International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Similarity Detection in Large Volume Data using Machine Learning Techniques

Viji Gopal, Varghese Paul, M Sudheep Elayidom, Sasi Gopalan

Abstract: ***When unauthorized copying or stealing of intellectual properties of others happen, it is called plagiarism. Two main approaches are used to counter this problem – external plagiarism detection and intrinsic plagiarism detection. External algorithms compare a suspicious file with numerous sources whereas intrinsic algorithms are allowed to solely inspect the suspicious file in order to predict plagiarism. In this work, the area chosen for detecting plagiarism is with programs or source code files. Copying the entire source code or logic used in a particular program without permissions or copyright is the stealing that happens in the case of source codes. There exist many ways to detect plagiarism in source code files. To perform plagiarism checking for a large dataset, the computational cost is very high and moreover it's a time consuming job. To achieve a computationally efficient similarity detection in source code files, the Hadoop framework is used where parallel computation is possible for large datasets. But the raw data available to us is not in a suitable form for the existing plagiarism checking tools to work with, as their size is too high and they possess features of big data. Thus a qualifying model is required for the dataset, to be fed into Hadoop so that it could efficiently process them to check for plagiarism in source codes. To generate such a model, machine learning is used which incorporates big data with machine learning.***

Index Terms: ***Plagiarism, Big Data, Similarity, Hadoop, Machine Learning***

I. INTRODUCTION

Based on continuing development of technology, applications in the field of software increase correspondingly and plagiarism stands out as a big problem. For example, using a source code of a program without permission which is developed specifically for a company is a common plagiarism situation. Another example can be seen is in academics. Especially, programming course instructors indicate that source code theft issues pose a major problem while evaluating students' projects and assignments.

Turning in someone else's work as your own, changing words but copying the sentence structure of a source without giving credit, or copying words or ideas from someone else without giving credit and also copying so many words or ideas from a source that it makes up majority of the work, whether you give credit or not, all these come under the area of plagiarism. Giving incorrect information about the source of a quotation or failing to put a quotation in quotation marks is also an act of plagiarism. In the case of source code files plagiarism is mainly done by replacing the original word with another word for identifiers and changing the structure or arrangement of the program. Certain plagiarists will try to understand the logic of the program and then will rewrite a new program.

There are many methods to understand whether the code is stolen or not and how to prevent code theft. One of these methods is evaluating a software tool which finds similarity ratios among source codes. The necessary steps to solve plagiarism problems on source codes are much harder than natural language processing (NLP) [11]. The traditional method is extracting source code metrics before similarity check. However, this traditional approach is not free from certain disadvantages.

Plagiarism checking can be combined with sequence alignment to obtain a good result in detection of duplicate contents. Therefore, performing this operation for a large corpus is computationally costly and also requires data space. So achieving an efficient computation in this area is a need.

The method proposed tries to find out plagiarized documents within source code files by performing global alignment using the Needleman-Wunsch algorithm in a very less expensive and computationally effective manner using the Hadoop framework. For this method to work successfully in big data, machine learning is used to study the behaviour of the corpus and finally generate a quality model from the analysis with which plagiarism detection can be done on source code files.

The paper is divided into following sections. Section II discusses the Literature Survey, a critical appraisal of the previous works published in the literature pertaining to the topic of the investigation. Section III is methodology that specifies the methods used to perform and analyze the work. Section IV explains the experiments done and results obtained from the work, which is followed by conclusion and future works in section V.

Manuscript published on 30 September 2019
* Correspondence Author
Viji Gopal*, Department of Information Technology, School of Engineering, Cochin University of Science and Technology, Cochin, Kerala, India. email: viji@scmsgroup.org.
Dr. Varghese Paul, Professor, Department of Information Technology, Rajagiri School of Engineering and Technology, Kakkanad, Kochi, Kerala, India. email: vp.itcusat@gmail.com
Dr.M Sudheep Elayidom, Professor, Department of Computer Science, School of Engineering, Cochin University of Science and Technology, Cochin, Kerala, India. email: sudheep@cusat.ac.in
Dr.Sasi Gopalan, Associate Professor, Division of Applied Sciences and Humanities, School of Engineering, Cochin University of Science and Technology, Cochin, Kerala, India. email: sasigopalan@cusat.ac.in

II. RELATED WORK

There are several works done in this field that has vital influence and one of them is the work of the authors of [1]. They have proposed LoPD, a formal program semantics-based method which searches for any dissimilarity between two programs by finding an input that will cause these two programs to behave differently, either with different output states or with semantically different execution paths. Another work of paper [2] proposes a hybrid approach to plagiarism detection in academic documents that integrate detection methods using citations, semantic argument structure, and semantic word similarity with character-based methods to achieve a higher detection performance for disguised plagiarism forms.

