by models favoring majority class. Oversampling methods can be used to tackle this imbalance problem in datasets. These oversampling methods artificially generate new data points to create a balanced dataset. In this study we use multiple oversampling techniques on code smell dataset and compare results using performance metrices like precession, recall, F1-score, accuracy.

Code can be represented using high dimensional feature space, where each feature represent a specific aspect of code structure but high dimensionality can cause challenge for machine learning models. This issue can be tacked by applying dimensionality reduction techniques like Linear Discriminant Analysis (LDA) which reduce dimensions by prioritizing features which can distinguish between resultant classes effectively.

3.Related Work

4.Methodology

**4.1 Severity Classification**

MLCQ datasets has four severity levels where meaning of each level is described in [Table1](#Table1)

|  |  |
| --- | --- |
| Severity Level | Description |
| 1 | A class or method that is unaffected receives a score of 1 for "No smell” |
| 2 | A class or function that is only marginally affected receives a score of 2 for a non-severe smell |
| 3 | A class or method receives a smell score of 3 if it possesses all of the qualities of a smell; |
| 4 | There is a severe smell, and its size, complexity, and coupling values are extremely high. It receives a score of 4 |

**Table 1: Severity Level Description**

**4.2 Structure of Dataset**

Each dataset has total 420 instances and distribution of severity level for each dataset is shown in [Table2](#Table2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 | Total |
| Data -Class | 151 | 32 | 113 | 124 | 420 |
| God -Class | 154 | 29 | 110 | 127 | 420 |
| Feature-Envy | 280 | 23 | 95 | 22 | 420 |
| Long-Method | 280 | 11 | 95 | 34 | 420 |

**Table2: Dataset Distribution**

**4.3 Workflow of study**

Work flow of our study is described in points below and in [Figure1](#Figure1):

1)Preprocessing is applied on data sets to remove less/not useful features.

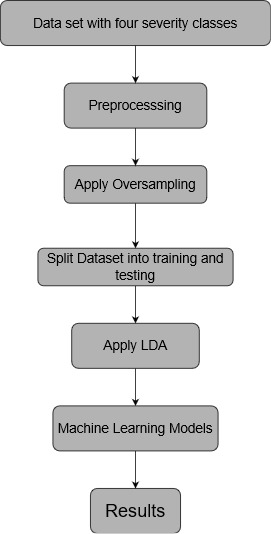
2)Datasets are imbalanced so oversampling methods (ADASYN, SMOTE, LoRAS, Borderline SMOTE, KMeansSMOTE, Random Oversampling) is applied.

3)Split dataset into training and testing(80:20)

4)Apply Dimensionality reduction technique – LDA

5)Apply Machine learning models (KNN, Random Forest, AdaBoost ,Gradient Boost , Support Vector Machine , Logistic Regression , Naïve Bayes)

6)Analyse results using various performance metrices (F1-Score, Precision , Recall , Accuracy)



**Figure 1 work flow of study**

**4.4 Preprocessing**

Each dataset contain wide range of features. Some of those features are not useful for our study and can prove to be problematic for machine learning models. So we have removed features which are in string format and are not useful for our study like name of project, complexity, package name.

**4.5 Oversampling Methods**

Inherent class imbalance in datasets[[Table2](#Table2)] pose as a obstacle for machine learning models so oversampling techniques are applied to counter class imbalance. These methods artificially generate new data points from minority class to create balanced dataset. we are using ADASYN, LoRAS, RandomOversampler, SMOTE, KmeansSMOTE, Borderline SMOTE in our study.

**4.5.1 ADASYN**

ADASYN stands for Adaptive Synthetic Sampling Method for Imbalanced Data. It is a technique which is used to handle datasets where one class is significantly more abundant than the others. ADASYN works by generating synthetic examples for the minority class, focusing more on areas where the class imbalance is more pronounced. This helps in training models to make better predictions by reducing biasedness toward majority class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 151 | 148 | 113 | 124 |
| God -Class | 154 | 159 | 110 | 127 |
| Feature-Envy | 280 | 179 | 295 | 275 |
| Long-Method | 280 | 277 | 289 | 287 |

**Table3 Class distribution after applying ADASYN**

**4.5.2 LoRAS**

LoRAS stands for Localised Random-Oversampling. It tackles class imbalance by artificially generating minority class instances but it does not do this randomly and focus more on regions where there is most class imbalance. Thus it eliminate class imbalance and increase performance of machine learning models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 151 | 151 | 151 | 151 |
| God -Class | 154 | 154 | 154 | 154 |
| Feature-Envy | 280 | 280 | 280 | 280 |
| Long-Method | 280 | 280 | 280 | 280 |

