Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th June 2024. Vol. 3. No. 3; e-ISSN : 2808-4519

Analysis of Markov Blanket Based Feature Ranking for Android Malware Detection

Yusfrizal[1]*, Andrian Syahputra[2], Yahya Tanjung[3], Safrizal[4]

[1, 2, 3, 4] *Universitas Potensi Utama*

yusfrizal80@gmail.com[1], andriansyahputra4@gmail.com[2] , yahyasbn25@gmail.com[3], rizalsyl75@gmail.com[4]*

Abstract

The ubiquity of Android applications in our daily lives has brought forth an indispensable need for robust app security mechanisms. Malware-infested applications not only jeopardize user privacy but also compromise data integrity and overall device security. Detecting and mitigating malicious behavior within Android applications is becoming increasingly challenging due to the high-dimensional nature of the data. Moving forward, employing Machine Learning (ML) techniques to detect malware in Android apps has become the norm. High dimensional feature space poses several formidable challenges, including data scarcity, over fitting, and heightened computational demands. While dimensionality reduction techniques are a potential remedy, they often result in loss of crucial information essential for comprehending and identifying malicious behaviors accurately. Feature ranking based feature subset selection emerges as a promising alternative, as it allows us to retain the original feature space, ensuring precise representation of app behaviors. This article explores various Markov Blanket (MB) based feature ranking techniques, proposes a variant of Grow Shrink (GS) MB technique, GS++ which demonstrates enhanced performance within the context of Android malware detection.

Keywords: *Malware, Feature Ranking, Markov Blanket, Android*

1. Introduction

Malware-infested applications pose significant threats to user privacy, data integrity, and overall device security. Detecting malicious behaviors in these Android applications becomes increasingly complex due to the high-dimensional nature of Android applications [1]. Detecting malware in Android apps using Machine Learning (ML) techniques is a complex task that poses various challenges. Dimensionality reduction, though a potential solution, may result in the loss of critical information integral to understanding malicious behaviors [2]. On the other hand, feature selection retains the original meaning and essence of the data, ensuring a more accurate representation of app behaviors.

Feature ranking techniques serve as prominent solutions to reduce the number of input variables [3]. In the context of ML, MB is a filter-based feature selection that has attracted much attention. The MB of a target is a set of features which are minimal that renders all other remaining attributes that have conditional independence over the target. Thus, making the MB of the target best suited for finding attributes. By selecting the reduced feature subset of the given input features, it is possible to train and test even complex learning algorithms with reduced computational cost as the classifiers [4].

MB serves as a feature ranking technique by identifying a subset of features that are directly relevant in predicting a target variable while considering their inter-dependencies. By including the parents, children and spouses of the target variable, MB encapsulates features that provide the most predictive power while minimizing the redundancy [5]. This subset of features can then be used for feature selection, prioritizing those that contribute the most to the predictive model while reducing computational complexity and over fitting.

Bayesian networks (BN) are probabilistic graphical models in which relationship between nodes in the BN graph are expressed as: P() denoting for Parent of a node, C() indicating the Child of a node and SP() standing for the Spouse of a node. Given an attribute which is the target, the feature's children, parents and co-parents of the target's children forms up together to be the MB of the target. MB encapsulates nodes dependencies and influences within the network, facilitating efficient probabilistic interference [6].

2. Research Methodology

These are the stages carried out in this research:

1. Preparation, this preparation stage is the beginning of the research process that will be carried out, the preparation carried out is determining the background of the problem, and this is done by looking for problems and obstacles that occur, by looking for information directly on Android malware detection. Formulate what problems have occurred and what the resolution process will be. Providing problem boundaries, this is done to provide limitations to this research, namely starting from the data used, variables, applications or systems built and the output that will be produced, namely Markov Blanket based feature ranking for Android malware detection. Determine the objectives, namely what results will be achieved from this research process. Benefits

of research, namely what benefits will result from research for Markov Blanket based feature ranking for Android malware detection.

