

Efficiency of oversampling methods for enhancing software defect prediction by using imbalanced data

Dr.B.Tirimula Rao
T.Karunya

Software defect prediction is important to identify defects in the early phases of software development life cycle. This early identification and thereby removal of software defects is crucial to yield a cost-effective and good quality software product.

Use of **machine learning techniques** for software defect prediction, yields biased results when applied on **imbalanced data sets**.

Use of imbalanced datasets leads to off-target predictions of the minority class, which is generally considered to be more important than the majority class. Thus, **handling imbalanced data** effectively is crucial for successful development of a competent defect prediction model.

Class Imbalance Problem

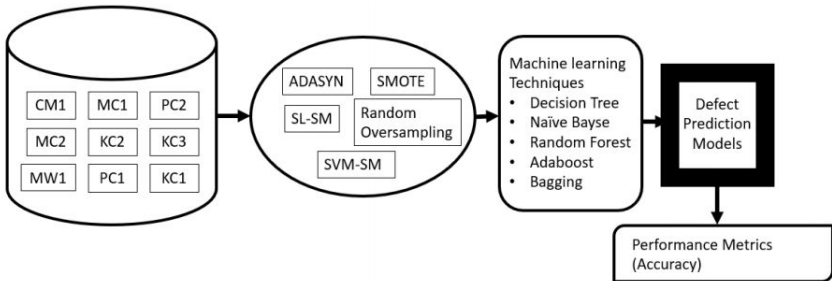
Datasets with a nonuniform distribution of class values suffer from class imbalance problems. For example, consider a dataset whose 95% class values belong to one class and only 5% class values belong to another class; this dataset is imbalanced. Only few instances of minority class values are reported for hundreds, thousands, or millions of instances of majority classes or class values. Thus, when ML classifiers are implemented on these datasets, the prediction capability of models decreases, and the model provides off-target results.

This project evaluates the effectiveness of five machine learning classifiers for software defect prediction on nine imbalanced NASA datasets by application of sampling methods.

We investigate five existing oversampling methods, which replicate the instances of minority class.

- ① **What is the effect of application of oversampling methods for developing defect prediction models using ML techniques?**
- ② **Does balancing of datasets using oversampling methods improve the performance of ML techniques for defect prediction? If yes, what is the extent of improvement in the performance of ML techniques for software defect prediction**
- ③ **Which is the best oversampling method to improve the performance of ML techniques for software defect prediction in this study?**

WORKFLOW:PROJECT



DATASETS

| Dataset | Number of Attributes | Number of Records | Class Values | |
|---------|----------------------|-------------------|--------------|-------|
| | | | True | False |
| CM1 | 22 | 498 | 49 | 449 |
| KC1 | 21 | 2109 | 326 | 1783 |
| KC2 | 22 | 522 | 107 | 415 |
| KC3 | 39 | 194 | 36 | 158 |
| MC1 | 38 | 1988 | 46 | 1942 |
| MC2 | 39 | 125 | 44 | 81 |
| MW1 | 37 | 253 | 27 | 226 |
| PC1 | 22 | 1109 | 77 | 1032 |
| PC2 | 36 | 745 | 16 | 729 |

Environment setting

The proposed methods are implemented on a laptop with an Intel(R) Core(TM) i7-8565U CPU@ 1.80 GHz 1992 MHz, 4 core(s), eight logical processor(s), and the Microsoft Windows 10 Pro 64-bit operating system.

A \times 64-based processor with Jupyter notebook 6.1.4, python 3.8.5., sklearn 1.0.2 (ML classifiers are imported from sklearn lib), imblearn 0.8.1 (oversampling methods are obtained from imblearn), matplotlib 3.2.2, and seaborn 0.11.2 are the libraries used for plotting graphs.

