JURNAL INFORMATIKA, Vol.6 No.1 April 2019, pp. 107~113
ISSN: 2355-6579
E-ISSN**:** 2528-2247

Mengatasi *Imbalanced Class* Pada *Software Defect Prediction* Menggunakan *Two-Step Clustering-Based Undersampling* dan *Bagging Tehcnique*

Muhammad Faittullah Akbar[1], Ilham Kurniawan[2], Ahmad Fauzi[3]

[1]Universitas Bina Sarana Informatika
e-mail: muhammad.mtl@bsi.ac.id

[2] Universitas Bina Sarana Informatika
e-mail: ilham.imk@bsi.ac.id

[3] Universitas Bina Sarana Informatika
e-mail: ahmad.fzx@bsi.ac.id

Abstrak

Ketidakseimbangan kelas seringkali menjadi masalah di berbagai set data dunia nyata, di mana satu kelas (yaitu kelas minoritas) berisi sejumlah kecil titik data dan yang lainnya (yaitu kelas mayoritas) berisi sejumlah besar titik data. Sangat sulit untuk mengembangkan model yang efektif dengan menggunakan data mining dan algoritma machine learning tanpa mempertimbangkan preprocessing data untuk menyeimbangkan set data yang tidak seimbang. Random undersampling dan oversampling telah digunakan dalam banyak penelitian untuk memastikan bahwa kelas yang berbeda mengandung jumlah titik data yang sama. Dalam penelitian ini, kami mengusulkan kombinasi two-step clustering-based random undersampling dan bagging technique untuk meningkatkan nilai akurasi software defect prediction. Metode yang diusulkan dievaluasi menggunakan lima set data dari repositori program data metrik NASA dan area under the curve (AUC) sebagai evaluasi utama. Hasil telah menunjukkan bahwa metode yang diusulkan menghasilkan kinerja yang sangat baik untuk semua dataset (AUC> 0,9). Dalam hal SN, percobaan kedua mengungguli percobaan pertama di hampir semua dataset (3 dari 5 dataset). Sementara itu, dalam hal SP, percobaan pertama tidak mengungguli percobaan kedua di semua dataset. Secara keseluruhan percobaan kedua mengungguli dan lebih baik daripada percobaan pertama karena evaluasi utama dalam klasifikasi kelas yang tidak seimbang seperti SDP adalah AUC Oleh karena itu, dapat disimpulkan bahwa metode yang diusulkan menghasilkan kinerja yang optimal baik untuk set data skala kecil maupun besar.
Keywords: Ketidakseimbangan kelas, Machine Learning, Clustering, Ensemble Classifier.

Abstract

Iimbalances class are often a problem in a variety of real-world data sets, where one class (i.e. minority class) contains a small number of data points and the other (i.e. majority class) contains a large number of data points. It is very difficult to develop effective models using data mining and machine learning algorithms without considering preprocessing data to balance unbalanced data sets. Random undersampling and oversampling have been used in many studies to ensure that different classes contain the same number of data points. In this study, we propose a combination of two-step clustering-based random undersampling and bagging technique to increase the accuracy of software defect prediction. The proposed method was evaluated using five data sets from the NASA metric data program repository and area under the curve (AUC) as the main evaluation. Results have shown that the proposed method produces very good performance for all datasets (AUC> 0.9). In the case of SN, the second experiment outperformed the first experiment in almost all datasets (3 of 5 datasets). Meanwhile, in the case of SP, the first trial did not outperform the second trial in all datasets. Overall the second experiment outperformed and was better than the first experiment because the main evaluation in class classification that was imbalanced class like SDP was AUC. Therefore, it could be concluded that the proposed method produced optimal performance for both small and large scale data sets.
Keywords*: Iimbalance Class, Machine Learning, Clustering, Ensemble Classifier.*

1. Pendahuluan

Software defect prediction (SDP) adalah proses memprediksi bagian mana dari kode yang rusak dan mana yang tidak (Siers & Islam, 2015). Prediksi akurat dari modul perangkat lunak rawan cacat dapat membantu upaya pengujian secara langsung, mengurangi biaya dan juga meningkatkan proses pengujian perangkat lunak dengan berfokus pada modul rawan kesalahan (Catal, 2011). Dalam *data mining* dan *machine learning*, sulit untuk melatih model pembelajaran yang efektif jika distribusi kelas dalam set data pelatihan yang diberikan tidak seimbang. Ini dikenal sebagai masalah ketidakseimbangan kelas. Satu kelas mungkin diwakili oleh sejumlah besar contoh, sedangkan contoh yang lain mungkin diwakili oleh hanya beberapa. Selain itu, untuk sebagian besar algoritma *data mining* (Lin, Tsai, Hu, & Jhang, 2017).

