



Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms

Thatipelli Sai Sharan,

Department of CSE, CMREC, Hyderabad, Telangana, India - 501401

Email: 248r1d5804@cmrec.ac.in

E. Suresh Babu,

Associate Professor, Department of CSE, CMREC, Hyderabad, Telangana, India - 501401

Email: sureshbabu.cmrec@gmail.com

Abstract— With the rising volume of Android applications, the threats of malware attacks have grown significantly, so there is a growing need for legitimate malware detection is a critical precondition for mobile device security. Common signature-based detection methods are often not able to detect new or disguised malware, so more intelligent detection methods are needed. In this paper, we offer a ML approach to detect malware in Android applications that is based on program properties obtained from the analysis of the application's static structure. The tools for reverse engineering are Jadx and Androguard, which are used to extract crucial static information from the APKs including permissions, API calls, intents, packages, services, and receivers. The above obtained traits are transformed into binary feature vectors and then trained and tested with three classifiers; namely, SVM, ANN and ANN-GA. The experiments were conducted on a set of 3,799 Android apps that were divided into 3,039 training and 760 testing samples. Comparative analysis shows that SVM classifier is able to detect highest accuracy (98.95%) which is better than ANN (98.68%) and ANN-GA (94.87%). The findings demonstrate that using reverse engineering, static feature analysis and ML is an accurate, efficient and scalable way to detect Android malware, without sacrificing classification accuracy for benign and malicious applications.

Index Terms — *Android Malware Detection, Reverse Engineering, Static Analysis, Machine Learning, Support Vector Machine (SVM), Feature Extraction.*

I. INTRODUCTION

Smartphones have transformed the mobile world with Android's rise giving customers the ability to use a flexible platform to communicate, entertain, access finances and business applications with a range of features. Android's open source design combined with the massive application environment has always made it the most popular cell phone operating system worldwide and a tempting goal for hackers to take advantage of application weaknesses. The number of Android apps keeps rising, and so do the intricacy of malware attacks. These attacks are designed to infringe on user privacy, steal information and trigger unwanted financial transactions and affect the normal operation of the device. Hence, security issues of Android applications are the top concerns of researchers, developers and cybersecurity experts.

The traditional Android malware detection approaches are based on signature detection which can only detect known malware variants. But, the technologies have their drawbacks as far as detecting new, polymorphic and obfuscated malware that's constantly evolving to evade the standard security techniques. To address these limitations, ML has been suggested as a potential solution to learn discriminative patterns from application attributes and to automatically classify the applications as benign or malicious, based solely

on signatures that have been pre-defined [2]. In recent years, static analysis techniques have been proved to be efficient alternatives for extracting informative characteristics from the APKs, without running the application, which reduces computation overhead while maintaining high detection accuracy [8].

Among the various static analysis techniques, reverse engineering has been a subject of much interest due to its ability to uncover the internal characteristics of the application which are not easily discerned using conventional inspection techniques. The AndroidManifest.xml file and the classes.dex bytecode can be important information to extract, using reverse engineering tools like Jadx and Androguard. This helps to perform a systematic analysis of features such as permissions, API calls, intents, packages, services and receivers. Previous works have shown that combining analysis and API-level information can significantly enhance the detection accuracy of the malware by applying the single feature type [10]. Likewise, a set of a few static feature sets can provide a complete description of application behaviour, enabling ML models to distinguish the malicious applications more accurately [13]. A reverse engineering is now an indispensable preprocessing phase for today's modern Android malware detection systems [20].

Encouraged by such advancements, a ML framework for Android Malware detection using Static features of APK files, which are reverse engineered is proposed in this paper. The proposed system generates a binary feature vector using the features retrieved from the application and evaluates the recognition performance of three classifiers: SVM, ANN and ANN-GA. We compare the performance against a set of malicious and benign Android Apps to understand the effectiveness of each model. SVM is selected as the main classifier due to its ability to perform well in a high dimensional feature space, especially in the context of Android Malware classification [9]. Furthermore, it compares its performance with the ANN based methods to get a comprehensive analysis of various ML techniques.

