Journal of Applied Informatics and Computing (JAIC)
Vol.8, No.2, December 2024, pp. 472~477
e-ISSN: 2548-6861

Comparison of Oversampling Techniques on Minority Data Using Imbalance Software Defect Prediction Dataset

Deni Hidayat [1]*, Lindung Parningotan Manik [2]**
* Ilmu Komputer, Universitas Nusa Mandiri
** Research Center of Informatics, Badan Riset dan Informatika Nasional
14002445@nusamandiri.ac.id [1], lind008@brin.go.id [2]

Article Info

Article history:

Received 2024-09-30
Revised 2024-10-07
Accepted 2024-10-08

Keyword:

Software Defect Prediction, Oversampling, SMOTE.

ABSTRACT

Software Defect Prediction Dataset as a component of the Software Defect Prediction model has a very vital role. However, NASA Software Defect Prediction has a problem with imbalance in minority data. This study compares the performance of oversampling techniques in overcoming this. A total of 90 oversampling techniques in the form of SMOTE and its variants were used. The results of this study indicate that there is no oversampling technique that is able to overcome this. The original dataset without oversampling shows good performance at the level of accuracy and f1-score but has low performance on auc-score and g-score. Several oversampling techniques show increased performance on auc-score and g-score, unfortunately at the same time showing a decrease in performance on accuracy and f1-score.

I. PENDAHULUAN

Software Defect Prediction (SDP) model has been widely researched to streamline software testing costs which ultimately reduces software development costs. In general, SDP models use various code and development metrics as features to classify target code fragments as bugs or not[1].

Software Defect Prediction model uses datasets that have been collected by various parties, one of which is NASA[2]. The problem that often occurs in this dataset is class imbalance. This can be seen in table I. This means that the data set contains more non-defective examples than defective examples. This creates problems for data mining algorithms because there is underrepresentation of defective examples and overrepresentation of non-defective examples. Resampling techniques can be used by data miners to account for class imbalance. The resampling technique involves changing the examples in the data set. This can be achieved using under-sampling or over-sampling techniques [3].

Technique most widely used to overcome this on the Software Defect Prediction dataset is SMOTE[3][4][5][6]. SMOTE[4] is called a synthetic oversampling technique because it creates new synthetic examples rather than duplicating existing ones. For each record in the minority class of a dataset, SMOTE finds its k nearest neighbors. One of the k nearest neighbors is chosen at random. For each attribute, the new synthetic record takes a value between the attribute value of the current instance and N. SMOTE uses user-defined parameters that decide how many new records to create. A value of 300 will create 300% more examples[7].

Methodological reviews and comparisons in imbalance data mining have been widely carried out[8][9][10][11][12]. Even research on various variations of SMOTE has been carried out [13]. However, this paper focuses on using oversampling techniques on the NASA Software Defect Prediction dataset which has not been carried out by previous research.

In research conducted by Kovacs[13] using 85 variants of the oversampling technique using 104 datasets without including a single Defect Prediction Dataset Software, therefore this research uses 12 datasets developed by NASA [2].

This research aims to obtain the most appropriate oversampling technique so that it can help improve the quality of defect prediction in software.

TABLE I
IMBALANCE DATASET SOFTWARE DEFECT PREDICTION

Dataset	N	N-	N+	IR	d
MC2	125	81	44	1.840909	39
PC5	1711	1240	471	2.632696	38
KC1	1183	869	314	2.767516	21
JM1	7782	6110	1672	3.654306	21
KC3	194	158	36	4.388889	39
PC4	1287	1110	177	6.271186	37
CM1	327	285	42	6.785714	37
PC3	1077	943	134	7.037313	37
MW1	253	226	27	8.37037	37
PC1	705	644	61	10.55738	37
MC1	1988	1942	46	42.21739	21
PC2	745	729	16	45.5625	36

II. METHODS

Research methodology that will be used in this research is testing 90 variants of oversampling techniques on 12 NASA datasets for defect prediction software. Testing was carried out by dividing the dataset into two parts, namely training data and testing data. Then the training data was carried out using an oversampling technique to be implemented in 5 classification models, namely Decision Tree (DT), k-nearest neighbors (KNN), Linear Discriminant Analysis (LDA), Logistic Regression (LR), Gaussian Naive Bayes (GNB). To further optimize the model, hyperparameters are used in each model. Next, the model was tested with testing data and performance data was collected in the form of accuracy, f1-score, auc-score and g-score.