Tanpa mempertimbangkan masalah ketidakseimbangan kelas, algoritma pembelajaran atau model yang dibangun dapat dikuasai oleh kelas mayoritas dan dapat mengabaikan kelas minoritas. Sebagai contoh, pertimbangkan kumpulan data dua kelas dengan rasio ketidakseimbangan 99%, di mana kelas mayoritas merupakan 99% dari kumpulan data dan kelas minoritas hanya mengandung 1%. Untuk meminimalkan tingkat kesalahan, algoritma pembelajaran mengklasifikasikan semua contoh ke dalam kelas mayoritas, yang menghasilkan tingkat kesalahan 1%. Dalam hal ini, semua contoh milik kelas minoritas adalah yang terpenting dan harus diidentifikasi sebagai klasifikasi yang salah (Learning, Liu, Wu, Zhou, & Member, 2009).

Berbagai metode telah diusulkan untuk menyelesaikan masalah ini. Metode tersebut dapat dibagi menjadi empat jenis: metode tingkat algoritmik, metode tingkat data, metode *cost sensitive*, dan *ensemble clasification* (Galar, Fern, Barrenechea, & Bustince, 2012). Secara khusus, metode tingkat data, yang fokus pada *preprocessing* set data yang tidak seimbang sebelum membangun pengklasifikasi, secara luas dipertimbangkan dalam literatur. Ini karena tugas-tugas *preprocessing* data dan pelatihan *classifier* dapat dilakukan secara independen. Selain itu, menurut Galar et al. (Galar et al., 2012), yang melakukan studi perbandingan berbagai pendekatan terkenal, kombinasi metode *preprocessing* data dengan *ensemble classifier* berkinerja lebih baik daripada metode lain.

Metode data *preprocessing* didasarkan pada *resampling* set data pelatihan yang tidak seimbang sebelum tahap pelatihan model. Untuk menciptakan keseimbangan, set data ketidakseimbangan yang asli dapat disampel ulang dengan melakukan *oversampling* kelas minoritas (Hu, 2009) dan / atau *undersampling* kelas mayoritas (Li, Liu, & Hu, 2010). Beberapa pendekatan representatif menggabungkan *preprocessing* data *oversampling* dan *undersampling* dengan *ensemble classifier* melalui teknik *boosting* (Schapire, 1990) atau *bagging* (Bbeiman, 1996); misalnya *SMOTEBoost* (Chawla, Lazarevic, Hall, & Bowyer, n.d.), *RUS Boost* (Seiffert, Khoshgoftaar, Hulse, & Napolitano, 2010), *OverBagging* (Solis, Avizzano, & Bergamasco, 2002), dan *UnderBagging* (Barandela, Sánchez, & Valdovinos, 2003).

Sebagian besar pendekatan ini melakukan beberapa putaran *resampling* acak untuk kelas mayoritas (misalkan *undersampling*) atau minoritas (misalkan *oversampling*). Dalam kelompok metode berikutnya, set pelatihan seimbang yang berbeda digunakan untuk melatih sejumlah pengklasifikasi spesifik untuk kombinasi selanjutnya sebagai *ensemble classifier*. Dari dua strategi *resampling* ini, *undersampling* telah terbukti menjadi pilihan yang lebih baik daripada *oversampling* (Galar et al., 2012). Ini karena strategi *oversampling* dapat meningkatkan kemungkinan *overfitting* dalam proses konstruksi model. Namun, dengan strategi *undersampling*, beberapa data berguna yang hadir di kelas mayoritas mungkin dihilangkan (Sun et al., 2015).