This involves the design of an efficient framework which integrates the reverse engineering, static feature extraction and ML classification of Android malware into one process. The proposed system is computationally efficient because there is no need to run the program at runtime and has high detection accuracy by the informative representation of features. The experimental results indicate that the proposed approach can be used to correctly classify malicious applications, which proves its applicability for real Android security applications, and establishes a good basis for future intelligent mobile malware detection systems.

II. RELATED WORK

The emergence of malicious applications for mobile platforms, has spurred a great deal of research into the detection of Android malware. In general, early detection methods were based on signature detection, which match application attributes to known signatures. Unfortunately, these technologies are effective at identifying known malware families, but can not identify new malware families, polymorphic threats and disguised threats. To overcome these limitations, static and dynamic analysis techniques have been very popular among researchers for analysing application behavior or analyzing code attributes to identify malicious behavior. Among these methods, static analysis has attracted a lot of attention as it allows malware identification without running apps, therefore decreasing the computational cost and security threats during analysis [2]. Furthermore, recent research has pointed out the importance of creating representative feature spaces to capture malicious app behaviour through permissions, API calls, intents, packages etc., which are static features [8].

Android malware detection is significantly enhanced using a ML approach, with the automated malware classification based on the app features gathered. Supervised learning algorithms are those that learn discriminative patterns from labeled data and transfer this knowledge to new applications that are not seen during training. SVM, DT, NB, RF and KNN have been widely studied for Android malware classification due to their capacity to efficiently handle the

high dimensional data space. SVM has always been a great classification technique and among all these techniques, maximising the decision border between benign and dangerous programs is the most suitable for static malware analysis [9]. Furthermore, a lot of works have demonstrated that if major permission and API-based features are utilized, the accuracy of the detection can be greatly enhanced and the redundant information can be minimized during the model training [10].

To further enhance the classification performance, researchers have worked on ensemble learning and feature fusion algorithms, which fuse multiple classifiers or fuse multiple feature categories. In ensemble methods such as AdaBoost, the predictive strengths are made more accurate by training on misclassified samples in every iteration, and combining weak learners with a strong learner [4]. Likewise, hybrid architectures (incorporating both permission-based features and API call metadata) have also been found to be more resilient than any single-feature approach to give a fuller application behavior profile [13]. Reverse engineering became also an important method for pre-processing to retrieve static information from APK files. [20] By reverse engineering AndroidManifest.xml and classes.dex bytecode, secret permissions, API usage, and structural characteristics can be identified, which can be used to make malware classification more reliable. These improvements have resulted in static feature extraction combined with ML being one of the most feasible ways for Android malware detection.

A lot has been done in the last few of years but there are still some issues to be dealt with. Numerous recent research applications have only explored limited feature sets, small datasets, or complex detection architectures, thereby adding computational complexity and reducing the potential for scaling to a real-world system. Some DL models have proven to be very effective in detecting malware. But they generally require large datasets to train, large computational power and longer training times, limiting their use in light weight security applications [16]. Not to mention, not all detection frameworks are based on feature diversity or classification performance but on a balanced evaluation of multiple ML algorithms within a single experimental setting. To overcome these limitations, the present work proposes a unified Android malware detection system which combines reverse engineering, extensive static feature extraction and comparative evaluation of SVM, ANN and ANN with Genetic Algorithm models. The proposed model allows high detection accuracy and low computing cost, which is appropriate for the actual malware analysis of Android and deployment.

III. METHODOLOGY

A. Overall Proposed Framework

The suggested framework shows a systematic method to Android malware detection by use of the reverse engineering and ML approaches. First, the prepared dataset contains

APKs that are extracted by reverse engineering the main components of the application without running the applications. Static information, like permissions, API calls, intents, packages, services and receivers, is extracted by analyzing decompiled files, and then binary-encoded as feature vectors for ML. The processed data is then split into two groups: training and testing group, for training the classification models by SVM, ANN and ANN-GA. Lastly, the trained models classify incoming Android applications as benign or malicious using the feature patterns retrieved.

