PROJECT 2

DATA WAREHOUSING AND DATA MINING

CS5483

Predicting Software Defects in Imbalanced Data

DAI Jingzhi **xxxxx**

KABIR Md Alamgir ***xxxxx***

PETINRIN Olutomilayo Olayemi ***xxxxx***

RAHAMAN Saifur **xxxxx**

CITY UNIVERSITY OF HONG KONG

Table of contents

1. Introduction

Software defect prediction (SDP) is essential to produce quality software. Prediction models are developed by using the historical data [1]. There have been developed various types of classification models over the years [2] [3] [5]. The classification models predict the defective modules in a software. By doing so, the testing resources can be utilized in an efficient way and project manager can do the proper utilization of scarce software testing resources.

However, the performance of the defect prediction models depends on the training set. Most of the conventional classifiers work on the balanced datasets to maximize the accuracy of prediction models. The historical software defect datasets are mostly imbalanced [4] [5]. Model trained on imbalanced training set will unintentionally predicts defective modules of a new software [6]. In software development scenario, project manager are likely more interested to such model which can give more accurate and high prediction accuracy on the minority class instances (i.e., defective modules).

In the situation of skewed class distribution, the most popular approaches are data sampling methods [6]. Data sampling methods are applied to the training sets before trained it to put the model construction. Synthetic Minority Over-sampling TEchnique (SMOTE) [6] is one of the popular techniques to create synthetic examples in minority classes. SMOTE adapts KNN approaches to create synthetic instances.

In this experimental study, we try to improve the prediction performance by taking into consideration the issue of class imbalance and using the attribute selection method with cross-validation techniques.

2. Experimental Settings

Benchmark Datasets

In this experimental study, we take nine projects from NASA metrics program [1], considered as benchmark datasets shown in Table 1. Every dataset contains defective and defect-free modules. Target class has two levels: defects {false,true} which means that module has/has not one or more reported defects. Table 1 shows that the datasets are skewed. Such as, PC1 dataset is prepared from flight software for earth satellite and McCabe and Halstead metrics are used to extract the source code to prepare the datasets. This dataset contains 22 attributes. Among the 1109 modules, 1032 modules are defective (93.06%) which executes the imbalance situation. PC2 contains 5589 from where 5566 models are defective and prepared with 37 attributes.

Table 1: 9 benchmark datasets used in this study

Dataset	Number of Modules	Defective Modules	Attribute
pc1	1109	1032	22
pc2	5589	5566	37
pc3	1563	1403	38
pc4	1458	1280	38
mc1	9466	9388	39
mc2	161	109	40
kc1	2109	1783	22
kc2	522	415	22
kc3	458	415	40

Evaluation measures

In experiments, for evaluating the performance of our considered technique, we investigate the defective and non-defective modules. Table 2 shows the confusion metrics. To measure the effectiveness of our approach, we consider probability of detection (i.e., recall). Recall means how much of the defective modules were actually defective. Higher recall denotes better performance. We didn't consider precision and f-measure in this study because it were reported as unsuitable for assessing the performance of imbalance datasets, as instructed by [1]. For real results, it is recommended to keep above the values of 75% by [7]. For more understanding, we also keep the values of accuracy and the model error-rate. The equation of recall, accuracy, and error-rate are given below.

Table 2: Defect Prediction Metric

	Defective Modules	Non-defective modules
Predict as defective	TP	FP
Predict as non-defective	FN	TN

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Error - rate = 1 - Accuracy$$

$$Recall\ (pd)\ = \frac{TP}{TP\ +\ FN}$$

Experimental setup

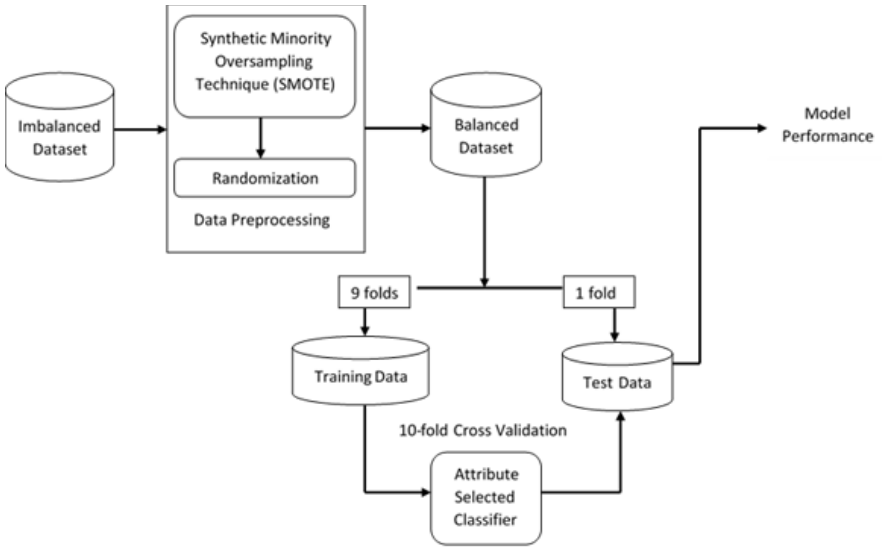


Figure 1: Experimental Setup

In our experiment, SMOTE filter is applied to the imbalanced dataset to add more instances bearing the class label which has a low count. SMOTE is applied using nearest neighbor. Since the newly added instances are appended to the original dataset, the new instances having the same class label are close together, and this situation affects the result in the case of cross validation. Hence, randomization filter is applied to shuffle the instances and have the class values randomly placed. Figure 1 portraits the outline of the experiment. The classifier KNN is used in this experiment as it classifiers an unknown instance by considering its nearest neighbors. In the study of [8], KNN achieved the best results in the class imbalance situation among all the considered classifiers.