- ① **HyperParameters For 5 Machine Learning Techniques over 9 datasets**
- ② **Compare performance of ML models after Oversampling:**
 - SMOTE (Synthetic Minority Oversampling Technique)
 - SL-SM (Safe-level SMOTE)
 - ADASYN (Adaptive synthetic sampling technique)
 - SVM-SMOTE (Support Vector Smote)
 - Random OverSampler
- ③ **Comparative analysis of Oversampling Techniques to find out best oversampling method to improve the performance of ML techniques for software defect pre-diction in this study**

① Machine Learning Techniques :

- Decision Tree
- Random Forest
- Naive Bayes
- Adaboost
- Bagging

② Imported machine learning techniques from various packages of sklearn :

```
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
```

Hyperparameter Tuning with GridSearchCV

It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. As the performance of a model significantly depends on the value of hyperparameters.

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.

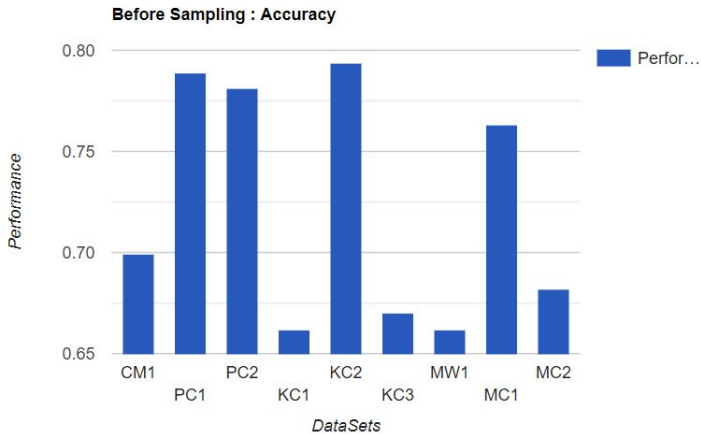
HYPER PARAMETERS

| HYPER PARAMETER TUNING | | | | | | | | | | | | | | | |
|------------------------|-----------------------------|-----------|------------------|-------------------|----------------------|--------------|------------------|-------------------|-------------|-----------------|-------------|-----------|----------------|-----------|-------------|
| Datasets | Machine Learning Techniques | | | | | | | | | | | | | | |
| | <u>Decision Tree</u> | | | | <u>Random Forest</u> | | | | | <u>Adaboost</u> | | | <u>Bagging</u> | | |
| | criteria | Max depth | Min Samples leaf | Min Samples split | Max depth | Max features | Min Samples leaf | Min Samples split | N-estimator | Base estimator | N-estimator | Algorithm | Base estimator | Max depth | Max samples |
| CM1 | gini | 2 | 2 | 4 | 80 | 2 | 3 | 10 | 100 | decision tree | 200 | SAMME.R | decision tree | 1 | 0.1 |
| KC1 | gini | 3 | 1 | 2 | 90 | 3 | 3 | 10 | 100 | Decision tree | 100 | SAMME.R | Decision tree | 5 | 0.5 |
| KC2 | gini | 5 | 2 | 2 | 80 | 3 | 4 | 8 | 100 | Decision tree | 300 | SAMME.R | Decision tree | 1 | 0.1 |
| KC3 | gini | 2 | 2 | 4 | 80 | 2 | 3 | 10 | 100 | Decision tree | 200 | SAMME.R | Decision tree | 5 | 0.2 |
| MC1 | gini | 2 | 1 | 4 | 80 | 2 | 3 | 8 | 100 | Decision tree | 300 | SAMME.R | Decision tree | 5 | 0.5 |
| MC2 | entropy | 3 | 1 | 3 | 80 | 3 | 3 | 10 | 100 | decision tree | 300 | SAMMER | decision tree | 1 | 0.2 |
| MW1 | gini | 3 | 1 | 4 | 80 | 2 | 3 | 8 | 100 | decision tree | 200 | SAMMER | decision tree | 5 | 0.1 |
| PC1 | entropy | 5 | 4 | 4 | 80 | 3 | 4 | 8 | 100 | decision tree | 100 | SAMMER | decision tree | 5 | 0.5 |
| PC2 | gini | 1 | 1 | 2 | 80 | 2 | 3 | 8 | 100 | decision tree | 100 | SAMMER | decision tree | 1 | 0.05 |