Automatic correctness checking is introduced in the paper [3], which is searching for plagiarisms in assignments submitted and checking the correct implementation of algorithms. Another work is a code clone-detection algorithm for finding Type-3 clone i.e. similar methods with changes in statements. In this paper [4], they propose a threshold-free approach to detect Type-3 clones at method granularity across a large number of applications.

In paper [5], a Dynamic Key Instruction Sequence Based Software Birthmark (DKISB) is proposed that introduce dynamic data flow analysis into birthmark generation, which was able to produce a high-quality birthmark that is closely correlated to program semantics, making it resilient to various kinds of semantic-preserving code obfuscation techniques.

The work [6] explores the concept of imbrication and advances a conceptual scaffold for imbricating services. The aim of the paper [7] is writing style investigation using re-sampling approach. This paper presents the text as a series of characters generated by distinct probability sources. The paper [8] focuses on creating an effective and fast tool for plagiarism detection for text-based electronic assignments, AntiPlag. It is developed using the tri-gram sequence matching technique.

The authors of paper [9] tried to improve the existing techniques by separating the suspected files and the non-plagiarized files, thus reducing the dataset for further comparison. One to many comparisons of source code files using the code metrics is calculated for each file which reduces the dataset. The contents of the files are compared using Greedy String Tiling algorithm. And then suspected files are separated and performed string-matching to detect the level of similarity.

The proposed methodology in the paper [10] is N-gram similarity calculation method and Vector Space Model. Information Retrieval System and Cosine Normalization methods were utilized to calculate similarity ratios. By changing source code examples to different forms in the dataset, this methodology worked fine.

III. METHODOLOGY

A. Data Collection and Organizing

Fig.1 depicts the proposed methodology. Source Code Repository that the project uses to research covers 2 datasets collected from two different sources. Dataset- 1 consists of students' C program files collected from an institute laboratory. Dataset-2 consists of java program files downloaded from [26], which is considered as the benchmark dataset. This dataset specifies the plagiarized and non-plagiarized files that exist within the dataset.

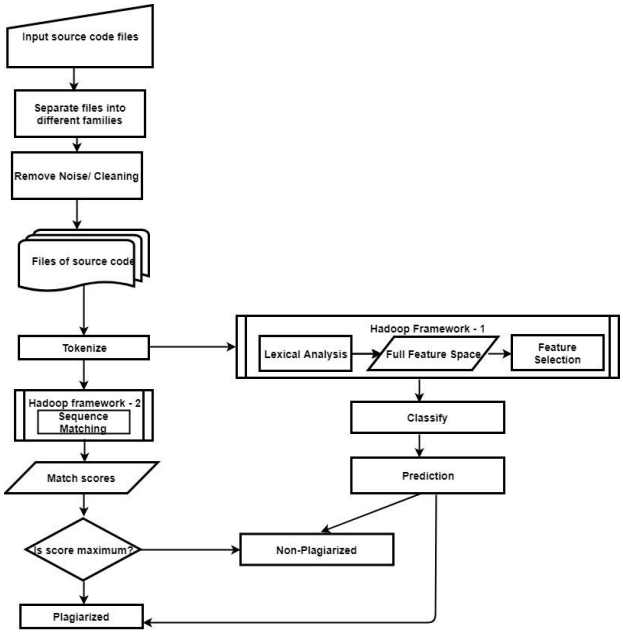

Fig. 1. Proposed Methodology

Dataset-1 was separated into 4 different classes (BST – binary search tree, CQ- circular queue, DQ – dequeue, MTX – matrix operations) by analyzing them manually. Same category program files are placed in a single folder thus obtaining four such categories.

B. Preprocessing

The preprocessing of Dataset involves cleaning of noise from the input files. The source code files of Dataset-1 contain noise like comment lines, sample outputs etc., typed in it other than the program lines. The Dataset-2 contains sample skeleton of the program given by the instructor for the assignment as noise in the source code files. These noises were cleaned and only files that consist of programs were considered for further analysis.