The research steps are as follows: First, twelve NASA defect prediction software datasets from the "D" collection are selected, including CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. Next, dimensionality reduction is applied to decrease the time and computational resources needed, as an excess of features does not necessarily enhance a machine learning model's performance and can, in fact, increase training time and resource demands. Principal Component Analysis (PCA), an unsupervised dimensionality reduction method, is used in this step to transform high-dimensional features into a smaller set of meaningful, uncorrelated principal components, retaining essential information from the original features. This reduction is achieved by transforming features into new variables, or principal components, that capture as much variance as possible from the original data without being correlated with each other. PCA determines these principal components by calculating the covariance matrix of the original data, along with the eigenvectors and eigenvalues that indicate each component's explained variance. The dataset is then projected into a lower-dimensional space by selecting the top n eigenvectors, with n representing the desired number of principal components.

In this research, the value of n is determined using random search from $\{n \in \mathbb{N} | 5 \leq n \leq N\}$ so that the best model performance is achieved, where N is the number of features. Before being entered into the classifier, the selected components are scaled into the interval [0,1] to avoid the dominance of certain components using the equation.

$$x_s = \frac{\mathrm{x} - \min(\mathrm{x})}{\max(x) - \min(x)}$$

Where xs is the scaled component, x is the original component, and max(x) and min(x) are the maximum and minimum of the original component.

The next steps in the research process involve splitting the dataset into two parts, with 85% allocated for training data and 15% for testing data. An oversampling technique is then applied, using 90 distinct oversampling techniques to balance the dataset. After this, a classification model is built using the hyperparameters specified in Table II. The model is subsequently tested using the testing data, and performance metrics are retrieved from the test results to evaluate the model's effectiveness.

a) Accuracy

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

b) F1-score

$$f1 = \frac{PR.RE}{PR + RE}$$

$$PR = \frac{TP}{TP + FP}$$

$$RE = \frac{TP}{TP + FN}$$

c) AUC-score (FPR)

$$FPR = \frac{FP}{FP + TN}$$

d) G-score

$$g = \sqrt{\frac{TP}{P} \cdot \frac{TN}{N}}$$

Next, the same evaluation was carried out, but the dataset was divided into two types, namely with data amounts of less than 1,000 and data amounts of more than 1,000 to see whether there was an influence of the amount of data on the overall comparison that had been carried out previously.

III. RESULT AND DISCUSSION

The research results that we will present below are divided into 3 parts, namely the results for the entire dataset, the results for the dataset with more than 1.000 data and the results for the dataset with less than 1.000 data. The results displayed will only show the top 10 highest of each data performance, then the baseline results will also be presented,

namely the dataset performance without using the oversampling technique.

TABLE II.
TOP 10 ACCURACY RANKING FOR OVERALL DATASET

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	SMOTE_ENN	0.7969	0.8396	0.8455	0.8401	0.847	0.8338
2	SPY	0.7998	0.8257	0.8546	0.8453	0.8311	0.8313
3	SMOTE_RSB	0.7944	0.8337	0.8458	0.8401	0.837	0.8302
4	kmeans_SMOTE	0.8023	0.8446	0.8268	0.8312	0.8355	0.8281
5	NRAS	0.8266	0.8489	0.8242	0.8047	0.827	0.8263
6	NEATER	0.7898	0.7995	0.8409	0.8392	0.8289	0.8197
7	AHC	0.7648	0.7996	0.8431	0.8538	0.8262	0.8175
8	SOI_CJ	0.7977	0.8402	0.8271	0.8208	0.801	0.8173
9	DSMOTE	0.8007	0.837	0.8107	0.802	0.7812	0.8063
10	MDO	0.7751	0.8221	0.7764	0.7999	0.7936	0.7934
Baseline (Origin)		0.7985	0.8358	0.8455	0.8401	0.8383	0.8316