① Performance Accuracy of Machine Learning Techniques Before OverSampling :

| NO_SAMPLING | | | | | | | | | |
|----------------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| <i>ML Technique</i> | <i>CM1</i> | <i>PC1</i> | <i>PC2</i> | <i>KC1</i> | <i>KC2</i> | <i>KC3</i> | <i>MW1</i> | <i>MC1</i> | <i>MC2</i> |
| Decision tree | 0.59 | 0.719 | 0.448 | 0.616 | 0.704 | 0.653 | 0.449 | 0.521 | 0.698 |
| Naïve Bayes | 0.69 | 0.844 | 0.836 | 0.701 | 0.825 | 0.736 | 0.716 | 0.883 | 0.717 |
| Random Forest | 0.76 | 0.768 | 0.877 | 0.633 | 0.832 | 0.661 | 0.728 | 0.708 | 0.702 |
| Adaboost | 0.71 | 0.793 | 0.914 | 0.669 | 0.784 | 0.573 | 0.711 | 0.842 | 0.616 |
| Bagging | 0.72 | 0.820 | 0.828 | 0.689 | 0.825 | 0.729 | 0.705 | 0.860 | 0.676 |
| Avg | 0.699 | 0.789 | 0.781 | 0.662 | 0.794 | 0.670 | 0.662 | 0.763 | 0.682 |

NoSampling : Performance



OverSampling Techniques

① OverSampling Techniques :

- SMOTE (Synthetic Minority Oversampling Technique)
- SL-SM (Safe-level SMOTE)
- ADASYN (Adaptive synthetic sampling technique)
- SVM-SMOTE (Support Vector Smote)
- Random OverSampler

② Imported OverSampling techniques from imblearn.overSampling package :

```
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import SVMSMOTE
from imblearn.over_sampling import RandomOverSampler
from imblearn.over_sampling import SLSM
from imblearn.over_sampling import ADASYN
```

Comparison: OverSampling Techniques

① Comparing OverSampling Techniques :

| <i>OverSampling Technique</i> | <i>Random Oversampler</i> | <i>SMOTE</i> | <i>ADASYN</i> | <i>SL-SM</i> | <i>SVM-SMOTE</i> |
|---|--|---|--|---|---|
| Minority class instances | Duplicates the values of minority classes | Between random data points and k-nearest neighbors | Using SMOTE and Borderline Smote | Done around a larger safe level | Done within the parameters of support vectors |
| Number of minority class instances | Equal to the number of majority class values | Proportional to the number of instances in majority class | Proportional to the Density distribution of minority class | Proportional to the number of instances in majority class | Proportional to the number of instances in majority class |

Datasets: After OverSampling

| Dataset | Class Values | Before OverSampling | After OverSampling | | | | |
|---------|--------------|---------------------|--------------------|--------|-------|-----------|--------------------|
| | | | SMOTE | ADASYN | SL-SM | SVM-SMOTE | Random Oversampler |
| CM1 | True | 39 | 359 | 373 | 359 | 141 | 359 |
| | False | 359 | 359 | 359 | 359 | 359 | 359 |
| PC1 | True | 824 | 824 | 824 | 824 | 824 | 824 |
| | False | 63 | 824 | 803 | 824 | 824 | 824 |
| PC2 | True | 584 | 584 | 584 | 584 | 584 | 584 |
| | False | 12 | 584 | 584 | 584 | 584 | 584 |
| KC1 | True | 256 | 1431 | 1415 | 1431 | 1431 | 1431 |
| | False | 1431 | 1431 | 1431 | 1431 | 143 | 1431 |
| KC2 | True | 94 | 323 | 322 | 323 | 323 | 323 |
| | False | 323 | 323 | 323 | 323 | 323 | 323 |
| KC3 | True | 33 | 33 | 122 | 122 | 122 | 122 |
| | False | 122 | 122 | 125 | 122 | 122 | 122 |
| MC1 | True | 38 | 1552 | 1550 | 1552 | 826 | 1552 |
| | False | 1552 | 1552 | 1552 | 1552 | 1552 | 1552 |
| MC2 | True | 67 | 67 | 67 | 67 | 67 | 67 |
| | False | 33 | 67 | 61 | 67 | 67 | 67 |
| MW1 | True | 178 | 178 | 178 | 178 | 178 | 178 |
| | False | 24 | 178 | 173 | 178 | 109 | 178 |