1) Tokenization: The source code files undergo tokenization which is the splitting of a stream of text into different pieces such as words, phrases, symbols, or other meaningful elements. These tokens become input for next level of processing. In this process, some characters like stop words, punctuation marks, white space or line breaks may or may not be discarded depending on the need. Here only white spaces and comment lines are discarded from every file.

C. Feature Extraction

Table I : Random Forest Classification on Dataset-1

CLASS	ACCURACY	TPR	FPR	ROC
BST	**80.7692%**	**0.808**	**0.115**	**0.853**
CQ	**96.1538%**	**0.962**	**0.090**	**0.994**
DQ	**88.4615%**	**0.885**	**0.295**	**0.892**
MTX	**89.4231%**	**0.894**	**0.292**	**0.873**

The features for the analysis of this dataset are taken from the files obtained after performing lexical analysis. So, only meaningful elements are involved in further analysis.

1) Lexical Analysis: The usage of Lexical Analysis in text mining is to study the word frequency distribution which makes sense here, as this helps to convert the text into data that is useful for analysis. Also in the text certain common words, library classes, header files etc. present in a source code are of less priority. So they are skipped and the tokens like keywords and functions that are of higher priority are identified using lexical analysis.

2) *Features*: Tokens like keywords, identifiers, basic data types, operators, delimiters, and functions that are predefined are considered as the features or attributes of these datasets.

D. Feature Selection using CPD

A feature Selection method CPD (Categorical Proportional Difference) is applied in order to select the features that are prominent in deriving the results. Also, this will help reduce the size of the feature length. CPD is chosen for feature selection as it measures the degree to which a token contributes in differentiating a specific class from other classes in the dataset. Here the values that would result after applying CPD is between -1 and +1, where -1 conveys that the token occurs approximately an equal number of times in files of all families or classes. But a value of +1 conveys that the token occurs only in files of one family or class.

$$CPD(F) = \max_k \{CPD(F, C_k)\} \quad (1)$$

$$CPD(F, C_k) = \frac{N_{F,C_k} - N_{F,\bar{C}_K}}{N_F} \quad (2)$$

1) Feature Occurrence Matrix: Feature Occurrence Matrix is generated using the attribute list and their term frequency in each file for the entire families with respect to one another. These matrices are used for further analysis.

E. Classification

Classification method used is Random Forest algorithm which is available in the WEKA tool. A 10-fold cross validation is done on the dataset. In the classification approaches the datasets have been passed onto WEKA as whole feature occurrence matrix and results are taken for analysis.

Table II : Random Forest Classification on Dataset-2

CLASS	ACCURACY	TPR	FPR	ROC
A1	92.5926%	0.958	0.333	0.972
A2	88.8889%	0.969	0.031	0.938

Table III : Classification Results After Applying CPD on Dataset-1

CLASS	ACCURACY	TPR	FPR	ROC
BST	**100%**	**1.000**	**0.000**	**1.000**
CQ	**85.5769%**	**0.500**	**0.026**	**0.855**
DQ	**89.4231%**	**0.85**	**0.00**	**0.891**
		9	**0**	
MTX	**98.0769%**	**0.923**	**0.000**	**0.958**

Also in another method, only the first token of each line in the files are considered for analysis. Thus the feature occurrence matrix generated for this method is classified and the results are obtained.

1) Tagging : Tagging is done by replacing each token with a category name based on the lexical files, which will further generalize the features or attribute list. By generalizing the attributes and then classifying its feature occurrence matrix, a better result is obtained using the classifier. So applying tagging conveys that generalizing the feature list will help to classify them easily. Because the variance between the term frequency values of each feature is very less, which may lead to miss-classification.

F. Sequence-Matching Similarity Check

After performing the lexical analysis, the resultant files are in the form of sequences. A sequence matching technique - the Needleman-Wunsch algorithm is applied in between the files within a class, to identify the files that have maximum match score. The files with maximum match score indicate that they are exactly similar. This method helps in not only identifying the plagiarized files but also in analyzing the quality of the model that is validated.

IV. EXPERIMENTS AND RESULTS

A. Classification Analysis

Both dataset-1 and dataset-2 undergo classification process to learn their behavior, how much refined and correctly classified they are. The classification method that was opted to learn them is Random Forest classifier. The WEKA data mining tool is used for the classification purposes. While classifying the dataset, cross-validation is chosen, as the size of the sample dataset is not considerable to perform a test and train process in classifying. But it is the right option when dealing with real big data.