TABLE III.
TOP 10 ACCURACY RANKING FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	SMOTE_ENN	0.7642	0.8342	0.8336	0.832	0.8334	0.8195
2	SMOTE_RSB	0.7635	0.8342	0.8342	0.832	0.8326	0.8193
3	SPY	0.7698	0.8258	0.8299	0.8332	0.8334	0.8184
4	kmeans_SMOTE	0.7639	0.8348	0.8212	0.8134	0.8316	0.813
5	NRAS	0.8259	0.8283	0.7959	0.7895	0.8037	0.8087
6	NEATER	0.7637	0.7786	0.8069	0.812	0.8135	0.7949
7	AHC	0.7384	0.7796	0.8217	0.8184	0.8098	0.7936
8	SOI_CJ	0.7608	0.818	0.8011	0.7941	0.7661	0.788
9	AMSCO	0.7309	0.7696	0.7925	0.7879	0.8003	0.7762
10	LVQ_SMOTE	0.7417	0.7744	0.7682	0.7644	0.81	0.7717
Baseline (Origin)		0.7609	0.837	0.8336	0.832	0.8334	0.8194

In general, in the results shown in tables II, III and IV, there is no significant increase in accuracy between the results with and without oversampling. Even only the SMOTE ENN oversampling technique has better results than without oversampling. Datasets less than 1.000 show a higher level of accuracy on all classification models than datasets more than 1.000. Meanwhile, the F1-score results shown in tables V, VI and VII, show no improvement between the results with and without oversampling. The average of the results without oversampling is still in the top 10 range. Datasets less than 1,000 show a much better F1-score than datasets more than 1,000 in all classification models.

TABLE IV.
TOP 10 ACCURACY RANKING FOR DATASETS WITH LESS THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	SMOTE_ENN	0.8296	0.8449	0.8575	0.8482	0.8606	0.8482
2	SOI_CJ	0.8345	0.8624	0.8531	0.8475	0.8359	0.8467
3	DSMOTE	0.8356	0.861	0.8553	0.8283	0.8434	0.8447
4	NEATER	0.816	0.8205	0.8749	0.8663	0.8443	0.8444
5	SPY	0.8299	0.8256	0.8794	0.8574	0.8288	0.8442
6	NRAS	0.8273	0.8696	0.8524	0.82	0.8502	0.8439
7	kmeans_SMOTE	0.8408	0.8543	0.8324	0.849	0.8395	0.8432
8	AHC	0.7912	0.8196	0.8646	0.8892	0.8426	0.8414
9	SMOTE_RSB	0.8252	0.8331	0.8575	0.8483	0.8415	0.8411
10	Stefanowski	0.8334	0.7424	0.827	0.8414	0.8411	0.8171
Baseline (Origin)		0.8361	0.8345	0.8575	0.8482	0.8431	0.8439

TABLE V.
TOP 10 F1-SCORE RANKING FOR OVERALL DATASET

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	AHC	0.8043	0.7903	0.8184	0.8317	0.8043	0.8098
2	SPY	0.804	0.8002	0.8202	0.8149	0.804	0.8087
3	kmeans_SMOTE	0.8075	0.8124	0.8039	0.811	0.8075	0.8085
4	NEATER	0.7997	0.7976	0.8188	0.8236	0.7997	0.8079
5	NRAS	0.8048	0.8151	0.8119	0.7955	0.8048	0.8064
6	SMOTE_ENN	0.8144	0.8018	0.8038	0.7858	0.8144	0.804
7	MDO	0.791	0.815	0.7926	0.8103	0.791	0.8
8	SOI_CJ	0.7946	0.8038	0.8018	0.7949	0.7946	0.798
9	SMOTE_RSB	0.8004	0.7953	0.8042	0.7868	0.8004	0.7974
10	Stefanowski	0.8119	0.735	0.8017	0.8073	0.8119	0.7936
Baseline (Origin)		0.803	0.7971	0.8038	0.7858	0.803	0.7985

