

**ANALYSIS OF CLASSIFICATION TECHNIQUES IN RANSOMWARE  
DETECTION USING MACHINE LEARNING APPROACH**



**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

ANALYSIS OF CLASSIFICATION TECHNIQUES IN RANSOMWARE  
DETECTION USING MACHINE LEARNING APPROACH

MEIZA CERMELLA BT ABDUL AZIZ



This report is submitted in partial fulfillment of the requirements for the  
Bachelor of Computer Science (Computer Security) with Honours.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

## DECLARATION

I hereby declare that this project report entitled  
**ANALYSIS OF CLASSIFICATION TECHNIQUES IN RANSOMWARE  
DETECTION USING MACHINE LEARNING APPROACH**

is written by me and is my own effort and that no part has been plagiarized  
without citations.

STUDENT :  Date : 20/9/2023  
(MEIZA CERMELLA BT ABDUL AZIZ)



I hereby declare that I have read this project report and found  
this project report is sufficient in term of the scope and quality for the award of  
Bachelor of Computer Science (Computer Security) with Honours.

SUPERVISOR :  Date : 20/9/2023  
(PM TS. DR ROBIAH BINTI YUSOF)

## DEDICATION

I dedicate this work to my parents, Saniah Binti Husin and Abdul Aziz Bin Hj Bakar for their love, motivation and support in the aspect of emotional and financial support that they have provided me throughout this project and as long as I live. My parents have always believed in me, and their sacrifices have shaped me into the person I am today. This accomplishment illustrates your unwavering trust in me, and I will be forever grateful for your unconditional love and guidance.

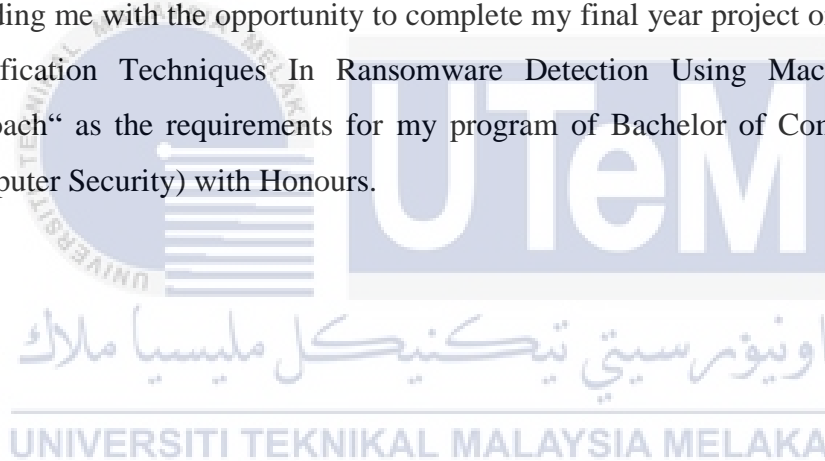
I also would like to dedicate this work to my inspiring supervisor Profesor Madya Ts. Dr Robiah Bt Yusof, my evaluator Ts. Haniza Bt. Nahar and especially all educators at the Universiti Teknikal Malaysia Melaka (UTeM) for their wisdom and guidance that has helped to shape my intellectual growth during my study. Additionally, I would like to dedicate this work to my precious sibling, relatives and my friends that have been with me providing motivational support and have supported me along the way. Your encouragement, advice, and prayers have meant the world to me.

In the name of Allah, the Most Gracious, the Most Merciful. I dedicate this Final Year Project to the God, the Almighty, for giving me the strength, wisdom, and guidance to complete this project. Without His divine intervention, I would not have been able to complete this Final Year Project. I am truly grateful for all the blessings that He has bestowed upon me.



## ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation to my supervisor Profesor Madya Ts. Dr Robiah Bt Yusof for her continuous support, advice, and patience to help me completing this project successfully. Her insights and expertise have helped me to develop a strong understanding of the topic and she always motivates me to explore new understanding of certain areas of topic that I'm not familiar with. I'm grateful and honored to be under her supervision and I enjoy having meetings and discussions with my supervisor as she always provides her best in making me understand the discussions topic. In addition to that, I would like to express my sincere gratitude to the Universiti Teknikal Malaysia Melaka (UTeM) for providing me with the opportunity to complete my final year project on "Analysis Of Classification Techniques In Ransomware Detection Using Machine Learning Approach" as the requirements for my program of Bachelor of Computer Science (Computer Security) with Honours.



## ABSTRACT

Ransomware is one of the most devastating cyberattacks in the malware category which involves the victim device being locked from accessing the system. The increase of ransomware attacks may be caused by several factors such as insufficient corporate security defense and the trends of ransomware as a service known as (RaaS) affiliate market. Additionally, most of the antivirus that use signature-based detection can be ineffective especially for detecting new variants of ransomware. There's also a challenge in selecting appropriate classification techniques due to the extensive scientific and technical materials involved. Therefore, taking all these problems into consideration this project objective is to evaluate the performance of various classification techniques for detection and classification of ransomware. The research methodology involves acquiring a comprehensive ransomware dataset from reputable sources such as Kaggle, UCI Machine Learning Repositories, and Resilient Information Systems Security (RISS) Ransomware Dataset. The dataset undergoes preprocessing steps, including data cleaning to handle missing values and noisy data. Feature selection methods are applied to identify the most informative features, thereby enhancing the accuracy of the ransomware detection system. Several machine learning classifiers, including Decision Tree, Random Forest, Support Vector Machines (SVM), and Naïve Bayes, are employed for training the ransomware detection model. The resulting models are then evaluated using various evaluation metrics such as accuracy, precision, recall, F-measure, and True Positive Rate (TPR) and False Positive Rates (FPR). The outcomes of this study contribute to the understanding of the performance of different classification techniques in the context of ransomware detection. The findings illustrate that performance consistently improves with larger balanced dataset sizes, notably Random Forest highest being 99.30% accuracy, exhibit remarkable accuracy gains when transitioning from imbalanced to balanced datasets. Future research directions include exploring deep learning methods, utilizing larger datasets, and conducting real-time testing to further enhance the accuracy and zero-day attack of ransomware detection systems. This research can serve as a reference for future work to combat the rising threat of ransomware attacks.