**Table4 Class distribution after applying LoRAS**

**4.5.3 SMOTE**

SMOTE or Synthetic Minority Over-sampling Technique is a popular oversampling technique. It synthetically generate instances of minority classes by interpolating between existing minority class instances. SMOTE balance class distribution and make prediction of machine learning models more robust and unbiased.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 151 | 151 | 151 | 151 |
| God -Class | 154 | 154 | 154 | 154 |
| Feature-Envy | 280 | 280 | 280 | 280 |
| Long-Method | 280 | 280 | 280 | 280 |

**Table5 Class distribution after applying SMOTE**

**4.5.4 KMeansSMOTE**

KMeansSMOTE combines KMeans clustering algorithm with SMOTE to artificially generate instances of minority class to bring balance in dataset. By clustering the minority class data into groups using KMeans and then applying SMOTE within each cluster, KMeansSMOTE aims to create synthetic examples that are more representative of the underlying data distribution.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 151 | 152 | 156 | 155 |
| God -Class | 154 | 155 | 159 | 160 |
| Feature-Envy | 280 | 281 | 285 | 280 |
| Long-Method | 280 | 280 | 286 | 283 |

**Table6 Class distribution after applying KmeansSMOTE**

**4.5.5 Random Oversampling**

Random Oversampling is a simple oversampling technique which works by randomly duplicating examples from minority class until a more balance dataset is not achieved. Even though this technique of oversampling is simple it can improve performance of machine learning models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 105 | 105 | 105 | 105 |
| God -Class | 105 | 105 | 105 | 105 |
| Feature-Envy | 105 | 105 | 105 | 105 |
| Long-Method | 105 | 105 | 105 | 105 |

**Table7 Class distribution after applying Random Oversampling**

**4.5.6 Borderline SMOTE**

Borderline SMOTE is a oversampling technique that focus on creating artificial data points on borderline of minority and majority classes. It aims to address the challenge of misclassification in the minority class while minimizing the risk of generating noisy samples thus overall improving performance and robustness of machine learning models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Severity1 | Severity2 | Severity3 | Severity4 |
| Data -Class | 151 | 151 | 151 | 151 |
| God -Class | 154 | 154 | 154 | 154 |
| Feature-Envy | 280 | 280 | 280 | 280 |
| Long-Method | 280 | 280 | 280 | 280 |

**Table8 Class distribution after applying Borderline SMOTE**

**4.6 Dimensionality Reduction**

Dimensionality reduction is applied in order to reduce or ignore features which are less important. In this study we have used LDA (Linear Discriminant Analysis) to achieve dimensionality reduction. It project data into a low-dimension space which increase separation between different classes thus improve performance of machine learning models for better prediction.

**4.7 Machine Learning models**

We have used 7 machine learning models K-Nearest Neighbour (KNN) , Random Forest (RF) , AdaBoost , Gradient Boosting ,Support Vector Machine (SVM), Logistic Regression , Naïve Bayes.

**4.7.1 K-Nearest Neighbor**

KNN is simple machine learning algorithm which is used in classification and regression problems. It assigns a class to a data point based on its neighbours , considering the K nearest data points in space. Majority class among neighbours is assigned to given data point. We have used two values of K for KNN that is K=3 and K=5 .

**4.7.2 Random Forest**

Random forest is an ensemble learning method which create multiple decision trees to provide classification or regression result. In classification it outputs class which is resulted by majority of decision trees and in case of regression problems it provide answer as a mean of results provided by decision trees. It enhances accuracy and controls overfitting.

**4.7.3 AdaBoost**

AdaBoost or Adaptive Boosting is also an ensemble method which works by combining multiple weak classifiers to finally creating a strong classifier. It improve predictive accuracy by combining multiple “weak learners”. It iteratively adjust the weights of incorrectly classified instances.

**4.7.4 Gradient Boost**

Gradient Boosting is also an ensemble technique which builds models sequentially in which each model fix errors of its previous model. It works by minimizing a loss function and combines strengths of multiple weak learners to create a robust predictive model.

**4.7.5 Support Vector Machine**

Support Vector Machine (SVM) is supervised machine learning algorithm used for classification and regression. It works by finding a optimal hyper plane that maximize margin between different classes in feature space. SVM is effective for high dimensional spaces.

**4.7.6 Logistic Regression**

Logistic regression is a statistical model. It models the probability of a outcome based on predictor variables using a logistic function. The algorithm estimates the likelihood of a particular class by fitting the data to a logistic curve.