2. Theoretical Study, at this stage a theoretical study will be carried out on the existing problem. The study was carried out to determine the concepts that will be used in the research, especially Markov Blanket based feature ranking for Android malware detection. Data Collection, this stage is intended to collect supporting data obtained from Android malware detection. For this reason, several methods were used to collect data, namely Observation, the thing observed was Android malware detection. Data Analysis, at this stage, supporting data will be analyzed, namely android malware detection data, by analyzing and testing Android malware detection data. Testing and Implementation, at this stage data variables and data implementation will be tested as well as system program preparation, namely by preparing the data to be analyzed, namely android malware detection data. Determine the parameters for data testing using a Markov Blanket based feature ranking process for Android malware detection. Classifying the results of the system work process, namely the results of Markov Blanket based feature ranking for Android malware detection.
3. Final stage, this stage is the stage of drawing conclusions and suggestions that can be made in preparing this research. With the conclusions, the results of the research carried out will be known and it is hoped that with suggestions there will be improvements and benefits for others. This conclusion is to answer what is the problem formulation based on the analysis that has been carried out in the previous stages.

2.1. Incremental Association Markov Blanket (IAMB)

IAMB has two phases: forward and backward. A set CMB is used to store an estimate of MB(T). During forward phase, all variables which belong to MB(T) along with many other false positives enter the CMB and during backward phase, false positives which entered during forward phase are discovered and removed from CMB. In the end CMB will be equal to MB(T) when all the false positives are removed [7].

2.2. Fast IAMB

Fast IAMB contains a growing phase and a shrinking phase. During growing phase of each iteration, it sorts features from most to least conditionally dependent, according to a heuristic function(h). Fast IAMB follows a greedy approach in the growing phase to make it more efficient. Instead of adding one feature to CMB(C) and triggering the backward phase of the algorithm, Fast-IAMB adds more features greedily which are conditionally dependent on C given the current CMB(C) therefore reducing the resorting cost for the remaining attributes after each MB modification [8].

2.3. Balanced MB (BAMB)

BAMB does not divide Parent Child (PC) learning and spouse identification into two phases. It locates the candidate PC and spouse set of Class Variable, (C) in one step and eliminates false positives from the candidate set. BAMB, in particular, uses Interleaving Forward-Backward feature selection (IFBS) technique to combine PC learning and spouse identification into a single approach. Once a new feature is added to the existing CPC(C) at each iteration, BAMB is triggered to locate the spouses of C (SP(C)) who are interested in this feature. After that, BAMB utilizes the discovered SP(C) to eliminate false positives from CP C(C), and then uses the updated CP C(C) to prune SP(C) one by one. Convergence is rapid as BAMB has no duplicate features [8].

2.4. Inter IAMB

Inter IAMB uses IFBS technique, which alternates between forward and reverse phases. If new features are introduced to the forward phase, it spontaneously starts the reverse phase and alternates between the two phases. The interleaving's purpose is to maintain CMB(C) as little as feasible throughout all stages of the algorithms' execution [9]. However, because Inter- IAMB employs the set of all presently selected features as the conditioning set, the number of data instances necessary to produce accurate findings is exponential to the size of the MB. Furthermore, it is unable to discriminate between PC and spouses in a found MB.

2.5. Semi Hilton MB

Semi-HITON-PC works in the same way as interleaved HITON-PC, but it doesn't do full variable elimination TPC (T) with each TPC (T) expansion. When a new variable is added it attempts to discards it, if it is successful. It is added to the TPC, if it is not removed (T) [10].

2.6. Simultaneous MB (STMB)

STMB method presents two novel ways for improving the computing efficiency of divide and conquers approaches in finding parents and children of each feature in the candidate set of target parent and child [11]. First, rather than the union of each feature's parents and children in CPC, STMB determines C's spouses from FCPC(C). Second, rather than employing the symmetric check, STMB eliminates false positives from CPC(C) by using the presently selected candidate spouses. STMB has been found in experiments to have comparable accuracy to other topology-based approaches.