An attribute selected classifier, to select the best attributes, is then applied on the new balanced dataset using a 10-fold cross validation method to generate the final performance of the model.

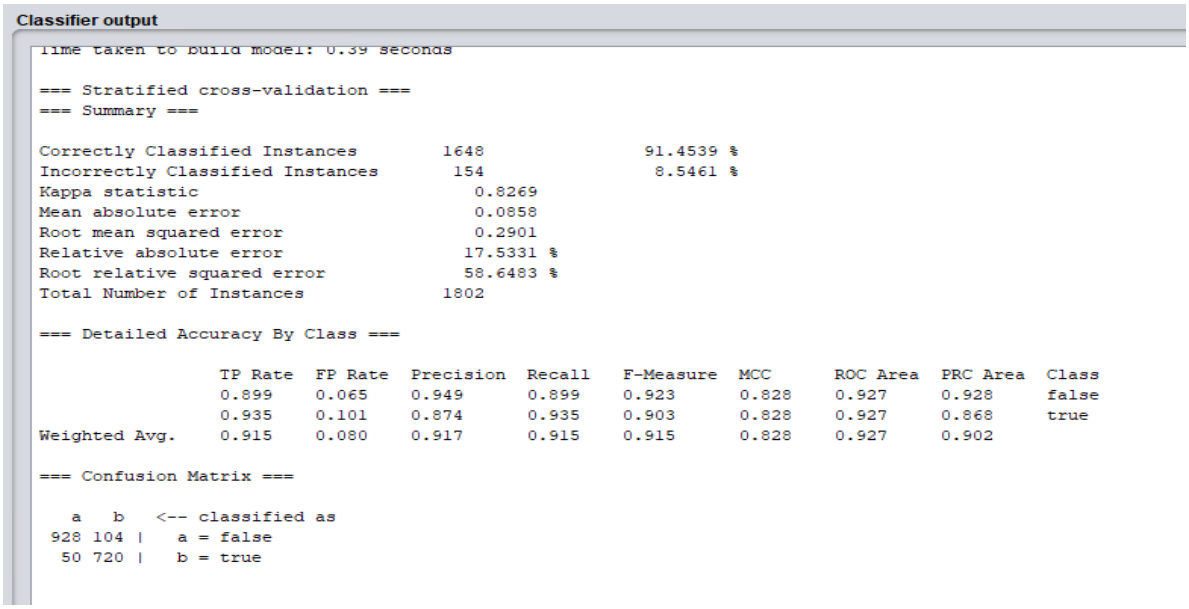
3. Results and Discussion

In this section, we describe the process of experiment for each dataset. For the dataset, PC1, the instances are increased to 1802 from 1109. PC1 dataset contains 22 attributes described in Table 1.
Details are given below:

1. Class Attribute (Defects) with 1032 FALSE labels, and 77 TRUE labels
2. Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 770
3. SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100, 100, and 25.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search techniques, the following result was derived:



Classifier output

```
Time taken to build model: 0.39 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1648               91.4539 %
Incorrectly Classified Instances       154                8.5461 %
Kappa statistic                          0.8269
Mean absolute error                      0.0858
Root mean squared error                  0.2901
Relative absolute error                 17.5331 %
Root relative squared error             58.6483 %
Total Number of Instances             1802

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.899    0.065    0.949      0.899    0.923      0.828    0.927     0.928     false
                 0.935    0.101    0.874      0.935    0.903      0.828    0.927     0.868     true
Weighted Avg.    0.915    0.080    0.917      0.915    0.915      0.828    0.927     0.902

=== Confusion Matrix ===

   a   b   <-- classified as
 928 104 |   a = false
  50 720 |   b = true
```

Figure 2: WEKA Classifier Output window for the PC1 dataset

After applying SMOTE, the total instances increases to 9982 from 5589 of the PC2 dataset that contains 37 attributes. Details are:

1. Class Attribute (c) with 5566 FALSE labels, and 23 TRUE labels
2. Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 4416
3. SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100, 100, 100, 100, 100, 100 and 50.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search techniques, the following result was derived:

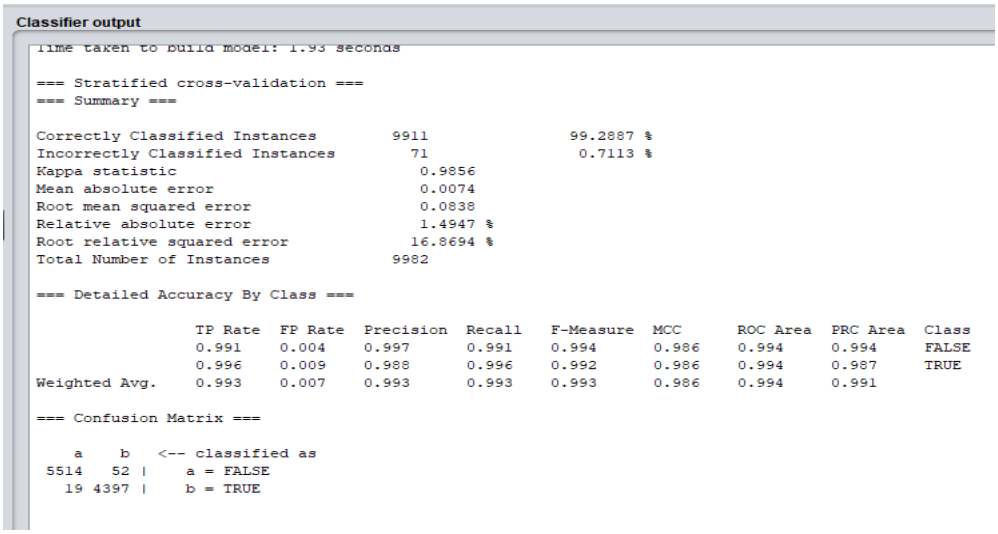


Figure 3: WEKA Classifier Output window for the PC2 dataset

Details for the Dataset PC3:

1. New Instances after SMOTE: 2203
2. Class Attribute (c) with 1403 FALSE labels, and 160 TRUE labels
3. Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 800
4. SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100 and 25.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search techniques, the following result was derived:

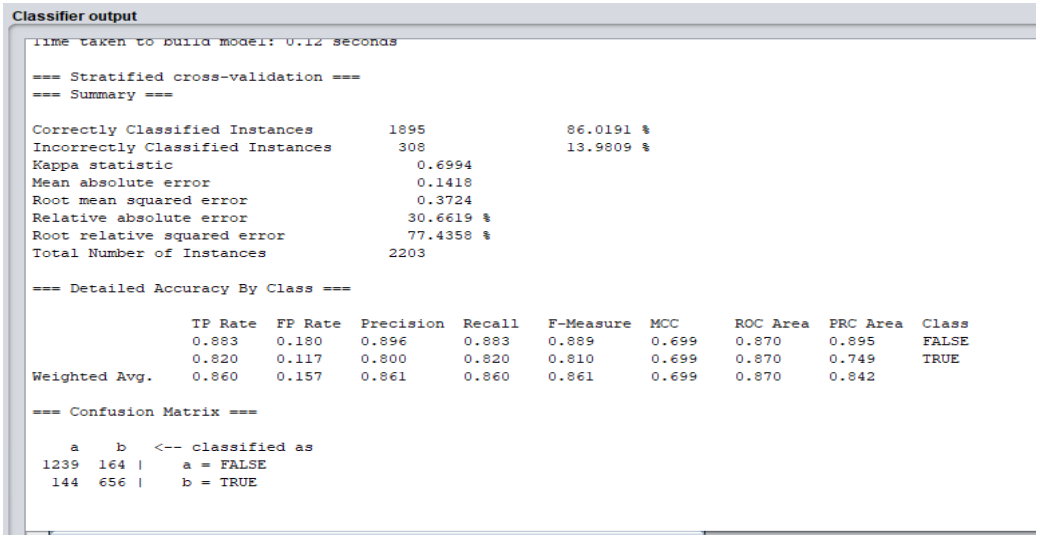


Figure 4: WEKA Classifier Output window for the PC3 dataset

Details for the Dataset PC4:

1. New Instances after SMOTE: 2170
2. Class Attribute (c) with 1280 FALSE labels, and 178 TRUE labels
3. Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 890
4. SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100 and 25.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:

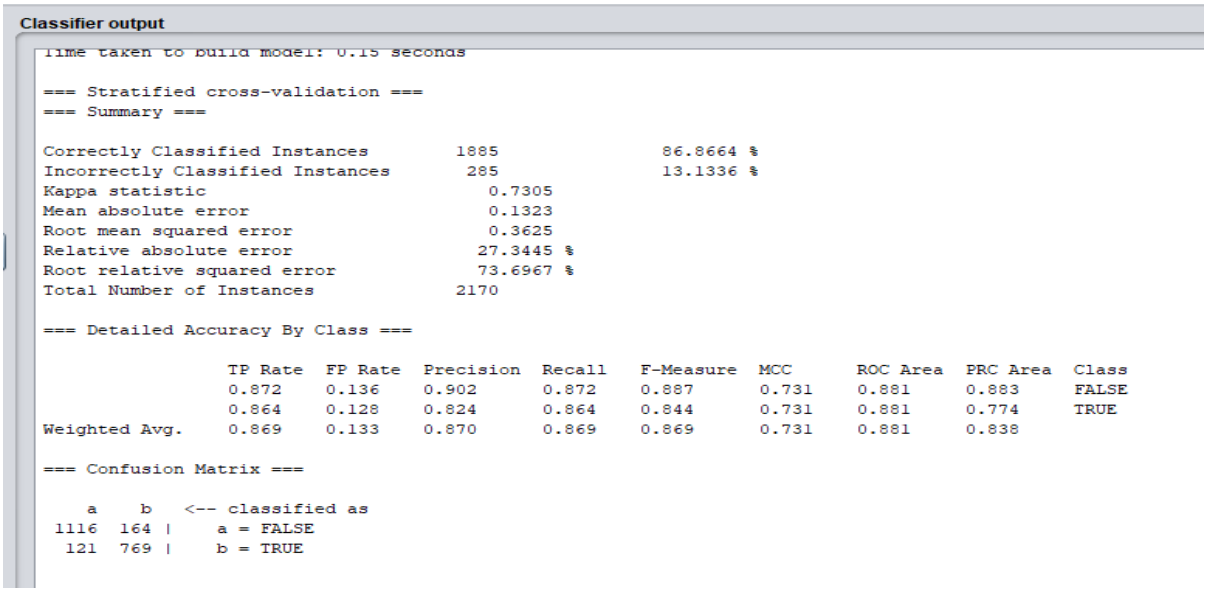


Figure 5: PC4

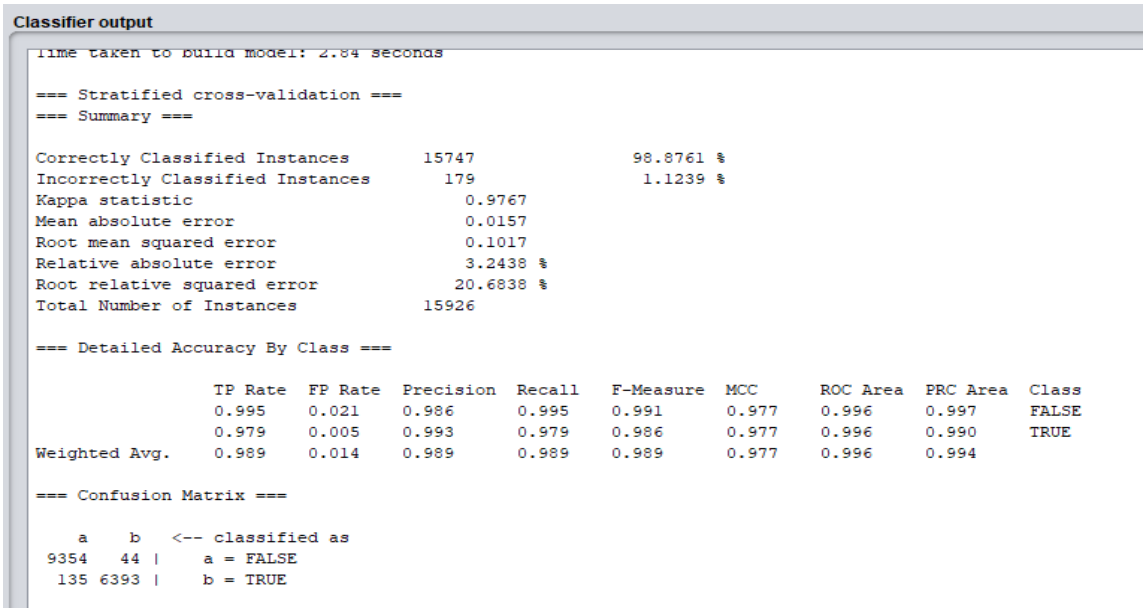
Instances: 9466
New Instances after SMOTE: 15926
Attributes: 39
Class Attribute (c) with 9398 FALSE labels, and 68 TRUE labels

Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 6528

SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100, 100, 100, 100, 100 and 50.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:

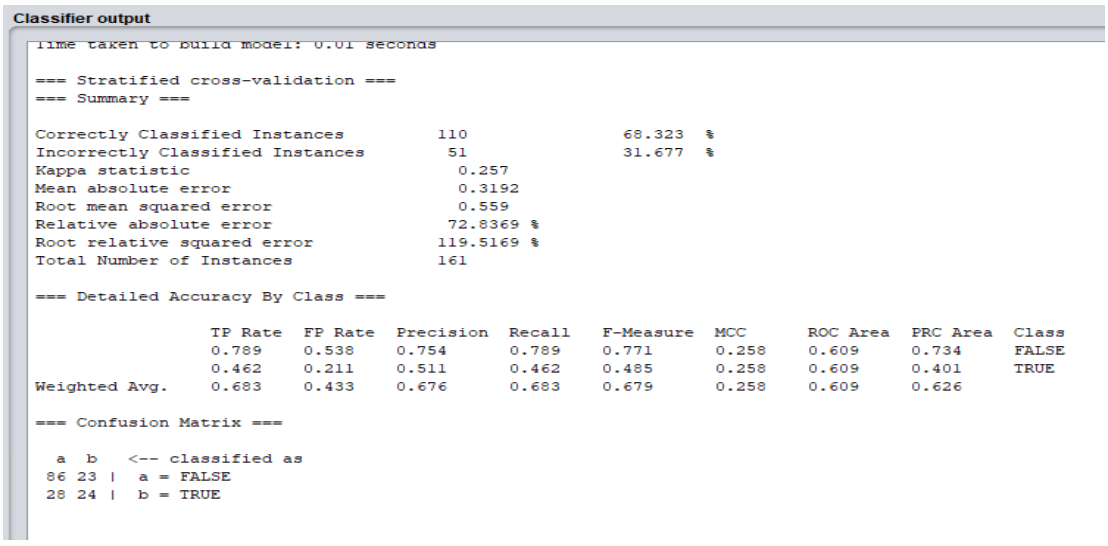


```
Classifier output
Time taken to build model: 2.84 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       15747               98.8761 %
Incorrectly Classified Instances       179                1.1239 %
Kappa statistic                          0.9767
Mean absolute error                      0.0157
Root mean squared error                  0.1017
Relative absolute error                  3.2438 %
Root relative squared error             20.6838 %
Total Number of Instances            15926

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.995    0.021    0.986      0.995    0.991      0.977    0.996     0.997     FALSE
                 0.979    0.005    0.993      0.979    0.986      0.977    0.996     0.990     TRUE
Weighted Avg.    0.989    0.014    0.989      0.989    0.989      0.977    0.996     0.994

=== Confusion Matrix ===

    a    b   <-- classified as
 9354   44 |    a = FALSE
  135 6393 |    b = TRUE
```