## ABSTRAK

“Ransomware” adalah salah satu serangan siber yang paling dahsyat dalam kategori perisian hasad yang melibatkan mangsa terkunci dari mengakses peranti. Peningkatan serangan “ransomware” disebabkan oleh beberapa faktor seperti pertahanan keselamatan korporat yang tidak mencukupi dan pola “ransomware” sebagai perkhidmatan yang dikenali sebagai (RaaS). Selain itu, kebanyakan antivirus yang tidak dapat mengesan varian “ransomware” baru. Terdapat juga cabaran dalam memilih teknik klasifikasi yang sesuai kerana banyak bahan saintifik dan teknikal yang terlibat. Oleh itu, dengan mempertimbangkan semua masalah ini, projek ini bertujuan untuk menilai prestasi pelbagai teknik klasifikasi untuk pengesanan dan klasifikasi “ransomware”. Metodologi penyelidikan melibatkan memperoleh set data “ransomware” yang komprehensif dari sumber terkenal seperti Kaggle, Repositori Pembelajaran Mesin UCI, dan Keselamatan Sistem Maklumat Berdaya Tahan (RISS) Ransomware Dataset. Set data menjalani langkah-langkah pra-pemprosesan, termasuk pembersihan data untuk menangani nilai yang hilang. Kaedah pemilihan ciri digunakan untuk mengenal pasti ciri yang paling bermaklumat, sehingga meningkatkan ketepatan sistem pengesanan “ransomware”. Beberapa mesin belajar klasifikasi, termasuk Decision Tree, Random Forest, Support Vector Machines (SVM), dan Naïve Bayes, digunakan untuk melatih model pengesanan ransomware. Model yang dihasilkan kemudian dinilai menggunakan pelbagai metrik penilaian seperti ketepatan, ketepatan, penarikan balik, ukuran F, dan Kadar Positif Sejati (TPR) dan Kadar Positif Palsu (FPR). Keputusan kajian ini menggambarkan bahawa prestasi konsisten bertambah baik dengan saiz set data yang lebih besar, terutamanya Random Forest tertinggi ketepatan 99.30%, membuktikan kecekapan bagus apabila beralih daripada set data tidak seimbang kepada seimbang. Arah penyelidikan masa depan termasuk meneroka kaedah lain, menggunakan set data yang lebih besar, dan menjalankan ujian masa nyata untuk meningkatkan lagi ketepatan dan serangan sifar hari sistem pengesanan perisian tebusan. Penyelidikan ini boleh menjadi rujukan untuk kerja masa depan dalam bidang pengesanan “ransomware” yang semakin meningkat.

## TABLE OF CONTENTS

	<b>PAGE</b>
<b>DECLARATION.....</b>	<b>II</b>
<b>DEDICATION.....</b>	<b>III</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>IV</b>
<b>ABSTRACT .....</b>	<b>V</b>
<b>ABSTRAK .....</b>	<b>VI</b>
<b>TABLE OF CONTENTS.....</b>	<b>VII</b>
<b>LIST OF TABLES .....</b>	<b>XIV</b>
<b>LIST OF FIGURES .....</b>	<b>XVI</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>XX</b>
<b>LIST OF ATTACHMENTS.....</b>	<b>XXI</b>
<b>CHAPTER 1: INTRODUCTION .....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Background.....	2
1.3 Problem Statement.....	3
1.4 Project Question.....	3
1.5 Project Objective.....	4
1.6 Project Scope .....	4
1.7 Project Contribution.....	5

1.8	Report Organization.....	6
1.9	Summary.....	7
<b>CHAPTER 2: LITERATURE REVIEW.....</b>		<b>8</b>
2.0	Introduction.....	8
2.1	Ransomware.....	10
2.1.1	Definition.....	10
2.1.2	Categories .....	12
2.1.2.1	Locker.....	12
2.1.2.2	Crypto .....	13
2.1.2.3	Scareware.....	14
2.1.2.4	Leakware.....	14
2.1.2.5	RaaS.....	14
2.1.3	Lifecycle .....	15
2.1.4	Type of Attacks.....	18
2.1.4.1	Exploitable Software Vulnerabilities.....	18
2.1.4.2	Brute-Force Credential attacks .....	18
2.1.4.3	Phishing emails.....	18
2.1.4.4	Remote Desktop Protocol Attack .....	18
2.2	Machine Learning .....	19
2.2.1	Definition.....	19
2.2.2	Types.....	20
2.2.2.1	Supervised.....	20
2.2.2.2	Unsupervised .....	21

2.2.2.3	Semi-supervised.....	21
2.2.2.4	Summary of type of machine learning.....	22
2.2.3	Tools .....	22
2.2.3.1	Weka.....	22
2.2.3.2	Orange.....	23
2.2.3.3	TensorFlow .....	23
2.2.3.4	Azure.....	23
2.2.3.5	Scikit-Learn .....	24
2.2.4	Summary of Machine Learning tools feature .....	25
2.3	Techniques.....	27
2.3.1	Definition.....	27
2.3.2	Classification Techniques.....	27
2.3.2.1	Decision Tree.....	28
2.3.2.2	Random Forest.....	29
2.3.2.3	Support Vector Machine.....	30
2.3.2.4	Naïve Bayes .....	31
2.3.2.5	Summary of machine learning algorithms.....	32
2.3.3	Analysis.....	35
2.3.3.1	Datasets .....	35
2.3.3.2	Parameters.....	37
2.3.3.3	Train and Test Ratio.....	41
2.3.3.4	Evaluation Techniques.....	42
2.4	Critical Review.....	48