Performance:After OverSampling

- ① **Performance Accuracy of datasets after application of Oversampling Techniques :**

| CM1 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE |
|-----|----------------|-------|--------|-------|-----------|
| Dt | 0.68 | 0.63 | 0.77 | 0.76 | 0.84 |
| Nb | 0.87 | 0.84 | 0.83 | 0.85 | 0.85 |
| Rf | 0.87 | 0.85 | 0.87 | 0.86 | 0.89 |
| Ada | 0.85 | 0.84 | 0.87 | 0.80 | 0.86 |
| bg | 0.71 | 0.72 | 0.66 | 0.74 | 0.87 |

Performance:After OverSampling

| MC2 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE | PC1 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE |
|-----|----------------|-------|--------|-------|-----------|-----|----------------|-------|--------|-------|-----------|
| Dt | 0.60 | 0.64 | 0.64 | 0.56 | 0.68 | Dt | 0.72 | 0.84 | 0.85 | 0.72 | 0.73 |
| Nb | 0.68 | 0.68 | 0.68 | 0.68 | 0.68 | Nb | 0.90 | 0.88 | 0.89 | 0.90 | 0.89 |
| Rf | 0.64 | 0.64 | 0.60 | 0.60 | 0.68 | Rf | 0.95 | 0.94 | 0.94 | 0.95 | 0.95 |
| Ada | 0.56 | 0.56 | 0.64 | 0.64 | 0.64 | Ada | 0.89 | 0.89 | 0.92 | 0.92 | 0.95 |
| bg | 0.56 | 0.72 | 0.68 | 0.64 | 0.56 | bg | 0.74 | 0.76 | 0.77 | 0.77 | 0.91 |

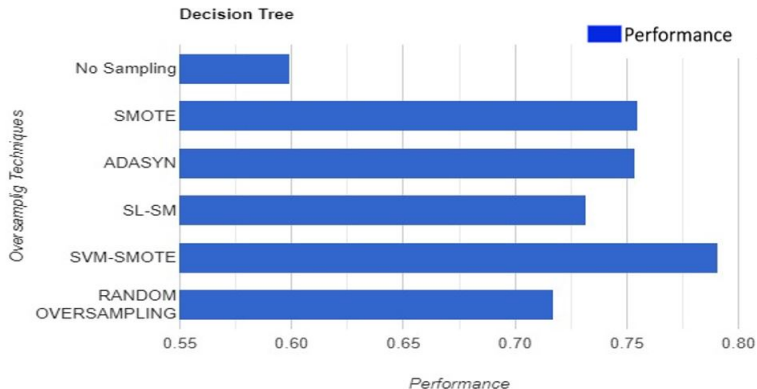
| MW1 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE | PC2 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE |
|-----|----------------|-------|--------|-------|-----------|-----|----------------|-------|--------|-------|-----------|
| Dt | 0.84 | 0.82 | 0.86 | 0.70 | 0.88 | Dt | 0.79 | 0.83 | 0.83 | 0.97 | 0.79 |
| Nb | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | Nb | 0.68 | 0.61 | 0.59 | 0.80 | 0.68 |
| Rf | 0.86 | 0.90 | 0.86 | 0.88 | 0.88 | Rf | 0.97 | 0.96 | 0.97 | 0.97 | 0.97 |
| Ada | 0.88 | 0.84 | 0.86 | 0.86 | 0.88 | Ada | 0.97 | 0.95 | 0.94 | 0.96 | 0.97 |
| bg | 0.80 | 0.78 | 0.78 | 0.82 | 0.82 | bg | 0.79 | 0.77 | 0.78 | 0.97 | 0.79 |