**4.7.7 Naïve Bayes**

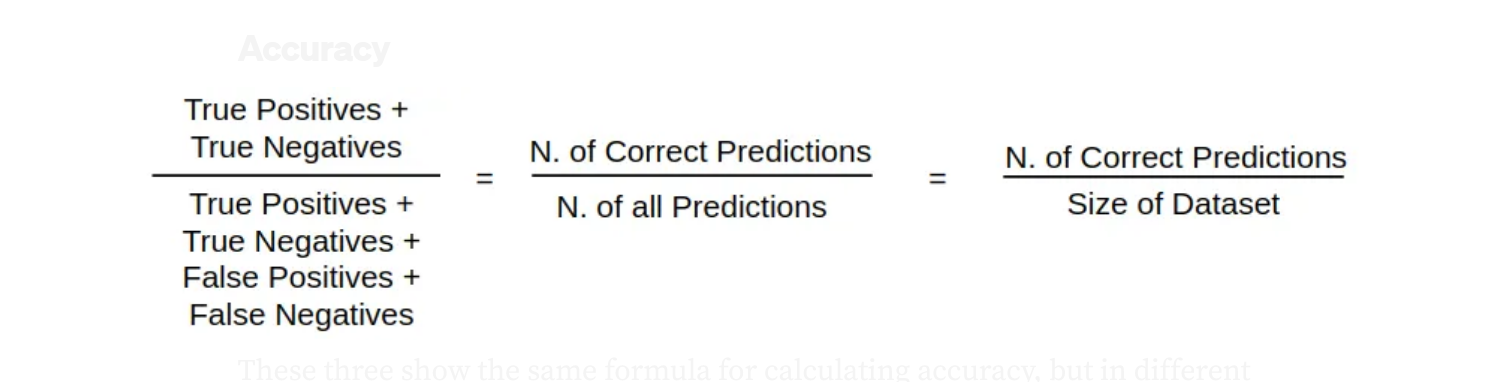
Naïve Bayes is a classifier which is based on Bayes Theorem and assumes independence between different features .Even after this assumption of independence between features it works very well in real world problems. So it is widely used in classification problems.

**4.8 Performance Metrics**

Performance metrics are quantitative measures which help us to evaluate the effectiveness of machine learning models. They help to understand how well model performs on given dataset and how accurate are the results . In our study we have used accuracy , F1-score , recall and precision to evaluate our models.

**4.8.1 Accuracy**

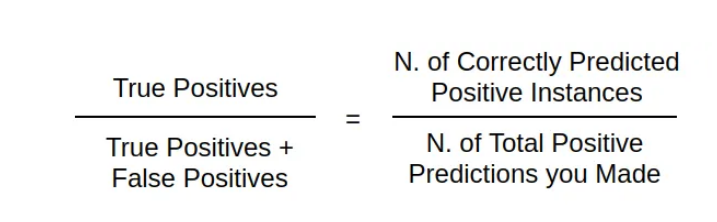
Accuracy is the ratio of correctly predicted instances to the total instances in the dataset



**Figure2. Formula of Accuracy**

**4.8.2 Precision**

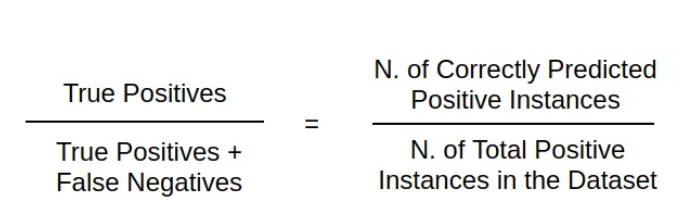
Precision is the ratio of true positive predictions to the sum of true positive and false positive predictions. It measures how many of the predicted positive cases were actually positive, focusing on the quality of the positive predictions.



**Figure3 Formula of Precision**

**4.8.3 Recall**

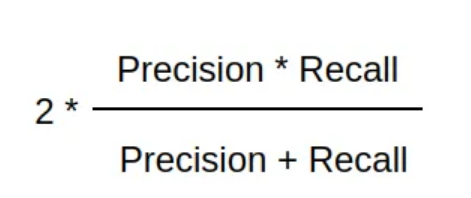
Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the sum of true positive and false negative predictions. It measures how many actual positive cases were correctly predicted by the model



**Figure4 Formula of Recall**

**4.8.4 F1-score**

The F1 Score is the harmonic mean of precision and recall. It provides a single metric that balances both the precision and recall of a model.



**Figure5 Formula of F1-Score**

5.Results