2.5 Proposed solution .....	52
2.6 Summary .....	52
<b>CHAPTER 3: METHODOLOGY .....</b>	<b>53</b>
3.0 Introduction.....	53
3.1 Project Methodology.....	54
3.1.1 Phase I: Identify and Gather Project Requirements.....	55
3.1.2 Phase II: Select Tools and Datasets .....	55
3.1.3 Phase III: Installing the Tools.....	56
3.1.4 Phase IV: Dataset Preparation .....	56
3.1.5 Phase V: Information Collection .....	57
3.1.6 Phase VI: Analyze the Information .....	57
3.1.7 Phase VII: Document Result .....	58
3.2 Research Milestone.....	58
3.3 Research Gantt Chart .....	59
3.4 Summary .....	60
<b>CHAPTER 4: ANALYSIS AND DESIGN.....</b>	<b>61</b>
4.0 Introduction.....	61
4.1 Research Workflow .....	62
4.2 Project Requirements Analysis .....	64
4.2.1 Hardware Requirements .....	64
4.2.2 Software Requirements.....	64
4.3 Architecture Analysis.....	66
4.4 Proposed Research Design.....	67
4.5 Flowchart Design of Research .....	69

4.6	Tools interface .....	71
4.8	Summary .....	72
<b>CHAPTER 5: IMPLEMENTATION.....</b>		<b>73</b>
5.0	Introduction.....	73
5.1	Research Implementation Activities .....	74
5.2	Step 1: Environment Setup based on Requirements .....	75
5.2.1	Hardware.....	75
5.2.2	Operating System.....	76
5.2.3	Software .....	76
5.3	Step 2: Installation and Configurations Machine Learning Setup .....	78
5.3.1	Installation and Configurations WEKA.....	78
5.4	Step 3: Acquiring Datasets.....	81
5.5	Step 4: Pre-Processing .....	82
5.5.1	Steps for Pre-Processing .....	82
5.5.2	Comparison Before and After.....	84
5.6	Step 5: Load Dataset .....	85
5.6.1	Steps to Load Dataset for WEKA.....	85
5.6.2	Steps to Load Dataset for ORANGE.....	87
5.7	Step 6: Classification .....	88
5.7.1	Steps to classify data in WEKA.....	89
5.7.2	Steps to classify data in ORANGE.....	90
5.8	Step 7: Generate Result.....	91
5.8.1	Step 7: Generate Result for WEKA.....	91
5.8.2	Step 7: Generate Result for ORANGE .....	92



5.9	Conclusion .....	93
<b>CHAPTER 6: TESTING AND EVALUATION .....</b>		<b>94</b>
6.0	Introduction.....	94
6.1	Testing.....	95
6.1.1	Test for various dataset size and unbalanced vs balanced dataset.	97
6.1.2	Test for various ratio of training and test <b>Error! Bookmark not defined.</b>	
6.2	Result and Analysis Dataset I .....	99
6.2.1	Evaluation Metric Result of 1000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Ransomware Detection ...	99
6.2.2	Evaluation Metric Result of 5000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Dataset Ransomware Detection.....	102
6.2.3	Evaluation Metric Result of 10 000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Dataset Ransomware Detection.....	105
6.2.4	Accuracy of Classification Model Across Different Sample Sizes ..	108
6.2.5	Accuracy of Classification Model for Different Ratio .....	109
6.2.6	Comparison between WEKA and Orange.....	112
6.3	Result and Analysis Dataset II.....	116
6.3.1	Evaluation Metrics Result of 1000 Android Samples for Unbalanced and Balanced Ransomware Detection .....	116
6.3.2	Evaluation Metric Result of 5000 Android Samples for Unbalanced and Balanced Dataset Ransomware Detection .....	119
6.3.3	Evaluation Metric Result of 10 000 Android Samples for Unbalanced and Balanced Dataset Ransomware Detection .....	122
6.3.4	Accuracy of Classification Model Across Different Sample Sizes ..	125
6.3.5	Accuracy of Classification Model for Different Ratio .....	126
6.3.6	Comparison between WEKA and Orange.....	129
6.4	Result and Analysis Dataset III.....	133

6.4.1	Evaluation Metrics Result of 1000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection.....	133
6.4.2	Evaluation Metrics Result of 5000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection.....	137
6.4.3	Evaluation Metrics Result of 10 000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection.....	140
6.4.4	Accuracy of Classification Model Across Different Sample Sizes ..	143
6.4.5	Accuracy of Classification Model for Different Ratio .....	144
6.4.6	Comparison between WEKA and Orange.....	147
6.5	Tools Comparison.....	151
6.6	Significant Results .....	156
6.7	Summary.....	159
<b>CHAPTER 7: CONCLUSION.....</b>		<b>160</b>
7.0	Introduction.....	160
7.1	Project Summarization.....	161
7.2	Project Contributions .....	162
7.3	Project Limitations.....	162
7.4	Future Work.....	163
7.5	Summary.....	164
<b>REFERENCES.....</b>		<b>165</b>
<b>APPENDIX A .....</b>		<b>172</b>
<b>APPENDIX B .....</b>		<b>175</b>
<b>APPENDIX C .....</b>		<b>182</b>
<b>APPENDIX D .....</b>		<b>184</b>

## LIST OF TABLES

	<b>PAGE</b>
Table 1.1 Summary of Problem Statement .....	3
Table 1.2 Summary of Project Question .....	3
Table 1.3 Summary of Project Objective .....	4
Table 1.4 Summary of Project Contribution .....	5
Table 2.1 Summarization of each Machine Learning tools feature based on reviewed literatures.....	25
Table 2.2 Summarization Machine Learning algorithms based on previous study ...	32
Table 2.3 indicator to represent the machine learning algorithms .....	33
Table 2.4 Summarization of each Machine Learning algorithms based on the tools	34
Table 2.5 Ransomware families of dataset 1 mapped to its category .....	36
Table 2.6 Ransomware of families of dataset 2 mapped to its category .....	36
Table 2.7 Ransomware of families of dataset 3 mapped to its category .....	38
Table 2.8 Parameters used in Kaggle Dataset Android Ransomware Detection .....	39
Table 2.9 Parameters used in Resilient Information Systems Security (RISS) Ransomware Dataset.....	40
Table 2.10 Parameters used in UCI BitcoinHeistRansomwareAddressDataset Data Set .....	40
Table 2.11 Summary review for evaluation metrics based on reviewed literatures ..	46
Table 2.12 Indicator to represent the evaluation metrics .....	47
Table 2.13 Summary of critical review for previous research articles .....	48
Table 3.1 Summary of Research Milestone .....	58
Table 3.2 Research Gantt Chart .....	9
Table 4.1 Summary of hardware specifications.....	64
Table 4.2 Summary of software specifications .....	65
	<b>PAGE</b>
Table 6.1 Evaluation Metrics Result of 1000 BitcoinHeist Samples.....	101