TABLE VI.
TOP 10 F1-SCORE RANKING FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	SPY	0.772	0.8017	0.7883	0.7958	0.8009	0.7918
2	NRAS	0.7964	0.7949	0.7881	0.7862	0.7843	0.79
3	NEATER	0.7655	0.7793	0.7838	0.7903	0.7908	0.7819
4	kmeans_SMOTE	0.7636	0.7942	0.7802	0.7799	0.7883	0.7813
5	SMOTE_ENN	0.76	0.7888	0.7835	0.7761	0.7916	0.78
6	SMOTE_RSB	0.7591	0.7888	0.7842	0.7761	0.7893	0.7795
7	AHC	0.7477	0.7714	0.7901	0.7909	0.7858	0.7772
8	MDO	0.7416	0.8083	0.7624	0.7842	0.7762	0.7745
9	SOI_CJ	0.7592	0.7802	0.7787	0.7804	0.7737	0.7745
10	SSO	0.7365	0.7967	0.7793	0.7733	0.7813	0.7734
Baseline (Origin)		0.7571	0.7939	0.7835	0.7761	0.7916	0.7804

TABLE VII.
TOP 10 F1-SCORE RANKING FOR DATASETS WITH LESS THAN 1,000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	kmeans_SMOTE	0.835	0.8307	0.8276	0.8421	0.8267	0.8324
2	AHC	0.7994	0.8091	0.8468	0.8724	0.8228	0.8301
3	NEATER	0.8011	0.8159	0.8537	0.8569	0.8086	0.8272
4	SPY	0.8291	0.7986	0.852	0.8341	0.807	0.8242
5	NRAS	0.8191	0.8353	0.8356	0.8047	0.8252	0.824
6	DSMOTE	0.8225	0.842	0.8327	0.7972	0.8199	0.8229
7	MDO	0.8231	0.8218	0.8227	0.8365	0.8059	0.822
8	SOI_CJ	0.829	0.8273	0.8249	0.8094	0.8155	0.8213
9	SMOTE_ENN	0.8215	0.8147	0.8242	0.7955	0.8372	0.8186
10	Stefanowski	0.8323	0.7515	0.8192	0.8452	0.8335	0.8163
Baseline (Origin)		0.828	0.8002	0.8242	0.7955	0.8144	0.8125

TABLE VIII.
TOP 10 AUC-SCORE RANKING FOR OVERALL DATASET

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.6604	0.6807	0.6558	0.6676	0.6335	0.6596
2	PDFOS	0.6376	0.6581	0.6558	0.6605	0.6154	0.6455
3	polynom_fit_SMOTE_mesh	0.6382	0.6543	0.6613	0.6627	0.6028	0.6438
4	ROSE	0.6016	0.6735	0.6449	0.6594	0.617	0.6393
5	polynom_fit_SMOTE_bus	0.6185	0.6442	0.6586	0.6678	0.5993	0.6377
6	SMOTE_Cosine	0.6216	0.6428	0.6596	0.6619	0.599	0.637
7	SVM_balance	0.6259	0.6178	0.6302	0.6724	0.6338	0.636
8	SMOTE_AMSR	0.6139	0.6587	0.6482	0.6401	0.6159	0.6354
9	polynom_fit_SMOTE_star	0.5747	0.648	0.6589	0.666	0.6226	0.6341
10	Assembled_SMOTE	0.608	0.6432	0.6507	0.6606	0.6061	0.6337
Baseline (Origin)		0.5559	0.5287	0.5386	0.529	0.5477	0.54

TABLE IX.
TOP 10 AUC-SCORE RANKING FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.6069	0.691	0.6342	0.6509	0.6345	0.6435
2	SVM_balance	0.6247	0.6337	0.6296	0.6444	0.6198	0.6304
3	ROSE	0.5943	0.6561	0.6426	0.6483	0.6054	0.6293
4	KernelADASYN	0.6031	0.6394	0.6439	0.6458	0.6049	0.6274
5	SMOTE_Cosine	0.5743	0.6676	0.6399	0.6423	0.5963	0.6241
6	polynom_fit_SMOTE_bus	0.5699	0.6512	0.6417	0.651	0.5989	0.6225
7	polynom_fit_SMOTE_mesh	0.5691	0.6505	0.6404	0.6444	0.6072	0.6223
8	Gaussian_SMOTE	0.5689	0.6665	0.6375	0.6472	0.5768	0.6194
9	PDFOS	0.5917	0.6274	0.6351	0.649	0.5908	0.6188
10	ANS	0.5967	0.5996	0.6332	0.6496	0.6137	0.6185
Baseline (Origin)		0.5355	0.5456	0.5341	0.5246	0.5539	0.5387