Untuk mengatasi keterbatasan *undersampling*, kami mengusulkan untuk mengganti strategi *random undersampling* dengan teknik *clustering*. Tujuan dari analisis *clustering* adalah untuk mengelompokkan objek yang serupa (yaitu sampel data) ke dalam kelompok yang sama; objek dalam kelompok berbeda, berbeda dalam hal representasi fitur mereka (Jain, A, Murty, M, & Flynn, P, 1999). Oleh karena itu, menggunakan analisis pengelompokan untuk menggaris bawahi kelas mayoritas menghasilkan sejumlah *cluster*, dengan masing-masing

cluster berisi data yang sama. Secara khusus, setiap *cluster centroid* (atau pusat), yang didasarkan pada rata-rata data yang sama dalam kelompok yang sama yang dihitung oleh algoritma *k-means* (Hartigan & Wong, 1979), dapat digunakan untuk mewakili data dalam seluruh kelompok. Dengan kata lain, data asli dalam kelompok yang sama diganti oleh pusat-pusat *cluster*, sehingga mengurangi ukuran kelas mayoritas.

Dalam penelitian ini, kami menunjukkan bahwa jenis strategi *undersampling* berbasis *clustering,* ini dapat mengurangi risiko menghapus data yang berguna dari kelas mayoritas, memungkinkan pengklasifikasi yang dibangun (termasuk kedua pengklasifikasi tunggal dan *ensemble classifier*) untuk mengungguli pengklasifikasi yang dikembangkan menggunakan strategi *random undersampling*. Kami mengusulkan kombinasi *two-step cluster* (TSC) *clustering-based undersampling* (RUS) dan *bagging technique* (TSC-RUS + B) untuk meningkatkan akurasi SDP. TSC-RUS diterapkan untuk menangani kelas yang tidak seimbang dan teknik *bagging* digunakan untuk meningkatkan kinerja *classifier* di SDP. RUS dipilih karena banyak penelitian sebelumnya menggunakan pendekatan ini ketika berhadapan dengan klasifikasi kelas yang tidak seimbang. Sementara TSC dipilih sebagai algoritma *cluster* karena TSC dapat menyelesaikan setidaknya beberapa masalah ini, misalnya: kemampuan untuk menangani variabel tipe campuran dan kumpulan data besar, penentuan otomatis jumlah *cluster optimal*, dan variabel yang mungkin tidak normal (Michailidou, Maheras, Arseni-Papadimititriou, Kolyva-Machera, & Anagnostopoulou, 2009).

2. Metode Penelitian

Kami mengusulkan metode yang disebut TSC-RUS + B, *two-step clustering based undersampling* dan teknik *bagging* untuk mengatasi masalah kelas yang tidak seimbang dalam prediksi cacat perangkat lunak. TSC adalah salah satu algoritma pengelompokan yang dikembangkan pertama kali oleh (Chiu, Fang, Chen, Wang, & Jeris, 2001) dan dirancang untuk menangani kumpulan data yang sangat besar. TSC mampu menangani variabel kontinu dan variabel kategorikal (Chiu et al., 2001), (Satish & Bharadhwaj, 2010). Dalam teknik *bagging*, Naive Bayes (NB), Logistic Regression (LR), dan k-Nearest Neighbor (kNN), digunakan sebagai pelajar dasar, Support Vector Machine (SVM) digunakan sebagai model *testing*. Gambar 1 menunjukkan diagram blok dari metode yang diusulkan.

Metode yang diusulkan dievaluasi menggunakan lima dataset program data metrik NASA (NASA MDP), yaitu: CM1, KC1, KC2, MC2, PC1. Seperti yang ditunjukkan pada Gambar 1, lima dataset NASA MDP dikalikan masing-masing dan dimasukkan ke fase pelatihan dan fase pengujian masing-masing; di mana fase pelatihan digunakan untuk membangun model dan fase pengujian digunakan untuk menguji model dan mengevaluasi kinerjanya.