2.7. Maximum Minimum MB (MMMB)

To discover the candidate set of parents and children of C, denoted CPC, MMMB approach employs the MMPC (Max- Min Parents and Children) algorithm. MMPC searches for the CPC(C) using the Sequential Floating Backward Search (SFBS) framework and eliminates false positives in the back- ward phase. At each iteration, the MMPC employs a greedy search technique to find the best feature from the feature set (F), CPC(C) [12]. This assesses the relationships between feature X and C on all other feasible subsets of CPC(C) and selects

the features with the least amount of association, followed by the next feature with the most amount of association among the features in FCPC(C) that are reliant on C. The independent features are removed from consideration for CPC and will never be evaluated again. The erroneous positives added by MMPC in the forward phase are deleted in the backward phase [13].

2.8. Grow Shrink MB

Grow-Shrink algorithm (GS) is the first theoretically efficient MB discovery technique, which learns MB of the class variable without learning a whole Bayesian network [14]. It comprises two stages, one for extending the candidate MB set and the other for decreasing it. If a feature is reliant on the specified goal conditional on the features now selected in the growing phase, GS adds the feature to the candidate MB set, and the Growing phase ends when all features have been verified. GS tests the Conditional Independence (CI) between a candidate and the target, conditionally on all other characteristics in the candidate set, to remove any false positives from the candidate MB set during the shrinking phase [15]. However, because GS employs the set of all presently selected features as the conditioning set, the number of data examples necessary to produce valid findings is exponential to the size of the MB . Furthermore, it is unable to discriminate between PC and spouses in a found MB.

3. Results and Discussion

Flow diagram of the proposed modified Grow and Shrink Algorithm (GS++) feature ranking mechanism is illustrated in figure 2 and explained as follows. Data collection comprises of benign and malware samples from Android applications. Features are extracted using static analysis approach resulting in a high dimensional feature vector.

Most of the existing MB discovery techniques do not perform well on datasets that doesn't hold the faithfulness assumption. Faithfulness assumption indicates that Independence's found in the data are due to separations in the true causal graph. A simple modification of the Grow and Shrink (GS) technique is proposed in this article to find Markov blankets that may contain strict 2-association as a proof of concept.

GS algorithm is divided into two parts: a grow phase in which we iteratively identify a superset of the MB of a target node T, and a shrink phase in which extraneous nodes are pruned. It is assumed that 2-adjacency faithfulness holds and that all spouses can be identified to ensure that MBs with strict

2-associations can be detected. So to achieve this there are two options, Either the target node, the spouse and the common child are connected via a strict 2-association, or the spouse node is only 1- or strictly 2-associated with the common child and not with the target node. From the generalized GS algorithm, where the grow phase has been changed in our proposed GS++ to include pairs of random variables, we can locate nodes with whom the target node is strictly 2- associated or spouses with whom a child node of T is strictly 2-associated. The shrink phase is unchanged, and it determines if singletons can be eliminated. Because we don't check for marginal dependencies, we won't remove individual nodes of a true strict 2-association to T or a child of T.

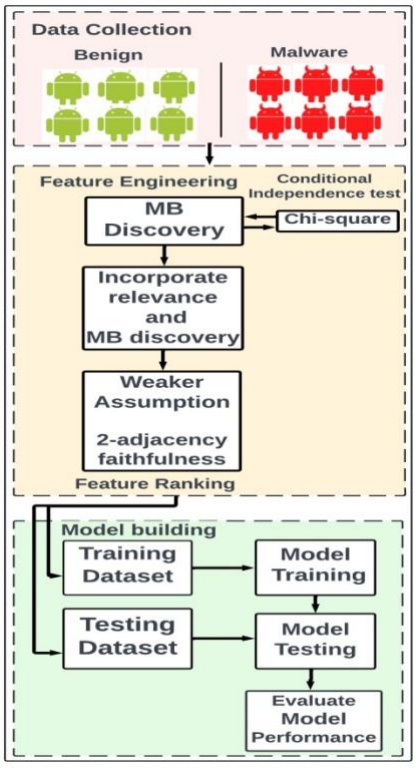

Fig. 1 : Flow of proposed GS++ MB based feature ranking technique for Android Malware Detection

We suggested 2-adjacency faithfulness as a weaker assumption in this article. We just need CI between a node and a subset of its MB that can contain up to two nodes. Features are ranked by incorporating the concept of MB. The score of a feature is measured by using a CI test. CI test is implemented for each feature depending on whether it is present in the MB of the target (T) or not. The features are then sorted in the descending order of its score (CI statistics or negative p- value). This ranking algorithm requires the MB (T) as input and it

can be obtained by any MB discovery algorithm. Testing for conditional independence is the fundamental step in ranking the features in a given dataset. CI tests are used to check if a feature is independent or NOT independent given a conditional set, which in this case is a MB set.