Table 6.2 Evaluation Metrics Result of 5000 BitcoinHeist Samples.....	104
Table 6.3 Evaluation Metrics Result of 10 000 BitcoinHeist Samples.....	107
Table 6.4 Summary of results Dataset I for TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA.....	114
Table 6.5 Summary of results Dataset I for TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE.....	116
Table 6.6 Evaluation Metrics Result of 1000 Android Samples.....	118
Table 6.7 Evaluation Metrics Result of 5000 Android Samples.....	121
Table 6.8 Evaluation Metrics Result of 10 000 Android Samples.....	124
Table 6.9 Summary of results Dataset II for TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA (70:30).....	131
Table 6.10 Summary of results Dataset II for TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE (70:30).....	133
Table 6.11 Evaluation Metrics Result of 1000 File System Behavior Ransomware dataset.....	135
Table 6.12 Evaluation Metrics Result of 5000 File System Behavior Ransomware dataset.....	138
Table 6.13 Evaluation Metrics Result of 10 000 File System Behavior Ransomware dataset.....	141
Table 6.14 Summary of results Dataset III TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA (70:30).....	148
Table 6.15 Summary of results Dataset III for TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE (70:30).....	150
Table 6.16 Classifier performance comparison between WEKA and Orange (10 000 samples ,10:90).....	152
Table 6.17 Classifier performance comparison between WEKA and Orange (10 000 samples ,30:70).....	153
Table 6.18 Classifier performance comparison between WEKA and Orange (10 000 samples ,50:50).....	154
Table 6.19 Classifier performance comparison between WEKA and Orange (10 000 samples ,70:30).....	155
Table 6.20 Classifier performance comparison between WEKA and Orange (10 000 samples ,70:30).....	156
Table 6.21 Significant Results for comparison of previous research.....	159

## LIST OF FIGURES

	PAGE
Figure 2.1 Research Framework .....	9
Figure 2.2 Known ransomware attacks by gang, March 2023 (Malwarebytes, 2023) .....	11
Figure 2.3 Ransom note display at the infected device by the Locker (Avast, 2022)	12
Figure 2.4 Payment method used for retrieving the private key due to the CryptoLocker (Avast, 2022).....	13
Figure 2.5 Infection phases of ransomware lifecycle (Abbasi, 2023).....	15
Figure 2.6 Categories of machine learning (Naqa et al., 2015) .....	20
Figure 2.7 Example of decision tree. (SecurityExperts.it).....	28
Figure 2.8 Example of Random Forest (jvatpoint).....	29
Figure 2.9 Example of Support Vector Machine (jvatpoint).....	30
Figure 2.10 Example of Naïve Bayes (Chaudhuri, 2022).....	31
Figure 2.11 Accuracy Formula (Kok et al., 2020).....	42
Figure 2.12 True Positive Rate (TPR) Formula (Kok et al., 2020).....	43
Figure 2.13 False Positive Rate (FPR) Formula (Kok et al., 2020).....	43
Figure 2.14 True Negative Rate (TPR) Formula (Kok et al., 2020) .....	43
Figure 2.15 False Negative Rate (FNR) Formula (Kok et al., 2020).....	44
Figure 2.16 Precision Formula (Kok et al., 2020).....	44
Figure 2.17 Recall Formula (Kok et al., 2020) .....	45
Figure 2.18 F-measure Formula (Kok et al., 2020).....	45
Figure 3.1 Project methodology .....	54
Figure 3.2 Process to Identify and gather project requirement .....	55
Figure 3.3 Process of selecting tools and dataset.....	55
Figure 3.4 Process of installing the selected machine learning tools.....	56
Figure 3.5 Process of Dataset Preparation .....	56

Figure 3.6 Process of Information collections .....	57
Figure 3.7 Process of analyze the information.....	57
Figure 3.8 Process of document result .....	58
Figure 4.1 Workflow for the preparation of dataset for Phase III of the research.....	62
Figure 4.2 Dataset “Android Ransomware” from Kaggle .....	63
Figure 4.3 Workflow of the training and testing.....	63
Figure 4.4 Overview of architecture analysis of this project .....	66
Figure 4.5 Proposed research design for ransomware detection model.....	67
Figure 4.6 Flowchart of WEKA to visualize possible scenario that might occur.....	69
Figure 4.7 WEKA GUI chooser.....	71
Figure 4.8 Example of Result when Load the Dataset I Andoroid Ransomware to the WEKA.....	71
Figure 4.9 Main user interface of Orange .....	72
Figure 5.1 Diagram Outlining Research Implementation Activities .....	74
Figure 5.2 hardware specifications .....	75
Figure 5.3 Operating System specifications.....	76
Figure 5.4 WEKA official website.....	76
Figure 5.5 Orange official website.....	77
Figure 5.6 Setup Wizard of WEKA .....	78
Figure 5.7 WEKA License Agreement.....	79
Figure 5.8 WEKA Associate Package Files.....	79
Figure 5.9 WEKA Installation Location .....	80
Figure 5.10 WEKA interface .....	80
Figure 5.11 Android Ransomware Detection dataset .....	81
Figure 5.12 BitcoinHeistRansomwareAddress Dataset .....	81
Figure 5.13 selecting the dataset in csv file format.....	82
Figure 5.14 Categorization of dataset attributes .....	82
Figure 5.15 Categorization of dataset attributes .....	83
Figure 5.16 Categorization of dataset attributes .....	83
Figure 5.17 Dataset 1 before Pre-Processing .....	84
Figure 5.18 Dataset 1 with defined relation and attributes .....	84
Figure 5.19 Dataset 1 with defined data.....	85
Figure 5.20 Weka Interface.....	85
Figure 5.21 Loading the .arff file format dataset into WEKA .....	86