Pada fase pelatihan, dataset kemudian dikelompokkan menggunakan algoritma TSC. Jumlah *cluster* diatur ke 2, *cluster* sebagai ide yang sama dengan membuat 2-*binning* atau 2 kuartil. Kemudian, kami melakukan *random undersampling* untuk setiap *cluster*, sehingga kelas mayoritas dan kelas minoritas adalah angka yang sama untuk setiap *cluster*. Setelah itu, dataset baru dibuat dengan menggabungkan dari semua cluster dengan proporsi yang sama untuk setiap kelas.

Setelah itu, dataset baru adalah teknik *bagging* dengan pendekatan *cross validation* 10 kali lipat dimana dataset akan dibagi menjadi 10 bagian dataset, 1 bagian sebagai dataset pengujian dan sisanya sebagai dataset pelatihan dan proses ini diulang 10 kali. Kami menggunakan NB, LR dan kNN sebagai pelajar dasar dan SVM sebagai model *testing* dalam teknik *bagging*. Setelah proses pembelajaran selesai, model akan memberi hasil dengan dataset pengujian dalam fase pengujian dan kemudian kami mencatat hasil evaluasi.

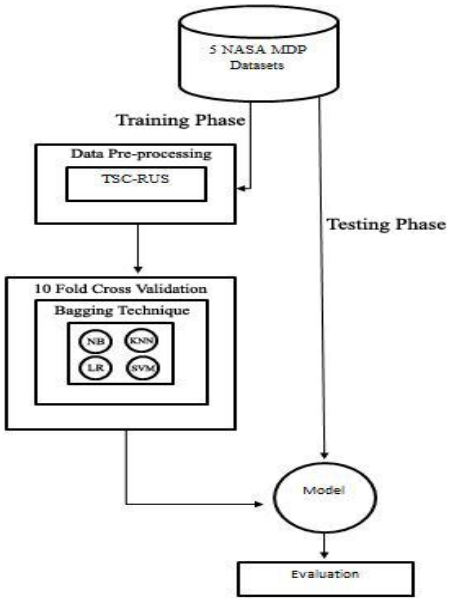
Dalam penelitian ini, metode yang diusulkan dievaluasi dengan menggunakan efektivitas *classifier* berdasarkan *confusion matrix* dengan evaluasi utama adalah *area under the curve* (AUC) seperti yang digunakan oleh (Laradji, Alshayeb, & Ghouti, 2015), (Rana, Mian, & Shamail, 2015), (Czibula, Marian, & Czibula, 2014), (Wahono & Herman, 2014), (Arar & Ayan, 2015) AUC memiliki potensi untuk secara signifikan meningkatkan konvergensi di seluruh eksperimen empiris dalam prediksi

cacat perangkat lunak (Laradji et al., 2015) dan penggunaan AUC untuk meningkatkan perbandingan studi silang (Lessmann, Baesens, Mues, & Pietsch, 2008). Panduan dasar untuk mengklasifikasikan keakuratan tes diagnostik berdasarkan AUC sebagaimana dinyatakan oleh (Gorunescu, 2011) sebagai berikut:

- 0,90 - 1,00 = klasifikasi sangat baik
- 0,80 - 0,90 = klasifikasi yang baik
- 0,70 - 0,80 = klasifikasi yang adil
- 0,60 - 0,70 = klasifikasi buruk
- 0,50 - 0,60 = kegagalan.

Evaluasi lain dari metode yang diusulkan yaitu: *recall* atau *sensitivity* (SN) seperti yang digunakan oleh (Laradji et al., 2015), (Czibula et al., 2014), (Wahono & Herman, 2014); *specipies* (SP) dan *precision* (PR) seperti yang digunakan oleh (Czibula et al., 2014), (Wahono & Herman, 2014).

Evaluasi ini didasarkan pada *confusion matrix* yang berisi nilai *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN) seperti yang ditunjukkan pada Tabel 1. TP berarti ketika label yang diprediksi rusak dan aktual label juga rusak. Ketika label yang diprediksi rusak tetapi label yang sebenarnya tidak rusak, itu disebut FP.