The dataset-1 contains four families of programs and they are BST (Binary Search Tree class), CQ (Circular Queue class), DQ (DeQueue class) and MTX (Matrix Operations class) which are simple programs done by students. These four families contain 26 files each which are classified here to validate whether these files really belong to their respective classes.

The dataset-2 consists of two assignments folder A1 and A2 from SSID dataset which contains 2 set of programs CitiesDriver.java and Genealogy.java in each folder. Each class consists of 32 files. They are also classified for same analysis as before.

Table IV : Classification Results After Applying CPD on Dataset-2

CLASS	ACCURACY	TPR	FPR	ROC
A1	**92.5926%**	**0.958**	**0.333**	**0.972**
A2	**91.6667%**	**0.969**	**0.500**	**0.906**

Table V : Classification Results - Tagging on Dataset-1

CLASS	ACCURACY	TPR	FPR	ROC
BST	96.1538%	0.846	0.000	0.979
CQ	94.2308%	0.923	0.051	0.984
DQ	98.0769%	1.000	0.026	0.998
MTX	98.0769%	0.923	0.000	0.999

Table VI : Classification Results - Tagging on Dataset-2

CLASS	ACCURACY	TPR	FPR	ROC
A1	92.5926%	0.958	0.333	0.958
A2	97.222%	1.000	0.250	0.828

1) *Using Random Forest*: Random Forest is one of the best ensemble learning methods for classification which operates by constructing many decision trees and gives the classes of program family as output. Results of performing Random Forest classification on both the dataset is given in Table I and Table II respectively. This classification involves entire feature list with all their instances provided with respective class labels, where it is necessary for a supervised learning.

2) *Analysis after using CPD:* CPD feature selection is applied to the primary feature list and a more refined list of important features is obtained. Normally feature selection is done to reduce the size of the feature occurrence matrix and the same time not to neglect the features that are important for the analysis so that the time required for classifying could be reduced.

The reduced new attribute relation file when again classified using the random forest classifier shows an improvement in the results obtained. It is given in Table III and Table IV.

3) *Results after Tagging:* Results after tagging the features based on general categories like Keywords, Functions, Data Types, Operators, Delimiters, and Identifiers give a better outcome, which is shown in Table V and Table VI. The criterion for assigning tags is based on the semantic information. The total number of tag categories here is six.

Table VII : Classification Results - Tagging Only First Tokens on Dataset-1

CLASS	ACCURACY	TPR	FPR	ROC
BST	98.0769%	0.923	0.000	1.000
CQ	97.1154%	0.962	0.026	0.995
DQ	100%	1.000	0.000	1.000
MTX	99.0385%	0.962	0.000	1.000

Table VIII : Sequence Maximum Match Scores - Dataset-1

CLASS	SCORE	FILES
BST	11300	File1 to all
BST	11300	File2 to all
BST	11300	File3 to all
BST	11300	File4 to all
BST	11300	File5 to File6
BST	10300	File5 to File7
BST	10500	File5 to File8,9,10

This tagging process is similar to the sequence labeling in machine learning which involves the assignment of a categorical label to each member of a sequence.

4) Result after selecting First Tokens: Another method is considering only the first token of each line from a file. The experiment is done where the first tokens were tagged based on category and its results given in Table VII. This method of selecting feature list produced the best result compared to all other arranging methods.

B. Sequence Matching Results

The source code files contain programs in the form similar to sequence file format after doing all the preprocessing and tokenization. For performing string matching operation, the Needleman-Wunsch Sequence Matching algorithm is chosen. This will finally output a maximum match score which will help in deciding whether the file is same to the compared one or not. Files with tokens are given as input to the sequence matching Hadoop framework.

Also as an experiment, the tagged source code files are fed into the sequence matching Hadoop framework which gives maximum match values result that are similar to the values obtained for files with entire tokens. The results of sequence matching are given in Table VIII where the source code files after tagging are given as input to Hadoop Framework 2.

C. Prediction

The datasets results were compared with existing SSID results [26]. They have obtained 100% accuracy in their results for the benchmark dataset, whereas this methodology gives accuracy values close to it when compared to SSID for both dataset-1 and the benchmark dataset itself. The results of the analysis clearly convey that by generalizing the feature space depending on a categorical basis, we can produce better results than being specific with the feature list. So when dealing with a non-uniform and noisy big data corpus of source code files, it is highly suggested to generalize the feature space contents while using semantic information for processing, by applying techniques like tagging.