Figure 5.22 graph for visualize the Ransomware and Benign samples .....	86
Figure 5.23 Loading dataset into Orange .....	87
Figure 5.24 Data Table representation in Orange .....	87
Figure 5.25 Assigning target class in Orange .....	88
Figure 5.26 WEKA explorer .....	89
Figure 5.27 Selection of classification algorithms in WEKA.....	89
Figure 5.28 Selection of classification algorithms in Orange .....	90
Figure 5.29 Connecting each nodes in Orange .....	90
Figure 5.30 Dataset splitting in WEKA .....	91
Figure 5.31 Evaluation Metrics result for Decision Tree in WEKA.....	92
Figure 5.32 Dataset splitting in Orange .....	92
Figure 5.33 Completion percentage of Test and Score process .....	93
Figure 5.34 Evaluation Metrics result for Decision Tree in Orange.....	93
Figure 6.1 Test Plan Strategy .....	97
Figure 6.2 Flowchart Conducting Test Plan .....	98
Figure 6.3 Command for Sub-sampling the Ransomware dataset using Python language in Jupyter Notebook.....	99
Figure 6.4 Command to show the Ransomware dataset dimentsion before and after sub-sampling method .....	100
Figure 6.5 Graph Accuracy for 1000 BitcoinHeist Samples.....	103
Figure 6.6 Graph Accuracy for 5000 BitcoinHeist Samples.....	106
Figure 6.7 Graph Accuracy for 10 000 BitcoinHeist Samples.....	109
Figure 6.8 Graph Accuracy for Classification Model Across Different Sample Sizes of BitcoinHeist Ransomware.....	110
Figure 6.9 Graph Accuracy for 1000 BitcoinHeist Samples across Different Ratio .....	111
Figure 6.10 Graph Accuracy for 5000 BitcoinHeist Samples across Different Ratio .....	112
Figure 6.11 Graph Accuracy for 10 000 BitcoinHeist Samples across Different Ratio .....	113
Figure 6.12 Graph Accuracy for 1000 Android Samples .....	120
Figure 6.13 Graph Accuracy for 5000 Android Samples .....	123
Figure 6.14 Graph Accuracy for Classification Model Across Different Sample Sizes of FileSystem Behavior Ransomware.....	141

Figure 6.15 Graph Accuracy for Classification Model Across Different Sample Sizes of Android Ransomware .....	127
Figure 6.16 Graph Accuracy for 1000 Android Samples across Different Ratio ....	128
Figure 6.17 Graph Accuracy for 5000 Android Samples across Different Ratio ....	129
Figure 6.18 Graph Accuracy for 10 000 Android Samples across Different Ratio .	130
Figure 6.19 Graph Accuracy for 1000 File System Behavior Ransomware .....	Dataset 137
Figure 6.20 Graph Accuracy for 5000 File System Behavior Ransomware .....	Dataset 140
Figure 6.21 Graph Accuracy for 10 000 File System Behavior Ransomware dataset	143
Figure 6.22 Graph Accuracy for Classification Model Across Different Sample Sizes of Filesystem Behavior Ransomware.....	144
Figure 6.23 Graph Accuracy for 1000 File System Behavior Ransomware Dataset across Different Ratio .....	145
Figure 6.24 Graph Accuracy for 5000 File System Behavior Ransomware Dataset across Different Ratio .....	146
Figure 6.25 Graph Accuracy for 10 000 File System Behavior Ransomware Dataset across Different Ratio 1 .....	47
Figure 6.26 Overall Test Plan Findings .....	157



## LIST OF ABBREVIATIONS

<b>FYP</b>	-	<b>Final Year Project</b>
Acc		Accuracy
TPR		True Positive Rate (also known as Sensitivity, Recall, or Hit Rate)
FPR		False Positive Rate
TNR		True Negative Rate (also known as Specificity)
FNR		False Negative Rate
Prec		Precision
Recall		Recall
F-m		F-measure (also known as F1 Score)
AUC		Area Under the ROC Curve
MCC		Matthews Correlation Coefficient
PLR		Positive Likelihood Ratio
NLR		Negative likelihood ratio
DOR		Diagnostic odds ratio
J		Youden's index
NND		Number needed to diagnose
NNM		Number needed to misdiagnose
NB		Net benefit

## LIST OF ATTACHMENTS

		<b>PAGE</b>
<b>Appendix A</b>	<b>Details of critical review</b>	<b>163</b>
<b>Appendix B</b>	<b>Implementation classification models</b>	<b>166</b>
<b>Appendix C</b>	<b>Sample of Python code for Sub-sampling</b>	<b>173</b>
<b>Appendix D</b>	<b>Results Generated for WEKA and Orange</b>	<b>175</b>



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## CHAPTER 1: INTRODUCTION

### 1.1 Introduction

Ransomware is one of the types of malicious software also known as malware. It's one of the most devastating cyberattacks in the malware category which involves the victim device being locked from accessing the system (Fedor, 2023). According to (Madani et al., 2023) ransomware has the ability to infect devices through multiple file format such as .exe, .docx, .ppt and etc. Other issues related to ransomware includes, it can spread via social engineering using email attachments and compromised websites links to attack individual and organizations (Mohammad, 2020). In addition, another issue concerns the financial aspect. According to the Cybersecurity venture predict, the cost of ransomware will be over \$42 billion by the end of 2024 and over \$265 billion by 2031 due to the rising number of cases (Fedor, 2022). The increase of ransomware attacks is caused by several factors such as insufficient corporate security defense and trends of RaaS (Ransomware-as-a-Service) affiliate market. Ransomware-as-a-Service is a business model developed by the ransomware creators which distributes and sells it to other cybercriminals (Kaspersky, n.d). The subscribers who take part in the Ransomware-as-a-Service are known as affiliates or users. The model is derived from the concept such as SaaS (Software-as-a-Service) but in this case it is being used for malicious purposes. The trades of ransomware can be found especially on the dark web and one of the famous cases involves the LockBit ransomware. (Antal, 2023) stated that Lockbit is the number one leading for Ransomware attacks by gang in January 2023. Other deadliest ransomware groups such as Conti, REvil, DarkSide and Doppelpaymer (Antal, 2023).