Chi-square test is used for testing the CI in a discrete dataset. Chi square test evaluates the relationship between two categorical variables in a contingency matrix to deter- mine whether they are statistically associated or not. Chi- square conveys the difference between the observed counts and expected counts along with the indication if there is high or low correlation between the two variables (target and the other variable).This chi-square statistic is fitted in the chi- square distribution to get a p-value. The null hypothesis and alternative hypothesis are stated. The alpha level is chosen. It is usually 0.05 (5%) or 0.01 (1%). Using the p-value and alpha value, the null hypothesis is rejected or accepted. The resulting chosen feature subset is then utilized for model building and performance evaluation is performed at inference stage.

Table 1: Example Datasets

MB	MB(T)	Features
IAMB	35,5, 752, 274, 155, 1020	6
BAMB	816, 752, 35, 697, 121, 1020, 749	7
Fast IAMB	4, 60, 273, 690, 950, 697, 1020	7
Inter IAMB	35, 5, 752, 274, 155, 1020	6
STMB	1, 21, 22, 543, 40, 41, 43, 44, 45, 46, 49, 53, 54, 59, 60, 61, 63, 67, 71, 72, 585, 586, 73, 74, 76, 82, 87, 88, 89, 90, 97, 99, 102, 103, 104, 109, 110, 112, 117, 118, 121, 127, 130, 138, 140, 144, 146, 148, 149, 150, 151, 152, 155, 158, 159, 160, 170, 190, 194, 195, 196, 203, 209, 211, 212, 214, 215, 216, 217, 218, 219, 221, 226, 227, 228, 229, 752, 243, 244, 245, 246, 256, 771, 265, 286, 292, 298, 299, 300, 301, 302, 303, 304, 824, 826, 832, 833, 836, 845, 846, 848, 849, 854, 855, 856, 859, 861, 862, 866, 867, 873, 874, 875, 886, 888, 889, 890, 891, 892, 893, 897, 898, 899, 900, 901, 903, 904, 905, 906, 912, 915, 919, 921, 413, 929, 930, 931, 933, 934, 935, 936, 937, 938, 428, 941, 942, 943, 944, 947, 948, 950, 951, 954, 955, 956, 957, 962, 963, 965, 966, 967, 970, 973, 974, 975, 978, 979, 980, 983, 984, 985, 989, 991, 992, 994, 995, 997, 1001, 1002, 1003, 1007, 1008, 1009, 1011, 1012, 1013, 1014, 1015, 1018, 1019, 1021, 1022	192
GSMB	2, 4, 5, 8, 10, 29	6
MMMB	2, 4, 907, 273, 786, 32, 33, 35, 294, 679, 47, 690, 836, 968, 201, 971, 78, 225, 997, 1001, 1010, 1012, 1018, 1020	24
semiHITON MB	2, 4, 907, 273, 786, 32, 33, 35, 294, 806, 169, 47,59, 194, 201, 971, 720, 225, 994, 995, 997, 874, 240, 1010, 1012, 1020	28
Proposed GS++	35,5,29,121,109,1010,1012,1008	8

3.1. Android Dataset

AndroZoo dataset is a collection of Android Applications from Google Play store and other sources [16]. Each app in this dataset has been analyzed by different Antivirus software to know which malware are. For demonstration of the efficiency of considered Markov Blanket based feature ranking techniques over existing filter based feature selection technique like Chi square techniques. Considered dataset contains 35000 malware samples and 35000 benign samples and 1024 processed features. For the considered binary classification task, target label 1 is used to denote malware and 0 is used to denote safe apps. Dataset is split into training, validation and testing datasets in the ratio of 60:20:20. All experiments were performed on Google colab with Python language. Over fitting was avoided by conducting 10 fold cross validation and stable results were obtained. For the purpose of finding the effective- ness of the proposed GS++ MB technique along with the other considered MB techniques, standard classifiers such as Multi-Layer Perceptron (MLP), Random Forest (RF), Decision Tree (DT), Adaptive Boosting (ADABOOST) and Gaussian Process Classifier (GPC) are utilized. Model effectiveness is assessed using the accuracy performance metric.