V. CONCLUSION AND FUTURE WORKS

Identification of plagiarized copies of program files in big data is the area in which analysis work is done. This experimental study aims to find a quality model through the usage of machine learning, which helps in generating a valid model from a large set of source code files. This work tries to incorporate machine learning with big data. The main idea is to process the semantic metrics of the source code to check for plagiarism and also to use sequence matching technique where both methods are guided by the previous works [8][14][24]. Classification of the attributes is performed using two methods to obtain a qualifying model that could really predict the plagiarized files from the original files. Using sequence matching technique ensures that the predicted results gained from classification are true.

From the analysis, it conveys that generalizing the attributes by tagging them based on category will produce more accurate results than being specific.

As a future work an efficient and fast plagiarism checking software for large source code files could be developed in Hadoop framework using this model.

REFERENCES

1. Fangfang Zhang, Dinghao Wu, Peng Liu, Sencun Zhu, "Program Logic Based Software Plagiarism Detection" in 2014 IEEE 25th International Symposium on Software Reliability Engineering, 3-6 Nov. 2014.

2. Norman Meuschke , Bele Gipp, "Reducing Computational Effort for Plagiarism Detection by using Citation Characteristics to Limit Retrieval Space", in IEEE/ACM Joint Conference on Digital Libraries.
3. Daniel Pohuba, Tom Dulk, Peter Jank "Automatic evaluation of correctness and originality of source codes", in 10th European Workshop on Microelectronics Education (EWME), 14-16 May 2014.
4. Iman Keivanloo, Feng Zang, Ying Zou, "Threshold-Free Code Clone Detection for a Large-Scale Heterogeneous Java Repository", 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER) 2-6 March 2015.
5. Zhenzhou Tian, Qinghua Zheng, Ting Liu, Ming Fan, "DKISB: Dynamic Key Instruction Sequence Birthmark for Software Plagiarism Detection", 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, 13-15 Nov. 2013.
6. George Ktistakis, Demosthenes Akoumianakis, Nik Bessis, "Sociomaterial Configurations of Human and NonHuman Actors: Re- Inventing Family Trip Planning Through Imbrication of Services", Science and Information Conference 2015 July 28-30, 2015 London, UK.
7. Oleg Granichin, Natalia Kizhaeva, Dmitry Shalymov, and Zeev Volkovich, "Writing Style Determination Using the KNN Text Model", 2015 IEEE International Symposium on Intelligent Control (ISIC) September 21-23, 2015. Sydney, Australia
8. MAC Jiffriya, MAC Akmal Jahan, Roshan G Ragel and Sampath Deegalla , "AntiPlag: Plagiarism Detection on Electronic Submissions of Text Based Assignments", 2013 IEEE 8th International Conference on Industrial and Information Systems, 17-20 Dec. 2013.
9. Omer Ajmal, M. M. Saad Missen, Tazeen Hashmat, M. Moosa, Tenvir Ali, "EPlag: A Two Layer Source Code Plagiarism Detection System", Eighth International Conference on Digital Information Management (ICDIM 2013),10-12 Sept. 2013.
10. Deniz Kln, Fatma Bozyiit, Alp Kut, Muhammet Kaya, "Overview of Source Code Plagiarism in Programming Courses", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-5 Issue-2, May 2015.
11. R. Dale, H. Mois, H. Somers, "Handbook of NLP", Marcel Dekker, 2000.
12. Stefan Janicke, Annette Gener, Marco Buchler, Gerik Scheuermann "Visualizations for Text Re-use", 014 International Conference on Information Visualization Theory and Applications (IVAPP), 5-8 Jan. 2014.
13. Patrick P.F. Chan, Lucas C.K. Hui and S.M. Yiu, "Heap Graph Based Software Theft Detection", IEEE Transactions On Information Forensics and Security, Vol.8, No.1, January 2013.
14. Georgina Cosma and Mike Joy, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis", IEEE Transactions on Computers, Vol. 61, No. 3, March 2012.
15. Chao Wang, Xi Li, Peng Chen, Aili Wang, Xuehai Zhou, and Hong Yu, "Heterogeneous Cloud Framework for Big Data Genome Sequencing", IEEE/ACM Transactions On Computational Biology and Bioinformatics, Vol. 12, No. 1, January/February 2015.
16. Yi Wang, Gagan Agarwal, Gulcin Ozer and Kun Huang, "Removing Sequential Bottlenecks in Analysis of Next–Generation Sequencing Data", 2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops.
17. Michael Tschuggnall and Gunther Specht, "Countering Plagiarism by Exposing Irregularities in Authors Grammars", 2013 IEEE European Intelligence and Security Informatics Conference, 12-14 Aug. 2013.
18. Wei Qu, Yuanyuan Jia and Michael Jiang, "Pattern Mining of Cloned Codes in Software Systems", ELSEVIER Information Sciences, 2014, Volume 259, Pages 544-554.
19. Andre S M. Bejarano, Lucy E. Garci A, Eduardo E. Zurek, "Detection of Source Code Similitude in Academic Environments", Journal-Computer Applications in Engineering Education archive, Volume 23 Issue 1, January 2015, Pages 13-22. Wiley Periodicals, Inc.
20. http://www.plagiarism.org
21. Eibe Frank, Machine Learning with WEKA, http://www.cs.waikato.ac.nz/ml/weka
22. Vidyullatha Pellakuri, Dr.D. Rajeswara Rao, "Hadoop Mapreduce Framework in Big Data Analytics", International Journal of Computer Trends and Technology (IJCTT) volume 8 number 3 Feb 2014.
23. Philip Russom, Big Data Analytics, Fourth Quarter 2011 TDWI Best Practices Report. https://tdwi.org/research/2011/09/~/media/ TDWI/TDWI/Research/BPR/2011/TDWI_BPReport_Q411_Big_Data_Analytics_Web/TDWI_BPReport_Q411_Big%20Data_ExecSummary.ashx
24. Saul B Needleman, Christian D Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Protein", J Mol. Biol.(1970) 48, 443-453.
25. Mondelle Simeon and Robert Hilderman, "Categorical Proportional Dierence: A Feature Selection Method for Text Categorization", Seventh Australasian Data Mining Conference 2008 Conferences in Research and Practice in Information Technology (CRPIT), Vol. 87.
26. Jonathan Y. H. Poon, Kazunari Sugiyama, Yee Fan Tan and Min-Yen Kan, "Instructor-Centric Source Code Plagiarism Detection and Plagiarism Corpus", ITiCSE12, July 35,2012 ACM.
27. http://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/SingleCluster.html