## 1.2 Background

One of the most significant cases includes President of Costa Rica, Rodrigo Chaves Robles that announced national state of emergency due to continuing ransomware attack in May 2022 by the Russian group Conti gang (Security Intelligence, 2023). Most antivirus that use signature-based detection can be ineffective especially for detecting new variants of ransomware that can evade detection using techniques such as polymorphism and code obfuscation. According to (Gagalic et al., 2023) there has been proposed research for Ransomware detection with Machine Learning in Storage Systems using the Random Forest classification technique with 97.3% of F1-Score. Besides, (Horduna et al., 2023) has stated that machine learning can be implemented to detect and predict ransomware attack behavior, in order to counter the current issues regarding ransomware. Other than that, according to (Khalil et al., 2022) there are many machine learning algorithms and classification techniques for detection of ransomware that have been proposed.

However, there's still a lack of knowledge on implementing various classification techniques for analyzing ransomware. According to (Smith et al., 2022) due to large scientific and technical resources, it's challenging to identify suitable machine learning classification techniques for analyzing ransomware. Therefore, taking all these problems into consideration this project proposed research to study more about various classification techniques that can be used to analyze ransomware using machine learning approach. Other than that, this project will also apply the classification techniques to ransomware dataset and evaluate the accuracy result of classification techniques using different evaluation metrics tool. The datasets can be obtained from official dataset repositories such as Kaggle (Chakraborty, 2023), UCI Machine Learning Repositories (Sgandurra at al., 2016), Resilient Information Systems Security (RISS) Ransomware Dataset which are collected and analyzed from the Cuckoo sandbox and others.

### 1.3 Problem Statement

Ransomware attacks are continually evolving which involve new groups of threat actors to grow and ransomware malware being developed. There is a need for improvement of previous research in accordance with ransomware attacks being one of the most significant issues in cybersecurity.

*Table 1.1 Summary of Problem Statement*

PS	Problem Statement
PS1	It's challenging to identify suitable machine learning classification techniques for analyzing Ransomware due to large scientific and technical materials being used.

Table 1.1 shows the problem statements in this project. Several studies (Khalil et al., 2022) have proposed numerous machine learning algorithms and classification techniques for ransomware detection. However, there remains a knowledge gap regarding the practical implementation of different classification techniques for analyzing ransomware. Additionally, the extensive scientific and technical resources available make it challenging to identify appropriate machine learning classification techniques for ransomware analysis, as highlighted by (Smith et al., 2022).

### 1.4 Project Question

*Table 1.2 Summary of Project Question*

PS	PQ	Project Question
PS1	PQ1	What are the different classification techniques in Machine Learning that can be used to analyze ransomware?
	PQ2	How can the classification techniques in Machine Learning be applied to ransomware datasets to accurately identify ransomware attacks?
	PQ3	How to identify the best classification techniques in detecting Ransomware.

Table 1.2 shows the project question that's derived based on the problem statement in Table 1.1. Project question 1 focuses on identifying different classification techniques in Machine Learning that can be used to analyze ransomware. Project question 2 focuses on the technique to apply the ransomware dataset to identify and categorize the ransomware attack. The last project question is to identify the best classification techniques in ransomware detection.

## 1.5 Project Objective

Based on the project questions above, Table 1.3 shows the summary of project objective that's constructed based on each of the project questions.

*Table 1.3 Summary of Project Objective*

PS	PQ	PO	Project Objective
PS1	PQ1	PO1	To <b>study the classification techniques</b> for analyzing Ransomware
	PQ2	PO2	To <b>apply the classification techniques</b> to Ransomware dataset.
	PQ3	PO3	To <b>evaluate the accuracy result</b> of classification techniques using different evaluation metrics tool.

## 1.6 Project Scope

**The scope of this project is listed down as below:**

- 1) Data collection involves datasets from various data repositories that will be used in this project.

- **Dataset 1 Ransomware: UCI**

**BitcoinHeistRansomwareAddress Data Set**

<https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset#>

- **Dataset 2 Ransomware: Kaggle Dataset Android Ransomware Detection**  
<https://doi.org/10.34740/KAGGLE/DSV/4987535>
- **Dataset 3 Ransomware: Kaggle Dataset Ransomware Detection File System Behavior**  
<https://www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set?resource=download>

- 2) This project will implement various classification techniques in machine learning such as Decision Tree, Random Forest, Support Vector Machine and Naïve Bayes to identify the best technique for detection of Ransomware.
- 3) Machine Learning tools that will be used in this project are Weka and Orange.
- 4) This project will evaluate the accuracy using evaluation metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Recall, Precision, Accuracy and F-measure score will be assessed.

### 1.7 Project Contribution

Table 1.4 shows the project contribution that's mapped based on the project objectives, project question and the problem statement.

*Table 1.4 Summary of Project Contribution*

PS	PQ	PO	PC	Project Contribution
PS1	PQ1	PO1	PC1	Taxonomy of Ransomware and its classification techniques.
	PQ2	PO2	PC2	Proposed the best classification technique to be applied for Ransomware dataset.
	PQ3	PO3	PC3	Verified the best classification technique based on the evaluation of the accuracy result.