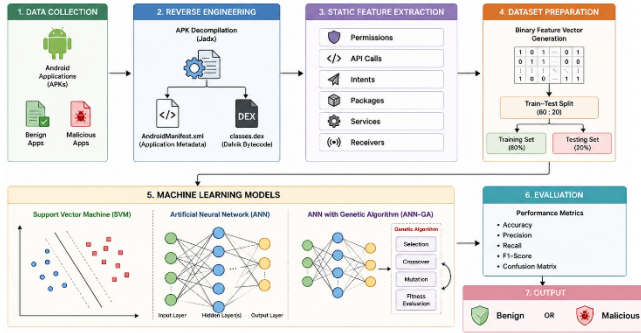


Fig.1 System Architecture

The entire design of the proposed Android malware detection system is shown in Figure 1. The method is initiated from the acquisition of the APK and then it follows the reverse engineering and static feature extraction process to generate binary feature vectors. The feature vectors are then fed into several ML models for training and testing of the models. The trained classifiers will be used to analyse the unknown Android apps. The last step towards the prediction will take place by the recognition of benign or dangerous class from the learnt feature patterns.

B. Reverse Engineering

Reverse engineering is used to analyse APKs statically without executing them, which is a secure and efficient way to undertake malware analysis. In the proposed framework, each APK will be decompiled using the Jadx tool, which will convert the application to a human-readable format for in-depth analysis. The decompilation process generates two key outputs: AndroidManifest.xml which contains application metadata such as permissions, activities, services and broadcast receivers; classes.dex which contains the decompiled Dalvik bytecode containing API calls and application logic. All the collected components have all the required information for static feature extraction and later for malware categorisation using ML techniques.

C. Static Feature Extraction

The reverse-engineered APK files are then analyzed by static feature extraction to extract useful features to distinguish between benign apps and malicious apps. The proposed framework is able to extract various categories of static features from AndroidManifest.xml file & classes.dex byte code such as permissions, API calls, intents, packages, services and receivers. Permissions describe the resources that an application is looking for and API calls expose the

functional behaviour of the application. Learn how components interact with each other and do background work: intents, packages, services and receivers. These features are retrieved and encoded into binary feature vectors, resulting in a structured dataset as input to train and evaluate ML classifiers.

D. Dataset Preparation

The set of data used for this study consists of two types of Android applications: benign and malicious ones that have been collected for the purpose of malware classification. After reverse engineering and static feature extraction, features extracted are converted to binary feature vectors, with the binary value being either present or not. This format reduces the size of the feature space to a format more conducive to ML. Full data set has 3,799 Android apps and is split into training and testing sets using 80:20. Overall, 3039 samples are used to train the model and 760 samples are used for the performance evaluation. Table I summarises the statistics of the dataset.

Table I. Dataset Statistics

Parameter	Value
Total Dataset Size	3,799
Training Samples	3,039
Testing Samples	760
Training Ratio	80%
Testing Ratio	20%

E. Machine Learning Models

ML models discover the discriminative patterns in the retrieved static information to categorise Android applications into benign or harmful. This paper evaluated three supervised learning models to see how well they perform in detecting malware in Android applications.

Support Vector Machine (SVM): SVM is a supervised classification method that finds the ideal decision boundary to maximise the separation between benign and malicious applications. It can be used for detecting malware having binary static features and can handle high dimensional feature space quite well. SVM has great generalisation capability, minimises the classification errors and ensures high prediction accuracy which makes SVM as the principal model in the proposed framework.

Artificial Neural Network (ANN): ANN is a computer model which is inspired from the structure of biological neural networks. It is composed of interconnected input, hidden and output layers that learn complicated nonlinear correlations between extracted features. The network is trained to learn to adjust the weights of the links iteratively in order to enhance the classification performance. ANN can learn complex interactions among features and is able to accurately identify malware in many Android applications.

ANN with Genetic Algorithm (ANN-GA): NN with Genetic Algorithm is a combination of learning ability of ANN and optimisation capability of GA. Genetic Algorithm

is used to optimise the parameters of the network by using selection, crossover and mutation to search for better weight configurations for the model. The hybrid technique is expected to boost the classification performance and lower the probability of falling into sub-optimal solutions in network training.

The comparative analysis of the three ML models enables them to be comprehensively compared for their classification capabilities, computational efficiency and suitability for the detection of static Android malware based on reverse engineered application properties.