Gambar 1. *Block Diagram*

TN sama dengan TP tetapi dalam hal label tidak cacat, sedangkan FN adalah ketika label yang diprediksi tidak rusak tetapi sebenarnya labelnya rusak. Ini dihitung berdasarkan *confusion matrix* yang dihasilkan dari model. Berdasarkan *confusion matrix*, perhitungan pengukuran adalah sebagai berikut:

a) SN: mengukur proporsi instance pola positif yang secara benar diakui sebagai positif
SN = TP / (TP + FN) (1)
b) SP: mengukur proporsi instance pola negatif yang dikenali dengan benar sebagai negatif.
SP = TN / (TN + FP) (2)
c) PR: mengukur probabilitas bahwa instance pola prediksi positif dilabeli sebagai positif
PR = TP / (TP + FP) (3)

Tabel 1. Confusion Matrix

Predicted	Actual	
	Defective	Non-Defective
Defective	TP	FP
Non-Defective	FN	TN

3. Hasil dan Pembahasan

Eksperimen dilakukan dengan menggunakan *platform* komputasi berbasis AMD *Core* 2 Duo 1.9 GHz CPU, 2 GB RAM dan sistem operasi *Microsoft Windows* 10 64-bit, Weka versi 3.9 sebagai alat analisis data. Weka akan menghasilkan AUC dan *confusion matrix* sebagai hasil perhitungan.

1. Teknik *Bagging*

Pertama-tama, kami melakukan percobaan pada lima dataset MDP NASA dengan teknik *bagging* saja. *Confusion matrix* seperti yang ditunjukkan pada Tabel 2 dihasilkan untuk setiap dataset dari Weka dan kemudian kami mengevaluasi metode dengan menghitung SN, SP dan PR, sementara AUC langsung dihitung oleh Weka. Tabel 3 menunjukkan evaluasi metode lengkap.

Tabel 2. Confusion Matrix Tehnik Bagging

Dataset	TP	TN	FP	FN
CM1	0	42	0	285
KC1	1777	6	314	12
KC2	410	5	81	26
PC1	2	59	1	643
MC2	18	26	13	68

Tabel 3. Metode Evaluasi Untuk Tehnik *Bagging*

Dataset	AUC	SN	SP	PR
CM1	0,778[c]	0	0,812	0,872
KC1	0,762[c]	0,993	0,848	0,822
KC2	0,768[c]	0,940	0,797	0,835
PC1	0,847[b]	0,003	0,878	0,915
MC2	0,662[d]	0,209	0,673	0,688

[a] Sangat baik, [b] Baik, [c] Biasa, [d] Buruk, [e] Gagal.

Seperti yang ditunjukkan pada Tabel 3, metode ini menghasilkan 1 AUC dengan klasifikasi baik, 3 klasifikasi biasa dan 1 klasifikasi buruk. Sementara itu, SN bervariasi dari 0 - 0,993, SP bervariasi dari 0,673 - 0,878 dan PR bervariasi dari 0,688 - 0,915. Dalam hal AUC, metode ini sebagian besar menghasilkan klasifikasi yang biasa; sementara dalam hal SN dan PR, metode yang dihasilkan sebagian besar hasilnya rendah; sedangkan dalam hal SP, metode ini menghasilkan hasil yang baik. Namun, berdasarkan hasil, metode ini cukup menjanjikan karena masih menghasilkan 1 klasifikasi yang baik, hanya saja recall (SN) tidak lebih baik daripada *precision* (PR).

2. TSC-RUS +B Technique

Dalam percobaan kedua, kami menerapkan undersampling acak berdasarkan kluster dua langkah (TSC-RUS) dan dikombinasikan dengan teknik *bagging* (TSC-RUS + B). Hasil percobaan ditunjukkan pada Tabel 4 dan Tabel 5. *Confusion matrix* dihasilkan untuk lima set data dari Weka dan kemudian kami menghitung SN, SP dan PR, sedangkan AUC langsung dihitung oleh Weka.