Performance:After OverSampling

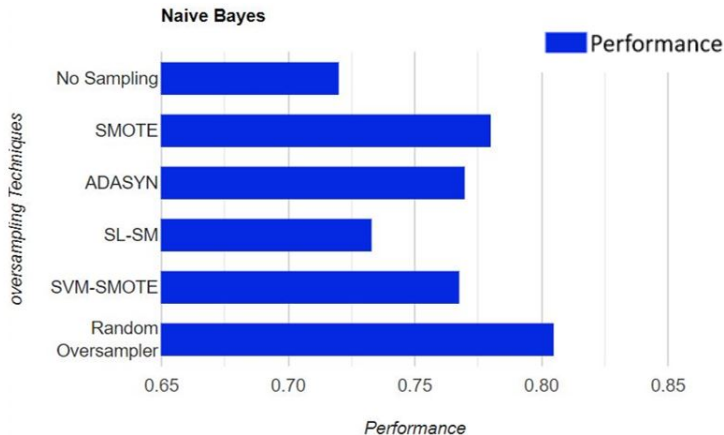
| KC2 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE | KC3 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE |
|-----|----------------|-------|--------|-------|-----------|-----|----------------|-------|--------|-------|-----------|
| Dt | 0.68 | 0.80 | 0.80 | 0.81 | 0.80 | Dt | 0.64 | 0.87 | 0.64 | 0.64 | 0.87 |
| Nb | 0.89 | 0.89 | 0.89 | 0.89 | 0.89 | Nb | 0.69 | 0.69 | 0.69 | 0.69 | 0.79 |
| Rf | 0.79 | 0.82 | 0.80 | 0.82 | 0.82 | Rf | 0.79 | 0.76 | 0.79 | 0.74 | 0.79 |
| Ada | 0.80 | 0.77 | 0.82 | 0.79 | 0.82 | Ada | 0.79 | 0.82 | 0.79 | 0.71 | 0.79 |
| bg | 0.79 | 0.79 | 0.80 | 0.78 | 0.78 | bg | 0.61 | 0.64 | 0.56 | 0.64 | 0.64 |

| MC1 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE | KC1 | Random Sampler | SMOTE | ADASYN | SL-SM | SVM-SMOTE |
|-----|----------------|-------|--------|-------|-----------|-----|----------------|-------|--------|-------|-----------|
| Dt | 0.86 | 0.79 | 0.77 | 0.78 | 0.85 | Dt | 0.65 | 0.58 | 0.63 | 0.65 | 0.68 |
| Nb | 0.93 | 0.88 | 0.85 | 0.22 | 0.47 | Nb | 0.81 | 0.80 | 0.79 | 0.77 | 0.81 |
| Rf | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | Rf | 0.80 | 0.80 | 0.81 | 0.81 | 0.80 |
| Ada | 0.93 | 0.95 | 0.95 | 0.96 | 0.97 | Ada | 0.72 | 0.79 | 0.77 | 0.79 | 0.80 |
| bg | 0.77 | 0.74 | 0.75 | 0.81 | 0.85 | bg | 0.65 | 0.67 | 0.63 | 0.66 | 0.68 |

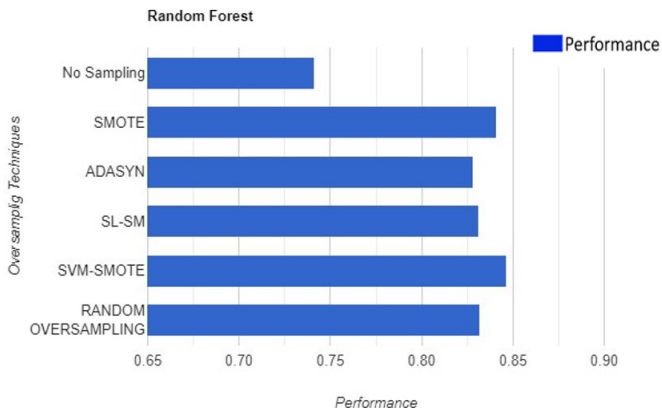
Comparison: Performance before and after OverSampling