IV. RESULTS & DISCUSSION

A. Experimental Setup

The proposed framework for detection of Android malware has been implemented on a windows-based system with Intel core i5 processor and 4 GB of RAM and implemented in python 3.8. The reverse engineering and feature extraction were performed using the Jadx tool and Androguard. Python libraries like Scikit-learn, NumPy, and Pandas were used to build ML models. To evaluate the three-mentioned supervised learning models, they were SVM, ANN and AANN-GA. The model was assessed by common evaluation indicators such as accuracy, precision, recall, and F1-score to quantify the classification performance of the model.

B. Comparative Performance Analysis

To test the performance of the proposed Android Malware Detection framework, 3 ML models were selected and compared, i.e., SVM, ANN and ANN-GA. The entire experimental pipeline starts from uploading the prepared dataset, partitioning the dataset into training and testing subsets, training the model and evaluating the performance. The interface used for uploading the dataset is depicted in Figure 2 and the generation of the training and testing datasets used for model development shown in Figure 3.

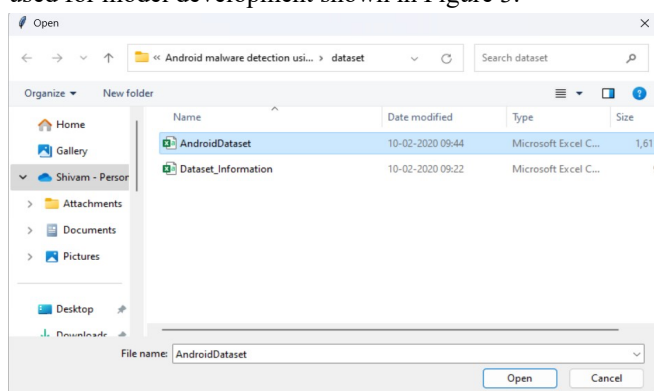


Fig.2. Uploading Dataset

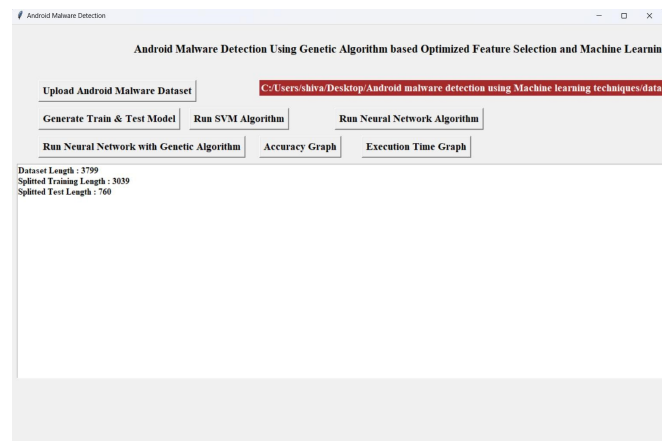


Fig.3. Generating Training and Testing Data

Each of the ML models has been trained separately, with the same training dataset after the data has been preprocessed, and tested on the test set. Figures 4 shows the prediction precision of SVM while Figures 5 and 6 show the performance of Artificial Neural Network and ANN with GA models respectively. The numbers are used to indicate how successfully each of the classifiers was executed and the rate of prediction accuracy.