Tabel 4. Confusion Matrix TSC-RUS + B

Dataset	TP	TN	FP	FN
CM1	110	1	1	215
KC1	367	4	11	1727
KC2	152	5	4	361
PC1	240	3	9	453
MC2	112	0	1	12

Tabel 5. Metode Evaluasi untuk TSC-RUS + B

Dataset	AUC	SN	SP	PR
CM1	0,996[a]	0,338	0.994	0.994
KC1	0.999[a]	0,175	0.993	0.993
KC2	0.995[a]	0.296	0.983	0.983
PC1	0.995[a]	0,346	0.983	0.983
MC2	1.000[a]	0.903	0.992	0.992

[a] Sangat baik, [b] Baik, [c] Biasa, [d] Buruk, [e] Gagal.

Seperti yang ditunjukkan pada Tabel 5, percobaan kedua menghasilkan hasil yang lebih baik di semua evaluasi daripada percobaan pertama. Dalam hal AUC, 5 diklasifikasikan sebagai sangat baik. SN bervariasi dari 0,175 – 0.903; SP bervariasi dari 0,983 - 0,994 dan PR bervariasi dari 0,992 - 0,994. Dalam soal SN dan PR, percobaan kedua menghasilkan lebih sedikit nilai lebih rendah dan lebih tinggi di nilai atas daripada percobaan pertama. Sedangkan dalam hal SP, percobaan kedua menghasilkan nilai lebih tinggi dan nilai lebih tinggi daripada percobaan pertama.

3. Perbandingan Hasil Eksperimen

Untuk perbandingan lebih rinci antara percobaan pertama dan kedua, kami menyajikan perbandingan pada Tabel 6. Font tebal menunjukkan nilai terbaik untuk setiap evaluasi. Seperti yang ditunjukkan pada Tabel 6, percobaan kedua (TSC-RUS + B) lebih baik dari semua dataset dalam hal AUC.

Tabel 6. Hasil Komparasi Tehnik Bagging vs TSC -RUS + B

Data set	**AUC**		**SN**		**SP**	
	(1)	**(2)**	**(1)**	**(2)**	**(1)**	**(2)**
CM1	0,778	**0,96**	0	**0,38**	0,812	**0.94**
KC1	0,762	**0.99**	**0,93**	0,175	0,848	**0.993**
KC2	0,768	**0.995**	**0,40**	0.296	0,797	**0.983**
PC1	0,847	**0.995**	0,003	**0,346**	0,878	**0.983**
MC2	0,662	**1.000**	0,209	**0.903**	0,673	**0.992**

(1) Teknik bagging, (2) TSC-RUS+ B

Dalam hal SN, percobaan kedua mengungguli percobaan pertama di hampir

semua dataset (3 dari 5 dataset). Sementara itu, dalam hal SP, percobaan pertama tidak mengungguli percobaan kedua di semua dataset. Secara keseluruhan percobaan kedua mengungguli dan lebih baik daripada percobaan pertama karena evaluasi utama dalam klasifikasi kelas yang tidak seimbang seperti SDP adalah AUC sebagaimana dinyatakan oleh (Rana et al., 2015), (Lessmann et al., 2008).

4. Kesimpulan

Metode *hybrid* baru yang mengintegrasikan *two-step clustering-based* dan *bagging technique* sebagai pendekatan algoritma diusulkan dalam makalah ini, untuk meningkatkan akurasi prediksi cacat perangkat lunak. Metode yang diusulkan diterapkan untuk menangani masalah ketidakseimbangan kelas di SDP. Hasil percobaan menunjukkan bahwa metode yang diusulkan menghasilkan peningkatan dalam kinerja prediksi untuk sebagian besar dataset.

Referensi

Arar, Ö. F., & Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. *Applied Soft Computing Journal*, *33*, 263–277. https://doi.org/10.1016/j.asoc.2015.04.045

Barandela, R., Sánchez, J. S., & Valdovinos, R. M. (2003). New Applications of Ensembles of Classifiers. *Pattern Analysis and Applications*, *6*(3), 245–256. https://doi.org/10.1007/s10044-003-0192-z

Bbeiman, L. E. O. (1996). Bagging Predictors, *140*, 123–140.

Catal, C. (2011). Expert Systems with Applications Software fault prediction : A literature review and current trends. *Expert Systems With Applications*, *38*(4), 4626–4636. https://doi.org/10.1016/j.eswa.2010.10.024

Chawla, N. V, Lazarevic, A., Hall, L. O., & Bowyer, K. W. (n.d.). SMOTEBoost : Improving Prediction, 107–119.