3.2. Analyzing MB

MB discovery is analyzed by using existing 8 different types of MB discovery algorithms for 1024 features along with proposed GS++ MB technique as tabulated in Table 1 and illustrated in figure 3. Selecting the optimal number of ranked features to be fed into the classifier is performed empirically by providing different cardinality of ranked features to the classifiers. IAMB technique is used as a base MB feature ranking technique to choose the optimal number of features to be used for comparing against the other MB options. Table 2 tabulates the accuracy metric of the considered ML models with features ranked using IAMB technique for various cardinality options such as 128, 256, 384, 512, 640, 768, 896 and 1024. MLP classifier has attained high accuracy of 0.956 when trained against $IAMB_{640}$. RF classifier has attained high accuracy of 0.874 when trained against $IAMB_{1024}$. DT classifier has attained high accuracy of

0.918 when trained against $IAMB_{512}$. AdaBoost classifier has attained high accuracy of 0.942 when trained against $IAMB_{768}$. GPC classifier has attained high accuracy of 0.954 when trained against $IAMB_{256}$ and $IAMB_{384}$.

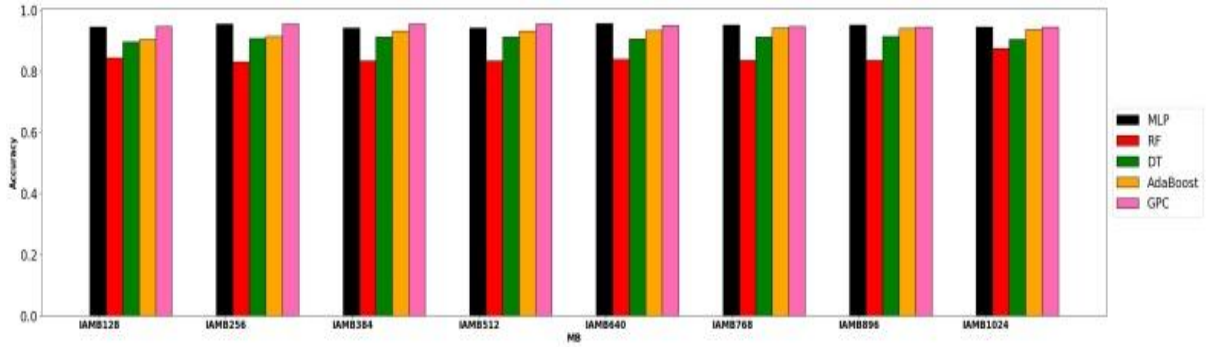

Fig. 2: Classifier Accuracy for various feature cardinality using IAMB

3.3. Performance Evaluation

Experiments are conducted to identify the best combination of MB methods and the classifiers with top 256 ranked features. Performance of existing MB discovery techniques and proposed GS++ MB technique in combination with various classifiers are analyzed using their accuracy in Table 3 and illustrated in figure 4. GS++ MB performs better than existing filter based feature selection technique and other existing MB techniques.