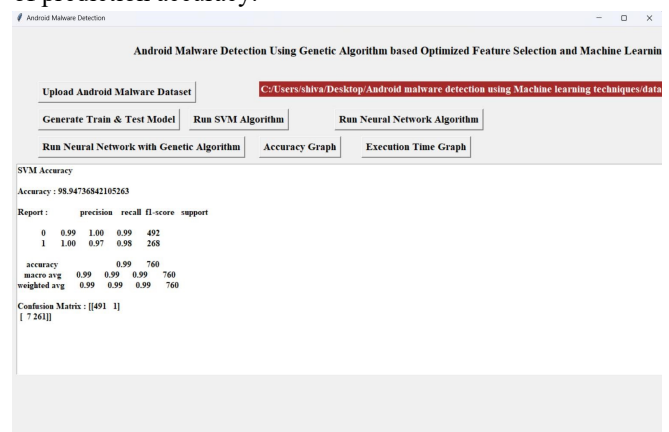


Fig.4. Calculates Accuracy using SVM

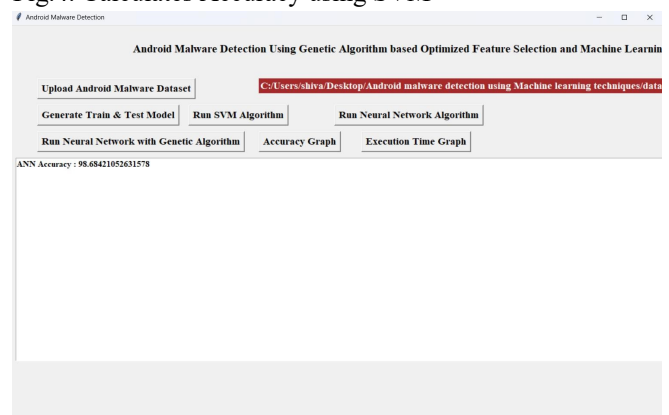


Fig.5. Calculates Accuracy using ANN

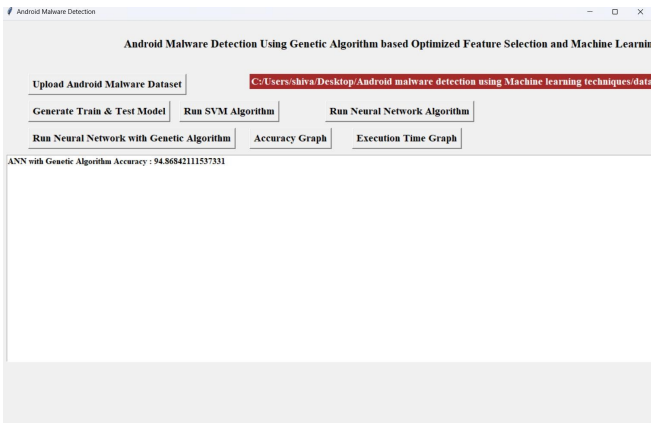


Fig.6. Calculates Accuracy using ANN with Genetic Algorithm

Table II shows the overall classification performance of the three ML models. Among the classifiers analyzed, the SVM obtained the highest classification accuracy of 98.95% while the Artificial Neural Network obtained an accuracy of 98.68% closely. The accuracy achieved by the ANN with the GA was 94.87% that is comparatively low performance for the provided dataset. Based on this finding, the SVM is the best classifier for the retrieved static feature set.

Table II. Model Performance Comparison

Machine Learning Model	Accuracy (%)
Support Vector Machine (SVM)	98.95
Artificial Neural Network (ANN)	98.68
ANN with Genetic Algorithm (ANN-GA)	94.87

Detailed classification performance of the highest performing model, SVM, is shown in Table III. The classifier achieved 0.99 precision on benign samples and 1.00 on malware samples and high recall and F1-score values for both classes were sustained. The results show that the model can remove the malicious application from the benign with very little error.

Table III. SVM Classification Report

Class	Precision	Recall	F1-Score	Support
Benign (0)	0.99	1.00	0.99	492
Malware (1)	1.00	0.97	0.98	268
Accuracy	0.99	0.99	0.99	760

The overall prediction accuracy of the classifiers is presented in Figure \ref{fig:Overall_Acc} and in this case SVM has the best performance in terms of detection accuracy among the analysed classifiers. In addition to the prediction performance, computational efficiency of the detection system is also important for realistic detection. The execution time for each classifier is given in Figure 8, which shows a comparison of the computational cost of model execution.