TABLE X.
TOP 10 AUC-SCORE RANKING FOR DATASETS WITH LESS THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.7139	0.6704	0.6774	0.6842	0.6324	0.6757
2	PDFOS	0.6836	0.6888	0.6765	0.672	0.6399	0.6722
3	polynom_fit_SMOTE_mesh	0.7073	0.6581	0.6821	0.6809	0.5983	0.6654
4	SMOTE_AMSR	0.6475	0.7008	0.6427	0.6454	0.6408	0.6554
5	polynom_fit_SMOTE_star	0.6044	0.6695	0.6766	0.6873	0.6314	0.6538
6	polynom_fit_SMOTE_bus	0.6672	0.6372	0.6755	0.6846	0.5998	0.6529
7	SMOTE_Cosine	0.6688	0.618	0.6793	0.6816	0.6018	0.6499
8	Assembled_SMOTE	0.6392	0.6708	0.6627	0.669	0.6067	0.6497
9	ROSE	0.6089	0.6909	0.6471	0.6705	0.6286	0.6492
10	OUPS	0.6123	0.6857	0.6764	0.6678	0.6006	0.6486
Baseline (Origin)		0.5763	0.5118	0.543	0.5334	0.5415	0.5412

TABLE XI.
TOP 10 G-SCORE RANKING FOR OVERALL DATASET

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.6443	0.6632	0.6362	0.6549	0.6135	0.6424
2	polynom_fit_SMOTE_mesh	0.6318	0.6425	0.6463	0.6516	0.5511	0.6247
3	SMOTE_AMSR	0.6097	0.6436	0.6402	0.6313	0.5802	0.621
4	polynom_fit_SMOTE_bus	0.6035	0.6375	0.6507	0.6591	0.5376	0.6177
5	ROSE	0.5946	0.6631	0.629	0.6495	0.5464	0.6165
6	SMOTE_Cosine	0.6141	0.6297	0.6453	0.6507	0.5389	0.6157
7	PDFOS	0.6308	0.618	0.6409	0.6502	0.5318	0.6143
8	SVM_balance	0.6155	0.5989	0.5945	0.6612	0.5983	0.6137
9	Assembled_SMOTE	0.5824	0.6232	0.6379	0.6513	0.5489	0.6087
10	polynom_fit_SMOTE_star	0.5106	0.6194	0.6494	0.6557	0.5851	0.604
Baseline (Origin)		0.3696	0.2019	0.2437	0.1787	0.2948	0.2577

TABLE XII.
TOP 10 G-SCORE RANKING FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.5845	0.6734	0.6035	0.6376	0.6103	0.6219
2	SVM_balance	0.6091	0.6185	0.5802	0.6373	0.5692	0.6029
3	KernelADASYN	0.5997	0.6181	0.629	0.6371	0.5228	0.6013
4	ROSE	0.5893	0.6494	0.6202	0.6376	0.5102	0.6013
5	SMOTE_AMSR	0.5768	0.6007	0.6414	0.6267	0.5534	0.5998
6	polynom_fit_SMOTE_bus	0.558	0.645	0.6335	0.644	0.5172	0.5995
7	SMOTE_Cosine	0.5686	0.6616	0.618	0.6301	0.512	0.598
8	polynom_fit_SMOTE_mesh	0.5636	0.6385	0.6169	0.6327	0.5311	0.5966
9	ANS	0.5771	0.5698	0.6072	0.639	0.5686	0.5924
10	SMOTE_FRST_2T	0.5357	0.6079	0.645	0.6316	0.5309	0.5902
Baseline (Origin)		0.3757	0.2591	0.2546	0.1967	0.3427	0.2858