Chiu, T., Fang, D., Chen, J., Wang, Y., & Jeris, C. (2001). A robust and scalable clustering algorithm for mixed type attributes in large database environment. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '01*, 263–268. https://doi.org/10.1145/502512.502549

Czibula, G., Marian, Z., & Czibula, I. G. (2014). Software defect prediction using relational association rule mining. *Information Sciences*, *264*, 260–278. https://doi.org/10.1016/j.ins.2013.12.031

Galar, M., Fern, A., Barrenechea, E., & Bustince, H. (2012). Hybrid-Based Approaches, *42*(4), 463–484.

Gorunescu, F. (2011). *Data Mining: Concepts,Models and Techniques*. Berlin: Springer-Verlag Berlin Heidelberg. https://doi.org/10.1360/zd-2013-43-6-1064

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*, *28*(1), 100. https://doi.org/10.2307/2346830

Hu, S. (2009). 2009 Second International Workshop on Computer Science and Engineering MSMOTE : Improving Classification Performance when Training Data is imbalanced, 627–631. https://doi.org/10.1109/WCSE.2009.756

Jain, A, K., Murty, M, P., & Flynn, P, J. (1999). Data clustering: a review. *ACM Computing Surveys*, *31*(3), 264–323. https://doi.org/10.1145/345966.346030

Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, *58*, 388–402. https://doi.org/10.1016/j.infsof.2014.07.005

Learning, C., Liu, X., Wu, J., Zhou, Z., & Member, S. (2009). Exploratory Undersampling for, *39*(2), 539–550.

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496. https://doi.org/10.1109/TSE.2008.35

Li, D., Liu, C., & Hu, S. C. (2010). A learning method for the class imbalance problem with medical data

sets. *Computers in Biology and Medicine*, *40*(5), 509–518. https://doi.org/10.1016/j.compbiomed.2010.03.005

Lin, W., Tsai, C., Hu, Y., & Jhang, J. (2017). Clustering-based undersampling in class-imbalanced data, *410*, 17–26. https://doi.org/10.1016/j.ins.2017.05.008

Michailidou, C., Maheras, P., Arseni-Papadimititriou, A., Kolyva-Machera, F., & Anagnostopoulou, C. (2009). A study of weather types at Athens and Thessaloniki and their relationship to circulation types for the cold-wet period, part I: Two-step cluster analysis. *Theoretical and Applied Climatology*, *97*(1–2), 163–177. https://doi.org/10.1007/s00704-008-0057-x

Rana, Z. A., Mian, M. A., & Shamail, S. (2015). Improving Recall of software defect prediction models using association mining. *Knowledge-Based Systems*, *90*, 1–13. https://doi.org/10.1016/j.knosys.2015.10.009

Satish, S. M., & Bharadhwaj, S. (2010). Information search behaviour among new car buyers: A two-step cluster analysis. *IIMB Management Review*, *22*(1–2), 5–15. https://doi.org/10.1016/j.iimb.2010.03.005

Schapire, R. E. (1990). The Strength of Weak Learnability, *227*, 197–227.

Seiffert, C., Khoshgoftaar, T. M., Hulse, J. Van, & Napolitano, A. (2010). RUSBoost: A Hybrid Approach to Alleviating, *40*(1), 185–197. https://doi.org/10.1109/TSMCA.2009.2029559

Siers, M. J., & Islam, Z. (2015). Software defect prediction using a cost sensitive decision forest and voting and a potential solution to the class imbalance problem. *Information Systems*, 1–10. https://doi.org/10.1016/j.is.2015.02.006

Solis, J., Avizzano, C. A., & Bergamasco, M. (2002). Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models. *Proceedings - 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, HAPTICS 2002*, 255–262. https://doi.org/10.1109/HAPTIC.2002.998966

Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., & Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, *48*(5), 1623–1637. https://doi.org/10.1016/j.patcog.2014.11.014

Wahono, R. S., & Herman, N. S. (2014). Genetic feature selection for software defect prediction. *Advanced Science Letters*, *20*(1), 239–244. https://doi.org/10.1166/asl.2014.5283