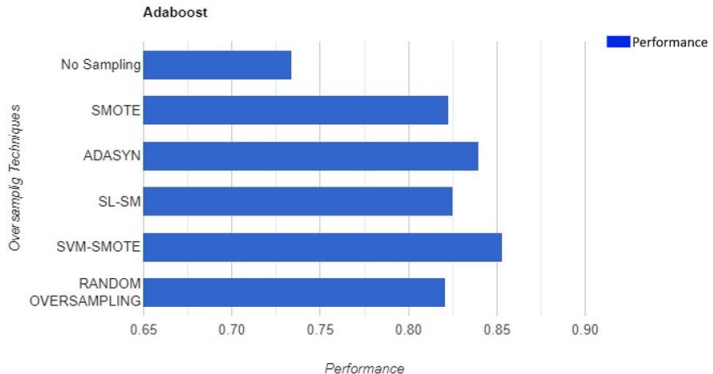
Comparison: Performance before and after OverSampling



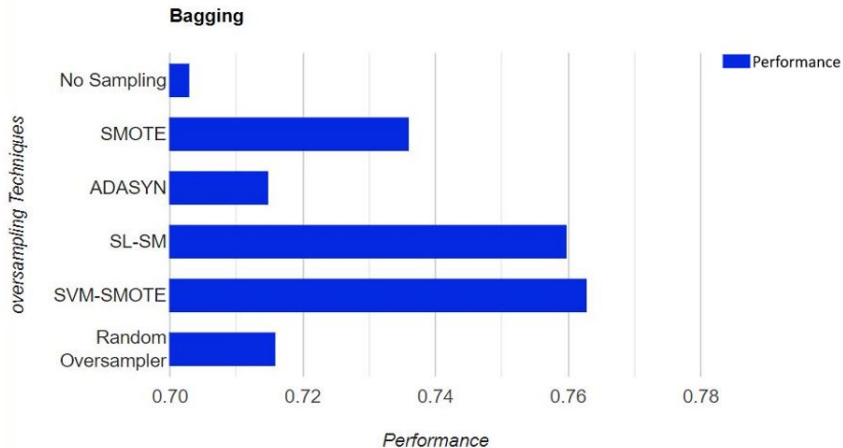
Comparison: Performance before and after OverSampling



Comparison: Performance before and after OverSampling

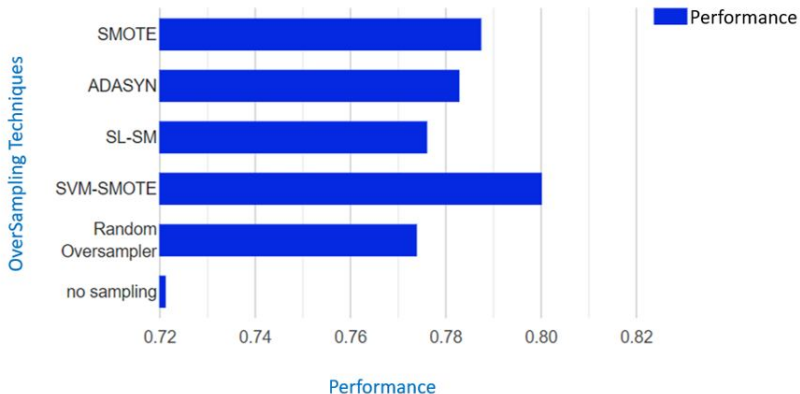


Comparison: Performance before and after OverSampling



Comparison: OverSampling Techniques

Performance of oversampling Techniques



- ① **What is the effect of application of oversampling methods for developing defect prediction models using ML techniques?**
 - Before application of Oversampling Techniques (that is NoSampling)the performance of machine learning techniques for defect prediction models is around 60-70 percent and when oversampling techniques are applied the performance increased to 85-90 percent.
- ② **Does balancing of datasets using oversampling methods improve the performance of ML techniques for defect prediction? If yes, what is the extent of improvement in the performance of ML techniques for software defect prediction**
 - Yes,balancing of datasets using oversampling methods improves the performance of ML techniques. the extent of improvement in the performance of ML techniques is 15-20 percent.

- ① **Which is the best oversampling method to improve the performance of ML techniques for software defect prediction in this study?**
 - From the results from performance analysis, it can be noted that SVM-SMOTE both SMOTE are perhaps the best oversampling method to improve the performance of ML techniques for software defect prediction in this study.