TABLE XIII.
TOP 10 G-SCORE RANKING FOR FOR DATASETS WITH LESS THAN 1.000

R	Oversampling	DT	KNN	LDA	LR	GNB	Mean
1	Gazzah	0.7041	0.6529	0.6689	0.6722	0.6167	0.663
2	polynom_fit_SMOTE_mesh	0.7	0.6465	0.6757	0.6705	0.5711	0.6527
3	PDFOS	0.6753	0.6504	0.6697	0.6633	0.5811	0.6479
4	SMOTE_AMSR	0.6425	0.6865	0.6391	0.6358	0.607	0.6422
5	polynom_fit_SMOTE_bus	0.6489	0.6301	0.6679	0.6743	0.558	0.6358
6	SMOTE_Cosine	0.6596	0.5978	0.6726	0.6713	0.5658	0.6334
7	ROSE	0.5999	0.6768	0.6378	0.6614	0.5826	0.6317
8	OUPS	0.5885	0.6781	0.6709	0.6593	0.5498	0.6293
9	Assembled_SMOTE	0.6141	0.6407	0.6572	0.6601	0.5646	0.6273
10	SVM_balance	0.6219	0.5793	0.6088	0.6851	0.6273	0.6245
Baseline (Origin)		0.3635	0.1446	0.2327	0.1607	0.2468	0.2297

TABLE XIV.
TOP 10 OVERALL RANKING FOR OVERALL DATASET

R	Oversampling	Acc Rank	f1 Rank	auc Rank	g Rank	Average Rank
1	PDFOS	33	23	2	7	16.25
2	Gazzah	40	30	1	1	18
3	ROSE	61	31	4	5	25.25
4	Supervised_SMOTE	16	14	23	54	26.75
5	MDO	11	7	29	66	28.25
6	V_SYNTH	38	32	16	33	29.75
7	Gaussian_SMOTE	49	28	22	21	30
8	polynom_fit_SMOTE_star	57	46	9	10	30.5
9	SSO	29	29	28	38	31
10	Assembled_SMOTE	59	47	10	9	31.25
60	Baseline(Origin)	2	8	89	90	47.25

TABLE XV.
TOP 10 OVERALL RANKING FOR FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	Acc Rank	f1 Rank	auc Rank	g Rank	Average Rank
1	Gazzah	40	29	1	1	17.75
2	Gaussian_SMOTE	34	31	8	19	23
3	SSO	15	11	22	52	25
4	SVM_balance	53	50	2	2	26.75
5	SL_graph_SMOTE	37	36	15	23	27.75
6	ProWSyn	36	34	13	33	29
7	PDFOS	48	45	9	17	29.75
8	ROSE	60	54	3	4	30.25
9	polynom_fit_SMOTE_pol_y	49	49	11	15	31
10	Borderline_SMOTE1	44	46	16	20	31.5
55	Baseline(Origin)	2	5	87	89	45.75

TABLE XVI.
TOP 10 OVERALL RANKING FOR FOR DATASETS WITH MORE THAN 1.000

R	Oversampling	Acc Rank	f1 Rank	auc Rank	g Rank	Average Rank
1	PDFOS	30	27	2	3	15.5
2	Gazzah	39	33	1	1	18.5
3	cluster_SMOTE	17	18	13	38	21.5
4	Supervised_SMOTE	15	15	18	49	24.25
5	V_SYNTH	31	30	11	25	24.25
6	MDO	13	7	17	61	24.5
7	polynom_fit_SMOTE_star	54	49	5	11	29.75
8	GASMOTE	19	24	27	55	31.25
9	polynom_fit_SMOTE_bus	57	57	6	5	31.25
10	Lee	14	12	33	67	31.5
63	Baseline(Origin)	7	13	89	90	49.75

The AUC-score results shown in tables VIII, IX and X, show a significant increase between the results with and without oversampling. The AUC-score on the qualification model without oversampling shows a very low number. Datasets over 1,000 show a lower AUC-score than datasets over 1,000 in all classification models.

Meanwhile, the g-score results shown in tables XI, XII and XIII show a significant increase between the results with and without oversampling. The g-score on the qualification model without oversampling is at a very low value and far from the average value. These results show that even without oversampling it shows very good accuracy, but only on the majority of data. Datasets over 1,000 show almost the same g-score compared to datasets over 1,000 in all classification models.