Table 2: Comparison of Classifier Accuracy with Different Input Sizes for Iamb Technique

Features Ranked with IAMB	Classifier Accuracy				
	MLP	RF	DT	AdaBoost	GPC
$IAMB_{128}$	0.944	0.844	0.896	0.904	0.946
$IAMB_{256}$	0.954	0.83	0.908	0.914	0.954
$IAMB_{384}$	0.942	0.834	0.912	0.932	0.954
$IAMB_{512}$	0.942	0.834	0.912	0.932	0.954
$IAMB_{640}$	0.956	0.838	0.906	0.934	0.95
$IAMB_{768}$	0.952	0.836	0.912	0.942	0.946
$IAMB_{896}$	0.952	0.836	0.914	0.94	0.944
$IAMB_{1024}$	0.944	0.874	0.904	0.936	0.944

Table 3: Comparison of Classifier Accuracy with Different Input Sizes for Iamb Technique

Feature Selection Technique	MLP	RF	DT	AdaBoost	GPC
$IAMB_{256}$	0.954	0.83	0.908	0.914	0.954
$BAMB_{256}$	0.942	0.84	0.916	0.922	0.946
Fast $IAMB_{256}$	0.946	0.828	0.91	0.922	0.948
Inter $IAMB_{256}$	0.954	0.83	0.908	0.914	0.954
$STMB_{256}$	0.95	0.84	0.926	0.912	0.954
$GSMB_{256}$	0.946	0.838	0.908	0.936	0.946
$MMMB_{256}$	0.942	0.776	0.92	0.916	0.952
semi HITON MB_{256}	0.938	0.832	0.908	0.924	0.958
existing Chi^2	0.938	0.868	0.912	0.92	0.94
existing $SVM_{W V}$	0.944	0.874	0.904	0.936	0.944
Proposed $GS{+}{+}_{256}$	0.955	0.87	0.929	0.939	0.957

1) Comparison of GS++ against existing MB discovery techniques: Table 3 tabulates the performance evaluation of proposed GS++ against existing MB techniques such as IAMB, BAMB, Fast IAMB, Inter IAMB, STMB, GSMB, MMMB, Semi Hiton MB. MLP classifier has attained high accuracy of 0.955 when comparing with IAMB. RF classifier has attained high accuracy of 0.87 when comparing with GSMB. DT classifier has attained high accuracy of 0.929 when comparing with STMB. AdaBoost classifier has attained high accuracy of 0.939 when comparing with GSMB. GPC classifier has attained high accuracy of 0.957 when comparing with Semi Hiton MB.

2) Comparison of GS++ against existing filter based feature selection: Support Vector Machine with Natural Weights ($SVM_{W V}$) and Chi2 techniques: Proposed Grow and Shrink variant, GS++ MB based feature ranking technique has better accuracy than other filter-based ranking techniques like chi2 and SVMW V techniques. MLP classifier has attained high accuracy of 0.955, RF classifier has attained high accuracy of 0.87, DT classifier has attained high accuracy of 0.929, AdaBoost classifier has attained high accuracy of 0.939. GPC classifier has attained high accuracy of 0.957 when comparing with existing chi2 and $SVM_{W V}$.