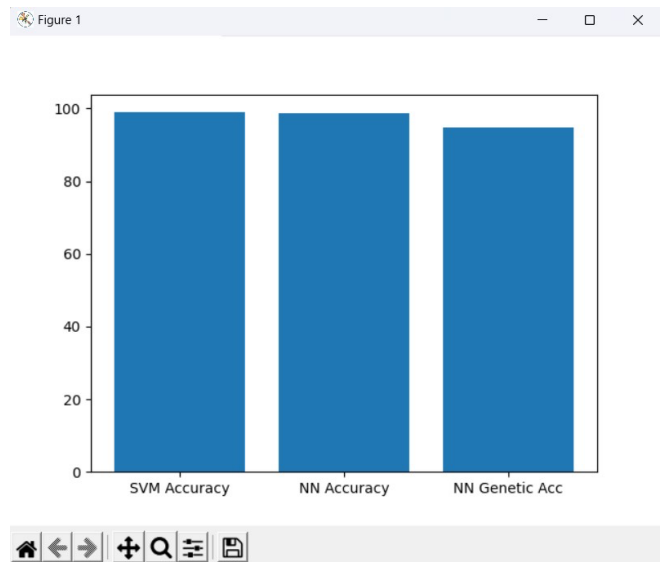


Fig.7. Accuracy of Different Algorithms

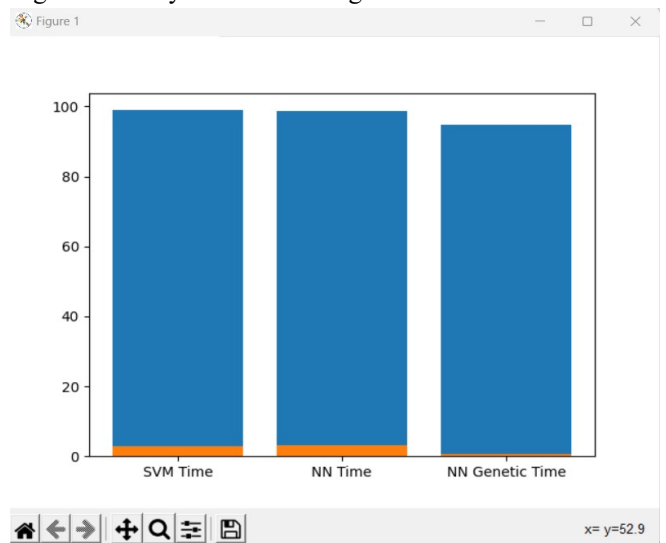


Fig.8. Time Taken for Execution by Different Algorithms

Based on the comparison study the optimum trade-off of classification accuracy and computational efficiency for the given reverse engineered static features is SVM. This constant precision, recall and F1 score further confirm its usefulness in the recognition of Android malware without high false classification rates.

C. Discussion

From the experimental results, it is observed that SVM is better compared to other models tried because SVM can optimally separate the high dimension binary feature vectors with an optimal decision boundary. The ANN is able to perform the same tasks and learns nonlinear feature associations during training, but it needs more computational power during training. In contrast, ANN-GA resulted in lower accuracy, which may be attributed to the extra optimisation process and the complexity of the model was increased for this data. Reverse-engineered static properties proved to be a good way to characterize malware without running the application, thereby reducing the cost of computing and achieving an outstanding detection accuracy.

The results demonstrate the practicability of the proposed framework in terms of its computational efficiency and applicability to real-world Android malware detection apps.

V. CONCLUSION

We proposed in this paper a ML based Android malware detection system that uses static features taken from reverse-engineered APKs. The proposed approach involves decompiling APKs, static extraction of features, creation of binary feature vectors and supervised ML to accurately classify the apps as malicious or benign. The number of Android apps in this work is 3,799, for which three classification models are evaluated: SVM, ANN and ANN-GA. Based on experimental results, it was observed that the SVM classifier achieved highest detection accuracy 98.95% followed by ANN (98.68%) and ANN-GA (94.87%). Results from the comparison analysis confirmed that the reverse-engineered static features are informative enough for malware classification and obviate the need for the application's execution. Furthermore, the suggested system provides low computing cost, dependable classification performance and scalability for actual application. Our results well indicate that reverse engineering, combined with static feature analysis and ML, is an efficient and reliable technique to uncover Android malware within the modern-day mobile security setting.

The suggested system can be enhanced by adding aspects of dynamic analysis to enhance the identification of complicated and obfuscated malware. Future research includes focusing on DL architectures, transformer-based models, and hybrid detection methods based on static and dynamic properties. In addition, the incorporation of real-time malware monitoring, automatic feature selection and cloud-based deployment can boost the detection accuracy, scalability and applicability in large-scale Android security systems.