V. CONCLUSION

From the research findings, it is evident that none of the oversampling techniques achieve the highest ranking across all metrics—accuracy, F1-score, AUC-score, and G-score—simultaneously. Techniques that rank high in accuracy and F1-score tend to perform lower in AUC-score and G-score, while those with high AUC-score and G-score rankings exhibit moderate rankings in accuracy and F1-score. The original dataset, without any oversampling applied, similarly shows high accuracy and F1-score, but performs poorly in terms of AUC-score and G-score. This pattern suggests that oversampling techniques with the highest accuracy rankings are often influenced by a high True Negative count, coupled with a low True Positive count and a high False Negative rate. Consequently, the techniques with higher overall rankings are typically those that score well in AUC-score and G-score.

The Gazzah oversampling technique demonstrates strong performance across datasets larger and smaller than 1,000 instances, outperforming the PDFOS technique in smaller datasets even though PDFOS ranks higher overall. This indicates that while PDFOS is effective, its performance on

larger datasets is less consistent. Further research is essential to develop oversampling techniques capable of improving all key metrics—accuracy, F1-score, AUC-score, and G-score—simultaneously, optimizing the balance between these performance measures.

ACKNOWLEDGEMENT

We would like to thank Nusa Mandiri University for providing tremendous support in our research.

DAFTAR PUSTAKA

[1] C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. Whitehead Jr., "Does Bug Prediction Support Human Developers?Findings from a Google Case Study," 2013.

[2] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," IEEE Transactions on Software Engineering, vol. 39, no. 9, pp. 1208– 1215, 2013, doi: 10.1109/TSE.2013.11.

[3] M. J. Siers and M. Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," Inf Syst, vol. 51, pp. 62–71, 2015, doi: 10.1016/j.is.2015.02.006.

[4] S. Choirunnisa, B. Meidyani, and S. Rochimah, "Software Defect Prediction using Oversampling Algorithm: A-SUWO," 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar, EECCIS 2018, pp. 337–341, 2018, doi: 10.1109/EECCIS.2018.8692874.

[5] H. Ghinaya, R. Herteno, M. R. Faisal, A. Farmadi, and F. Indriani, "Analysis of Important Features in Software Defect Prediction using Synthetic Minority Oversampling Techniques (SMOTE), Recursive Feature Elimination (RFE) and Random Forest," Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 2, pp. 276–288, 2024.

[6] S. Feng et al., "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction," Inf Softw Technol, vol. 129, no. September, p. 106432, 2021, doi: 10.1016/j.infsof.2020.106432.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, no. February, pp. 321– 357, 2002, doi: 10.1613/jair.953.

[8] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," Data Mining and Knowledge Discovery Handbook, no. May, pp. 875–886, 2009, doi: 10.1007/978-0-387-09823- 4_45.

[9] V. López, A. Fernández, and F. Herrera, "On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed," Inf Sci (N Y), vol. 257, pp. 1–13, 2014, doi: 10.1016/j.ins.2013.09.038.

[10] T. Raeder, G. Forman, and N. V. Chawla, "Learning from Imbalanced Data: Evaluation Matters," Intelligent Systems Reference Library, vol. 23, pp. 315–331, 2012, doi: 10.1007/978- 3-642-23166-7_12.

[11] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," Inf Sci (N Y), vol. 250, pp. 113–141, 2013, doi: 10.1016/j.ins.2013.07.007.

[12] T. Ryan Hoens and N. V. Chawla, "Imbalanced datasets: From sampling to classifiers," Imbalanced Learning: Foundations, Algorithms, and Applications, pp. 43–59, 2013, doi: 10.1002/9781118646106.ch3.

[13] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," Applied Soft Computing Journal, vol. 83, no. July, 2019, doi: 10.1016/j.asoc.2019.105662.

[14] M. Z. F. N. Siswantoro and U. L. Yuhana, "Software Defect Prediction Based on Optimized Machine Learning Models: A Comparative Study," Teknika, vol. 12, no. 2, pp. 166–172, 2023, doi: 10.34148/teknika.v12i2.634.

[15] I. T. Jolliffe, "Principal components," Data Handling in Science and Technology, vol. 20, no. PART A, pp. 519–556, 1998, doi: 10.1016/S0922-3487(97)80047-0.