## **1.8 Report Organization**

This section illustrates the organization of this report, which includes a total of six chapters.

### **Chapter 1: Introduction**

The first chapter introduces the background of ransomware and its problem statements based on current issues. This chapter also outlines the research project questions, project objectives, the scope, project contribution, project organization and conclusion as the summarization of chapter 1.

### **Chapter 2: Literature Review**

The second chapter will include analysis of previous study regarding ransomware. This chapter will provide reviewed of research papers to study the approaches used in machine learning for identifying ransomware attacks.

### **Chapter 3: Methodology**

The third chapter explains the approaches/methodology that will be used to carry out the research as justified in previous chapter. This chapter will also outline the project schedule and milestone to ensure completion of the project activities are according to the timeline.

### **Chapter 4: Analysis and Design**

The fourth chapter will focus on the workflow of the research. This chapter will also illustrate the architecture design that will be used for the next chapter.

### **Chapter 5: Implementation**

The fifth chapter will focus on the implementation of the project. This chapter will apply the classification techniques in machine learning to the Ransomware dataset using tools such as Weka and Orange.



## **Chapter 6: Testing and Evaluation**

The sixth chapter will discuss the testing and evaluation of the project. This chapter will analyze and evaluate the accuracy result of classification techniques using different evaluation metrics tools.

## **Chapter 7: Conclusion**

This chapter will include the project summarization, the contribution, project limitation and improvement for future work.

### **1.9 Summary**

In conclusion, this project aims to have a better understanding in implementing various classification techniques for analyzing Ransomware. A comparison will be made to propose which classification techniques is the best to be applied for Ransomware dataset. In addition, this project will also verify the best classification technique based on the evaluation of the accuracy result. This chapter includes background of Ransomware, problem statement, project questions, project objective, project scope, project contribution and the report organization. In the next chapter, review of various research papers will be analyzed to study the approaches used in machine learning for identifying ransomware attacks.

## CHAPTER 2: LITERATURE REVIEW

### 2.0 Introduction

The second chapter will include analysis of previous study regarding ransomware. This chapter will provide reviewed of research papers to study the approaches used in machine learning for identifying ransomware attacks. In addition to that, the objective is to provide an overview of existing research and knowledge in relation to the research. Indirectly it helps in identifying existing research restrictions, such as the type of classification method used, the type of research method, and the accuracy of the algorithms. Therefore, this can assist in identifying gaps in the available literature and help construct the theoretical framework.

Figure 2.1 below shows the framework for this chapter. The framework is important which will be used as guideline to follow what will be done in this chapter. The next section will discuss in detail about the ransomware such as the definition, categories, lifecycle, type of attacks, cases timeline, machine learning types, tools available, classification techniques and the analysis.