Figure 6: MC1

Instances: 161
Attributes: 40
Class Attribute (c) with 109 FALSE labels, and 52 TRUE labels

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:



```
Classifier output
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         110               68.323  %
Incorrectly Classified Instances        51               31.677  %
Kappa statistic                          0.257
Mean absolute error                      0.3192
Root mean squared error                  0.559
Relative absolute error                 72.8369 %
Root relative squared error            119.5169 %
Total Number of Instances              161

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.789    0.538    0.754      0.789    0.771      0.258    0.609     0.734     FALSE
                 0.462    0.211    0.511      0.462    0.485      0.258    0.609     0.401     TRUE
Weighted Avg.    0.683    0.433    0.676      0.683    0.679      0.258    0.609     0.626

=== Confusion Matrix ===

  a  b   <-- classified as
 86 23 |  a = FALSE
 28 24 |  b = TRUE
```

Figure 7: MC2 Best attribute selection

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using Naïve Bayes as the classifier, Information Gain / Gain Ratio as the evaluator, and Ranker technique, the following result was derived:

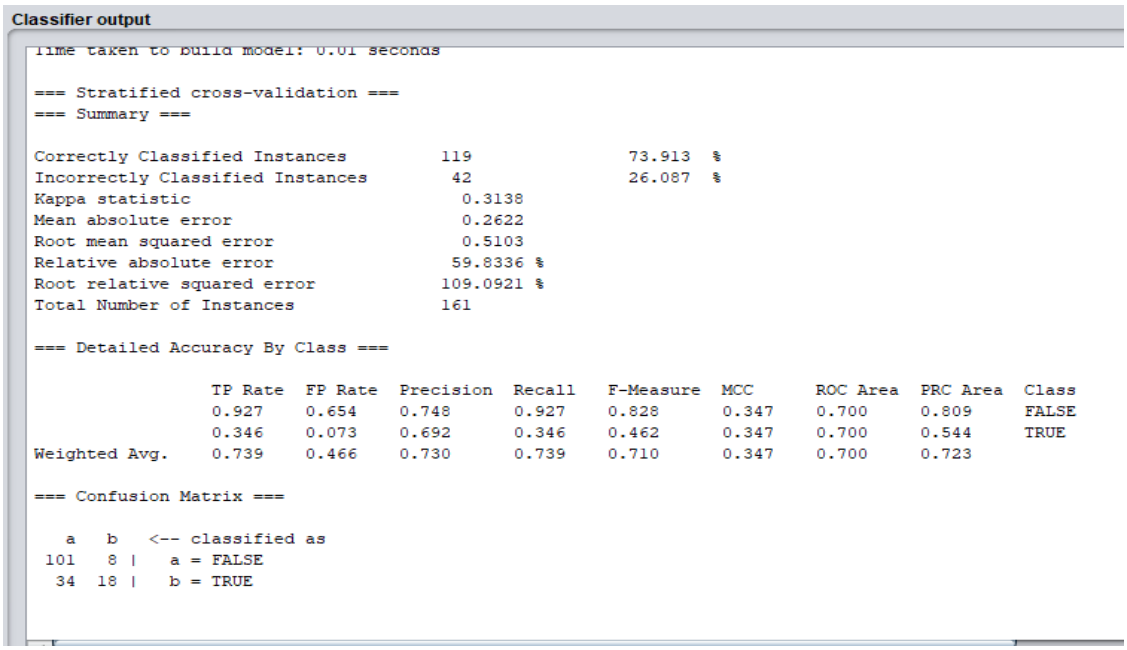


Figure 8: MC2

Dataset: KC1
Instances: 2109
New Instances after SMOTE: 3087
Attributes: 22
Class Attribute (defects) with 1783 FALSE labels, and 326 TRUE labels

Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 1304

SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100 and100.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:

Classifier output

```
Time taken to build model: 0.08 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2582               83.6411 %
Incorrectly Classified Instances       505               16.3589 %
Kappa statistic                          0.6624
Mean absolute error                      0.1816
Root mean squared error                  0.4007
Relative absolute error                 37.2105 %
Root relative squared error             81.118  %
Total Number of Instances             3087

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.877    0.219    0.845      0.877    0.861      0.663    0.822     0.825     false
                 0.781    0.123    0.823      0.781    0.801      0.663    0.822     0.764     true
Weighted Avg.    0.836    0.179    0.836      0.836    0.836      0.663    0.822     0.799

=== Confusion Matrix ===

    a    b   <-- classified as
 1564  219 |    a = false
  286 1018 |    b = true
```

Figure 9: KC1 classifier output window

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, Gain Ratio as the evaluator, and Ranker technique, the following result was derived:

Classifier output

```
Time taken to build model: 0.07 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2708               87.7227 %
Incorrectly Classified Instances       379               12.2773 %
Kappa statistic                          0.7491
Mean absolute error                      0.1301
Root mean squared error                  0.3489
Relative absolute error                 26.6648 %
Root relative squared error             70.6344 %
Total Number of Instances             3087

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.886    0.134    0.900      0.886    0.893      0.749    0.896     0.909     false
                 0.866    0.114    0.847      0.866    0.856      0.749    0.896     0.820     true
Weighted Avg.    0.877    0.126    0.878      0.877    0.877      0.749    0.896     0.871

=== Confusion Matrix ===

    a    b   <-- classified as
 1579  204 |    a = false
  175 1129 |    b = true
```

Figure 10: KC1 classifier output window

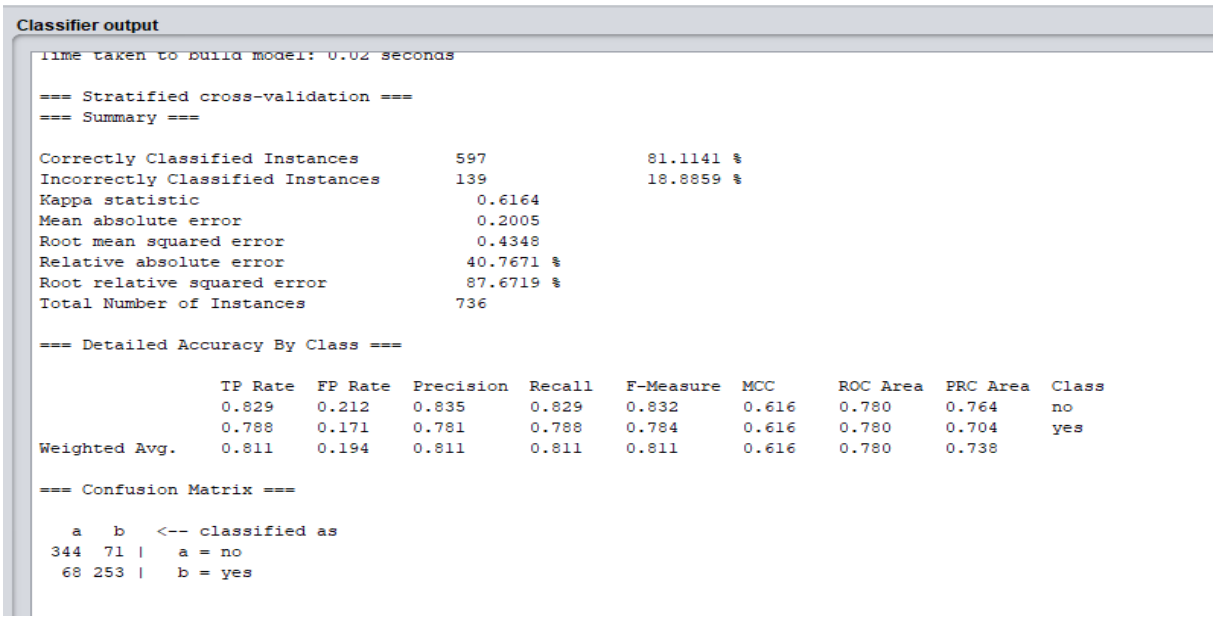
Dataset: KC2
Instances: 522
New Instances after SMOTE: 736
Attributes: 22
Class Attribute (problems) with 415 NO labels, and 107 YES labels

Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with YES label to 321

SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100 and 50.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:



Classifier output

```
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         597               81.1141 %
Incorrectly Classified Instances       139               18.8859 %
Kappa statistic                          0.6164
Mean absolute error                      0.2005
Root mean squared error                  0.4348
Relative absolute error                 40.7671 %
Root relative squared error             87.6719 %
Total Number of Instances              736

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.829    0.212    0.835      0.829    0.832      0.616    0.780     0.764     no
                 0.788    0.171    0.781      0.788    0.784      0.616    0.780     0.704     yes
Weighted Avg.    0.811    0.194    0.811      0.811    0.811      0.616    0.780     0.738

=== Confusion Matrix ===

   a   b   <-- classified as
 344  71 |   a = no
  68 253 |   b = yes
```

Figure 11: KC2 classifier output window

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, Gain Ratio as the evaluator, and Ranker technique, the following result was derived:

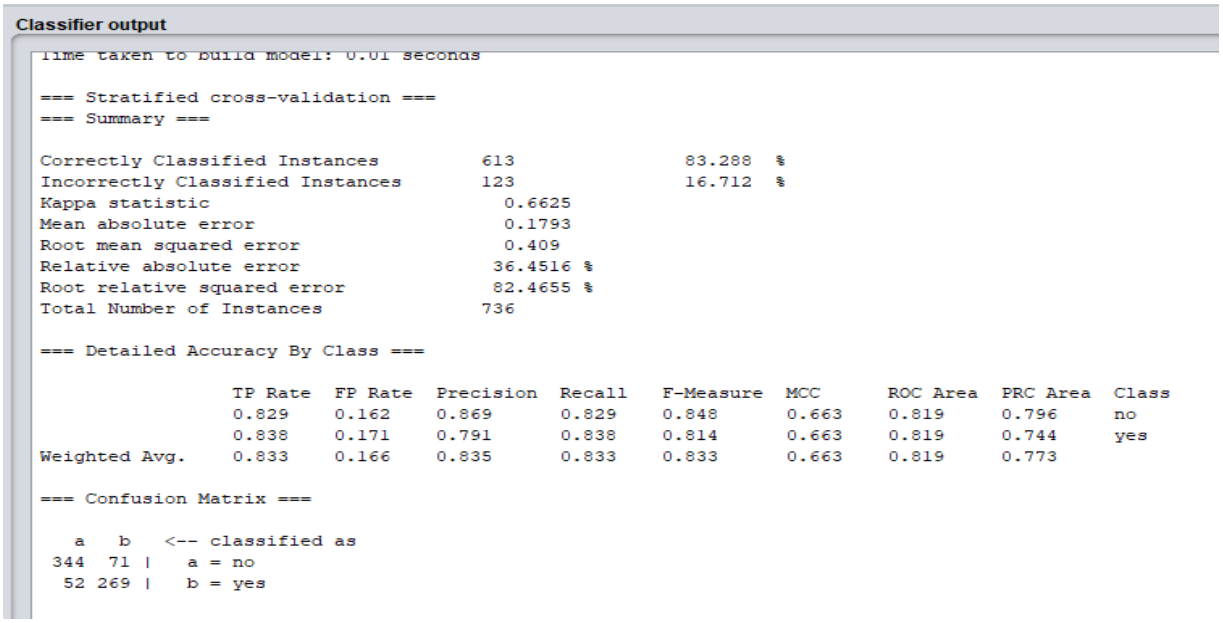


Figure 12: KC2 classifier output window

Dataset: KC3
Instances: 458
New Instances after SMOTE: 673
Attributes: 40
Class Attribute (c) with 415 FALSE labels, and 43 TRUE labels

Applied SMOTE (weka.filter.supervised.instance.SMOTE) to increase the number of instances with TRUE label to 258

SMOTE was used with nearest neighbors, and random seed set to 1 and percentage was iteratively set as 100, 100 and 50.

Afterwards, the Randomization filter (weka.filter.unsupervised.instance.Randomize) was applied to shuffle the dataset since the new instances had to be shuffled to affect the result of the cross validation.

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, CfsSubsetEval as the evaluator, and Best First Search (forward) technique, the following result was derived:

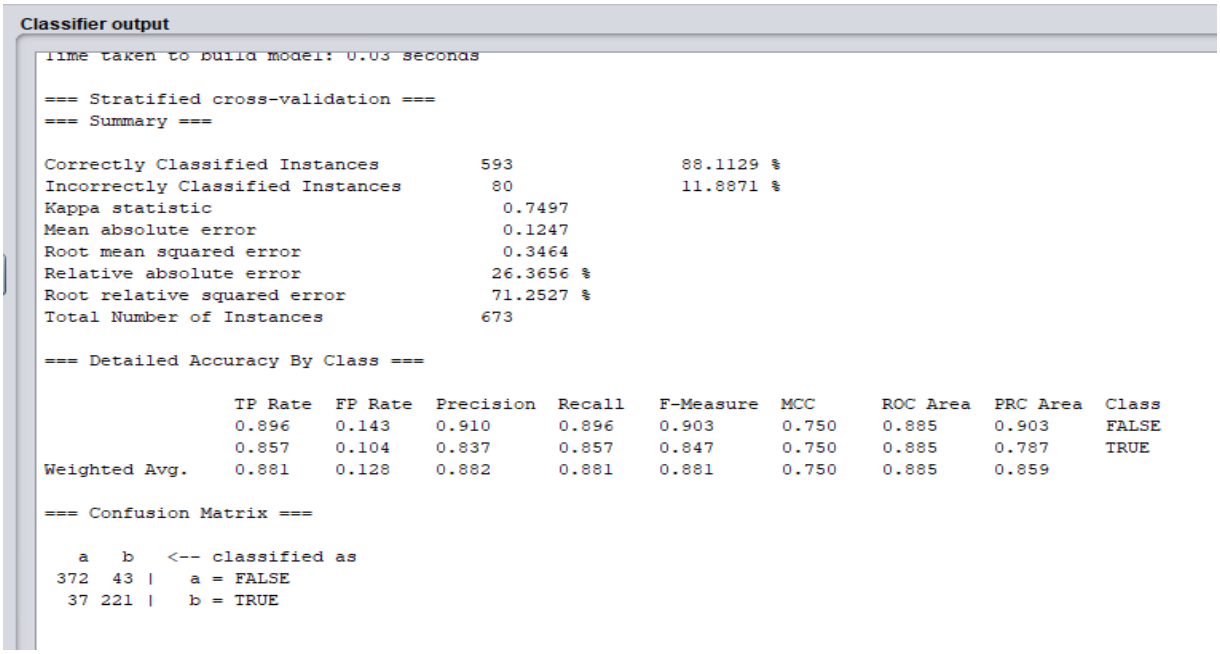


Figure 13: KC3

Using the Attribute Selected Classifier (weka.classifiers.meta.AttributeSelectedClassifier), using K-Nearest Neighbor as the classifier, Gain Ratio as the evaluator, and Ranker technique, the following result was derived:

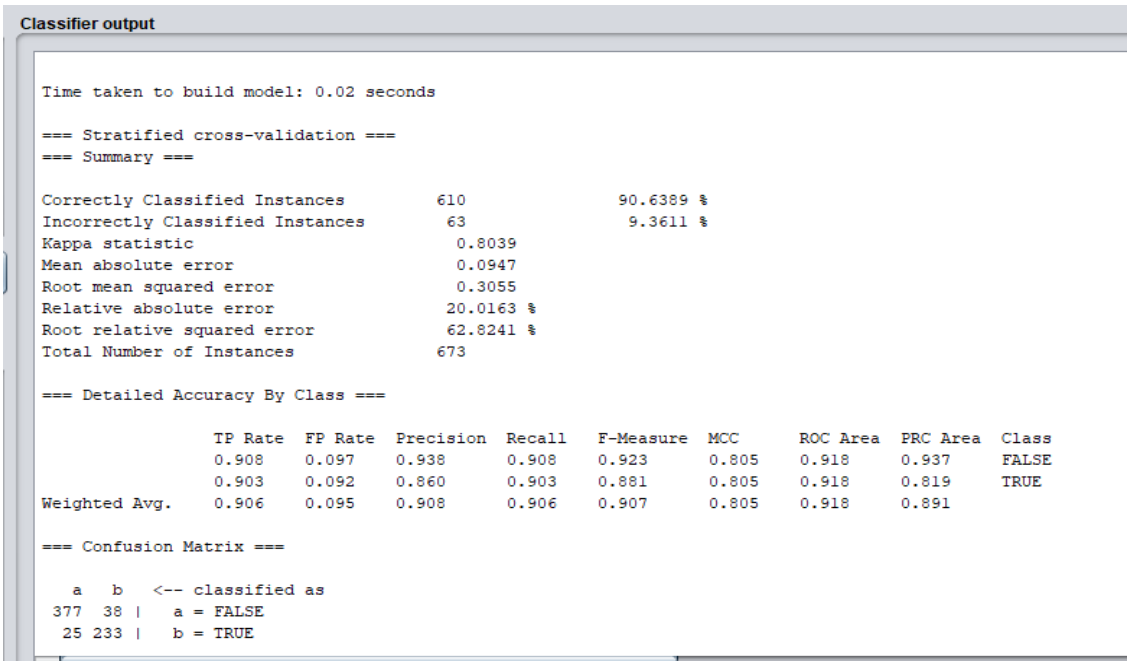


Figure 14: KC3 classifier output window

4. Conclusion

The most real software defect datasets are highly imbalanced. Resampling techniques are applied to mitigate the class imbalance issue. In this empirical study, we found that class imbalance also exists in software defect datasets. By applying the over-sampling technique, SMOTE, we try to alleviate the imbalance issue in to improve the prediction performance in SDP. We use KNN model and calculate the performance measures considered.

However, synthetic based methods tend to introduce biases towards the minority class, which increases the performance of the minority class.We will further consider the other oversampling techniques as to avoid the issue of bias in future, to improve the prediction performance of SDP.

References

1. Jing, X. Y., Wu, F., Dong, X., & Xu, B. (2016). An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems. *IEEE Transactions on Software Engineering*, *43*(4), 321-339.
2. Nam, J., Fu, W., Kim, S., Menzies, T., & Tan, L. (2017). Heterogeneous defect prediction. IEEE Transactions on Software Engineering, 44(9), 874-896.
3. Yu, Q., Jiang, S., & Zhang, Y. (2017). A feature matching and transfer approach for cross-company defect prediction. Journal of Systems and Software, 132, 366-378.
4. Wu, F., Jing, X. Y., Sun, Y., Sun, J., Huang, L., Cui, F., & Sun, Y. (2018). Cross-project and within-project semi-supervised software defect prediction: A unified approach. IEEE Transactions on Reliability, 67(2), 581-597.
5. Wan, Z., Xia, X., Hassan, A. E., Lo, D., Yin, J., & Yang, X. (2018). Perceptions, expectations, and challenges in defect prediction. IEEE Transactions on Software Engineering.
6. Bennin, K. E., Keung, J., Monden, A., Phannachitta, P., & Mensah, S. (2017, November). The significant effects of data sampling approaches on software defect prioritization and classification. In Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 364-373). IEEE Press.
7. T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," in Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009, pp. 91–100
8. Bennin, K. E., Keung, J., Phannachitta, P., Monden, A., & Mensah, S. (2017). Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Transactions on Software Engineering, 44(6), 534-550.