AUTHORS PROFILE

Ms. Viji Gopal completed her B-Tech from College of Engineering, Karunagappally and M.Tech from RMK Engineering College affiliated to Anna University. She registered as a Research Shcolar in the Department of Information Technology in School of Engineering in Cochin University of Science and Technology in the year 2015. She works as an Assistant Professor in the Department of Computer Science in SCMS School of Engineering and Technology, Karukutty, Cochin. She has 14 years of teaching experience. Her areas of interest are Networking, Database, Web mining, Big data, Cloud computing and Security. She has published papers in several national and international publications including Springer.

Dr.Varghese Paul completed his B.Sc (Engg) in Electrical Engineering from Kerala University, M.Tech in Electronics and Ph.D in Computer Science from Cochin University of Science and Technology. A few positions held by him are as Industrial Engineer with O/E/N India Ltd Cochin, Communication Engineer with KSE Board, Head of IT Department, CUSAT and Dean (CS, IT and Research) in Toc H Institute of Science and Technology. He is a Research Supervisor of CUSAT, MG University Kottayam, APJ Abdul Kalam Kerala Technological University etc. Research areas are Data Security using Cryptography, Data Compression, Data Mining, Image Processing and E_Governance. He developed TDMRC Coding System for character representation in computer systems and encryption system using this coding system.

Dr.M Sudheep Elayidom was a first rank holder in both B.Tech and M.Tech. He completed his PhD in Computer Science from Cochin University of Science and Technology. Currently he is working as a Professor and PhD guide in Department of Computer Science and Engineering in School of Engineering in CUSAT, Cochin. His areas of interest are data mining, cloud computing, big data, Database Management Systems(DBMS), Neural networks, Object oriented modelling and design and Software engineering. He has published a reference book by an international publisher Cengage learning, titled "Data mining and warehousing". He has done several international publications and working as a Ph.D guide in CUSAT, Kochi, India.

Dr. Sasi Gopalan works as Head of Division of Applied Sciences and Humanities, School of Engineering, CUSAT, Cochin. He possesses 16 years of teaching experience and is a guide for PhD programmes in CUSAT. He has published several papers in reputed journals. His areas of interest are Mathematics, Data mining, Linear Algebra, image processing and approximation theory.