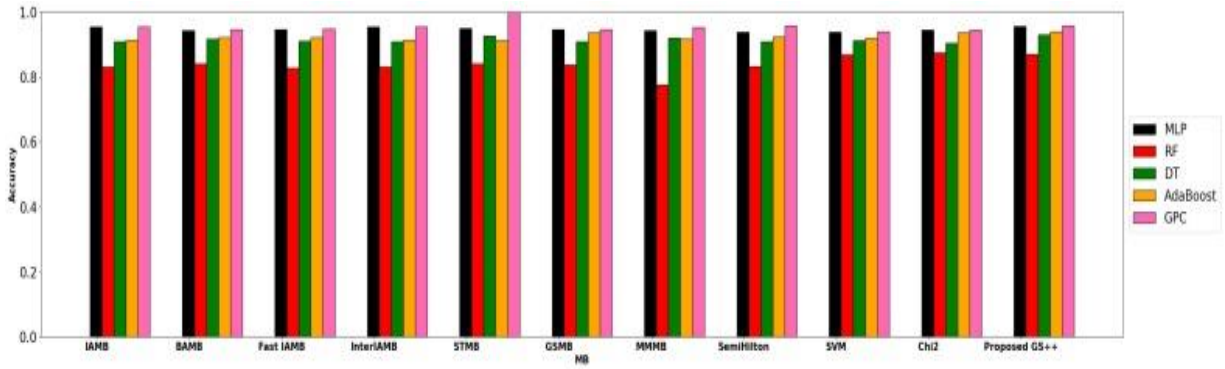

Fig. 3: Comparison of proposed GS++ MB with other existing techniques

4. Conclusion

The ever-expanding landscape of Android applications has necessitated a robust approach to ensuring app security, particularly in the face of growing menace of malware. This article has addressed the formidable challenges posed by high- dimensional data in the realm of Android malware detection. Proposed GS++ MB feature ranking technique has effectively identified a minimal set of features that maintain conditional independence from the target variable, streamlining the process of Android malware detection than existing MB techniques. The future research could focus on improving the efficiency and accuracy of MB discovery on nodes in a BN with a large parent-children size. Current chosen MB based technique can be compared state-of-the-art baselines, such as using genetic algorithm, simulated annealing or reinforcement learning as future work.

Acknowledgement

Thank you to all those who have helped complete this research.

References

[1] U. J. Butt, M. F. Abbod, and A. Kumar, "Cyber threat ransomware and marketing to networked consumers," in *Handbook of research on innovations in technology and marketing for the connected consumer*, IGI Global, 2020, pp. 155–185.

[2] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–36, 2020.

[3] S. He, F. Guo, and Q. Zou, "MRMD2. 0: a python tool for machine learning with feature ranking and reduction," *Curr. Bioinform.*, vol. 15, no. 10, pp. 1213–1221, 2020.

[4] Z. Ling, K. Yu, Y. Zhang, L. Liu, and J. Li, "Causal learner: A toolbox for causal structure and markov blanket learning," *Pattern Recognit. Lett.*, vol. 163, pp. 92–95, 2022.

[5] H. H. Htun, M. Biehl, and N. Petkov, "Survey of feature selection and extraction techniques for stock market prediction," *Financ. Innov.*, vol. 9, no. 1, p. 26, 2023.

[6] M. Harris and M. Zwick, "Graphical Models in Reconstructability Analysis and Bayesian Networks," *Entropy*, vol. 23, no. 8, p. 986, 2021.

[7] X. Wu, B. Jiang, X. Wang, T. Ban, and H. Chen, "Feature selection in the data stream based on incremental Markov boundary learning," *IEEE Trans. Neural Networks Learn. Syst.*, 2023.

[8] H. Wang, Z. Ling, K. Yu, and X. Wu, "Towards efficient and effective discovery of Markov blankets for feature selection," *Inf. Sci. (Ny).*, vol. 509, pp. 227–242, 2020.

[9] K. Yu *et al.*, "Causality-based feature selection: Methods and evaluations," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–36, 2020.

[10] J. Lemmon *et al.*, "Evaluation of feature selection methods for preserving machine learning performance in the presence of temporal dataset shift in clinical medicine," *Methods Inf. Med.*, vol. 62, no. 01/02, pp. 60–70, 2023.

[11] N. Wang, H. Liu, L. Zhang, Y. Cai, and Q. Shi, "Loose-to-strict Markov blanket learning algorithm for feature selection," *Knowledge-Based Syst.*, vol. 283, p. 111216, 2024.

[12] R. Duangsoithong and Y. Zhao, "Floating search and conditional independence testing for causal feature selection.," *Songklanakarin J. Sci. Technol.*, vol. 43, no. 6, 2021.

[13] S. del Río and E. Villanueva, "A Novel Method to Estimate Parents and Children for Local Bayesian Network Learning," in *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 2*, 2022, pp. 468–485.

[14] M. A. Javidian, M. Valtorta, and P. Jamshidi, "Learning LWF chain graphs: A Markov blanket discovery approach," in *Conference on Uncertainty in Artificial Intelligence*, 2020, pp. 1069–1078.

[15] A. Hassan, J. H. Paik, S. Khare, and S. A. Hassan, "Ppfs: Predictive permutation feature selection," *arXiv Prepr. arXiv2110.10713*, 2021.

[16] P. Liu, L. Li, Y. Zhao, X. Sun, and J. Grundy, "Androzooopen: Collecting large-scale open source android apps for the research community," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 548–552.