REFERENCES

- [1] A. M. Syed, M. U. Akram, T. Akram, M. Muzammal, S. Khalid, and M. A. Khan, "Fundus images-based detection and grading of macular edema using robust macula localization," *IEEE Access*, vol. 6, pp. 58784–58793, 2018.
- [2] A. Cahn, A. Shoshan, T. Sagiv et al., "Prediction of progression from pre-diabetes to diabetes: development and validation of a machine learning model," *Diabetes*, vol. 36, no. 2, pp. 1–8, 2020.
- [3] Veena Vijayan V. And Anjali C, Prediction and Diagnosis of Diabetes Mellitus, "A Machine Learning Approach", 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS) | 10- 12 December 2015 | Trivandrum.
- [4] M. K. Hasan, M. A. Alam, D. Das, E. Hossain, and M. Hasan, "Diabetes prediction using ensembling of different machine learning classifiers," *IEEE Access*, vol. 8, pp. 76516–76531, 2020.
- [5] S. Islam Ayon, and M. Milon Islam, "Diabetes prediction: a deep learning approach," *International Journal of Information Engineering and Electronic Business*, vol. 11, no. 2, pp. 21–27, 2019.
- [6] M. Maniruzzaman, M. J. Rahman, B. Ahammed, and M. M. Abedin, "Classification and prediction of diabetes disease using machine learning paradigm," *Health Information Science and Systems*, vol. 8, no. 1, pp. 1–14, 2020.
- [7] P. Suresh Kumar and V. Umatejaswi, "Diagnosing Diabetes using Data Mining Techniques", *International Journal of Scientific and Research Publications*, Volume 7, Issue 6, June 2017 705 ISSN 2250-3153, pp. 705-709.
- [8] Ridam Pal, Dr. Jayanta Poray, and Mainak Sen, "Application of Machine Learning Algorithms on Diabetic Retinopathy", 2017 2nd IEEE International Conference on Recent Trends in Electronics Information & Communication Technology, May 19-20, 2017, India.
- [9] Berina Alic, Lejla Gurbeta and Almir Badnjevic, "Machine Learning Techniques for Classification of Diabetes and Cardiovascular Diseases", 2017 6th Mediterranean Conference on Embedded Computing (MECO), 11-15 JUNE 2017, BAR, MONTENEGRO.
- [10] Dr. M. Renuka Devi and J. Maria Shyla, "Analysis of Various Data Mining Techniques to Predict Diabetes Mellitus", *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 11, Number 1 (2016) pp 727-730 © Research India Publications. <http://www.ripublication.com>
- [11] Martinsson, John, Alexander Schliep, Bjorn Eliasson, Christian Meijner, Simon Persson, and Olof Mogren. "Automatic blood glucose prediction with confidence using recurrent neural networks." In *Khd@ ijcai*. 2018.
- [12] Minyechil Alehegn and Rahul Joshi, "Analysis and prediction of diabetes diseases using machine learning algorithm": Ensemble approach, *International Research Journal of Engineering and Technology* Volume: 04 Issue: 10 | Oct -2017
- [13] T. E. Idriss, A. Idri, I. Abnane and Z. Bakkoury, "Predicting Blood Glucose using an LSTM Neural Network," 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), 2019, pp. 35-41, doi: 10.15439/2019F159.
- [14] Michael Kahn, MD, PhD, Washington University, St. Louis, MO, <https://archive.ics.uci.edu/ml/datasets/diabetes> (accessed on 10/06/2022).
- [15] D. Zeevi et al., "Personalized nutrition prediction using machine learning," *Nature*, vol. 528, no. 7583, pp. 484–489, 2015.

- [16] V. Borisov et al., "Deep learning for dynamic treatment regimes in chronic diseases," *J. Med. Syst.*, vol. 45, no. 4, pp. 1–12, 2021.
- [17] E. Daskalaki et al., "Real-time glucose prediction in type 1 diabetes using adaptive models," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 6, pp. 1211–1221, 2016.