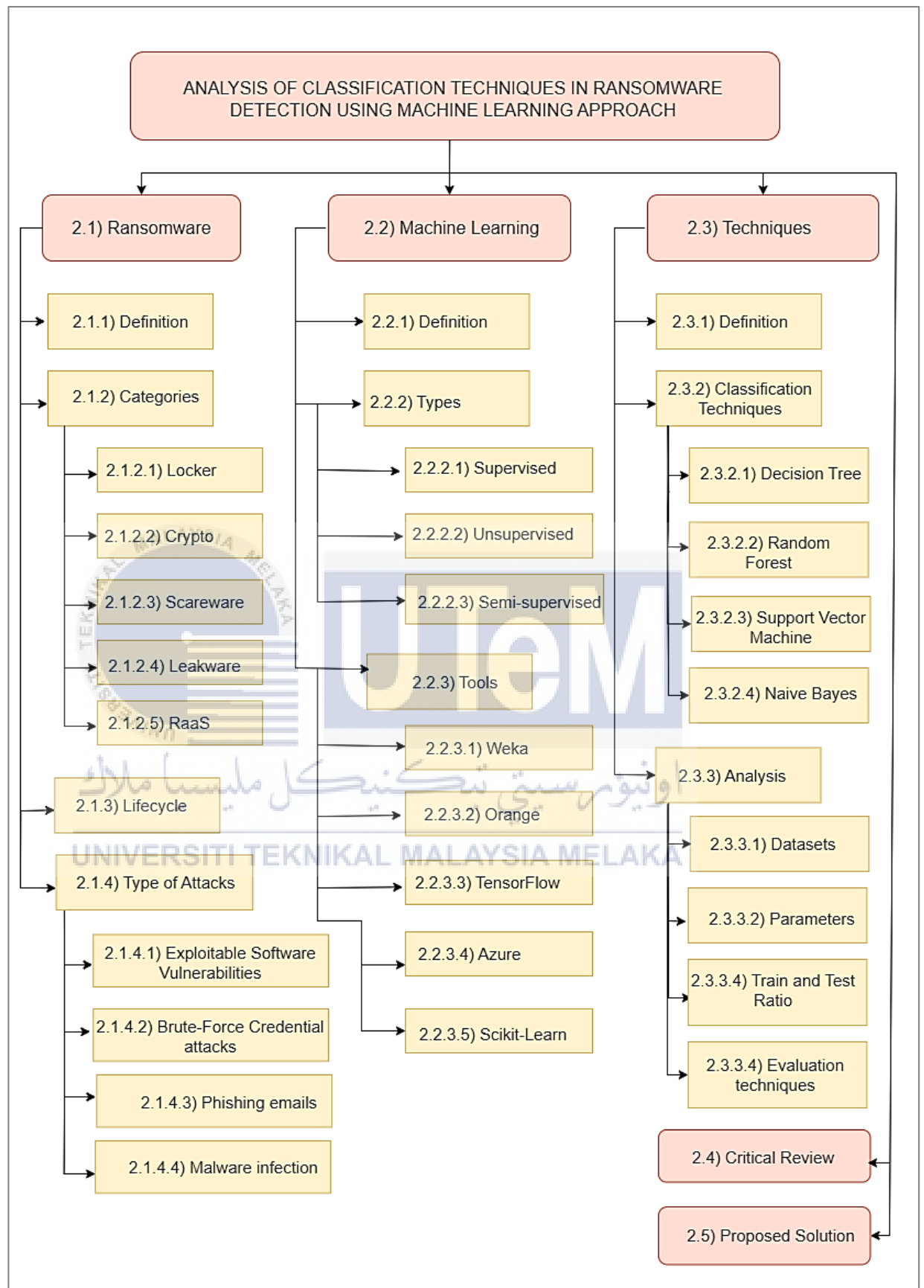


Figure 2.1 Research Framework

## 2.1 Ransomware

In this section, it includes the discussion of ransomware definition based on author from multiple existing journals, different categories of ransomware, its lifecycle, and the type of ransomware attack that are available. In addition, this section will discuss and review the ransomware cases based on the timeline. Analysis of these cases allows us to identify the pattern and common features that contribute to understanding of ransomware detection in this domain.

### 2.1.1 Definition

According to (Fernando et al., 2020) ransomware is one of the malware types that's designed to prevent user access to the target device files and the entire system. Most of the threat actors will demand fees in form of cryptocurrency such as Bitcoin as the bargain to the victim until it's paid (Horduna et al., 2023). The reason why bitcoin is the most preferred method of payment is because cryptocurrency transactions are anonymous and non-reversible. Therefore, this will make the law enforcement become harder to track down the attacker since the actual identities of the individual that is involved in the transactions are not revealed. Additionally non-reversible in this context means once the attacker receives the payment, the transaction is added to the blockchain which is non-refundable unlike the traditional financial system that allows it. Therefore, this will eliminate the risk of the victim reversing the transaction after regaining access to their data.

Other than that, similar to the research work by (Madani et al., 2023) that defines ransomware as the malware type that causes damage to the system by encrypting all the files in the victim device. The author also states that, usually to regain access to the data, the victim will need to pay the ransom fee in exchange of the decryption key. Additionally, the author specifically states that ransomware has the ability to infect devices through multiple file formats such as .exe, .doc, .docx, .ppt, .jpg, .xlsx and others. This can lead to the situation where, the user unintentionally opens a file that seems harmless but actually infected by the ransomware that can damage the metadata.

In addition to that, (Kaspersky, 2023) mentioned that the word ransom in the “ransomware” itself conveys the meaning of it. It suggests that ransomware is intended to hold the victim's file or system captive in exchange for a ransom payment. It also defines the term “malware” as the malicious software that causes harm to the victim computer system.

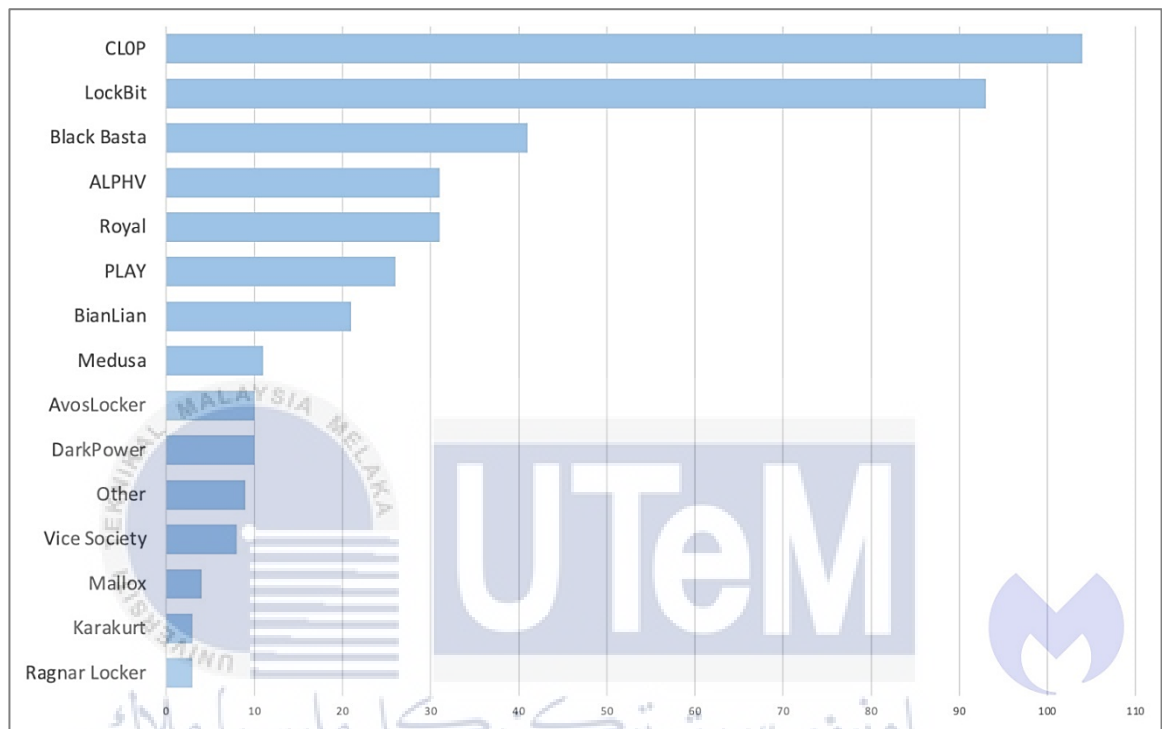


Figure 2.2 Known ransomware attacks by gang, March 2023 (Malwarebytes, 2023)

Figure 2.2 shows the ransomware attack according to the ransomware gang offender based on the report from Malwarebytes. Based from this figure we can see that the top three offender is “CLOP”, “LockBit” and “BlackBasta”. According to the news by BleepingComputer on 28<sup>th</sup> April 2023, there have been reports on ransomware activities. Microsoft has linked recent assaults on PaperCut servers to the Clop and LockBit ransomware operations, which is an intriguing scenario. Clop claims to have exploited these servers starting from April 13th, which coincided with Microsoft's observation of active exploitation of vulnerabilities. The ransomware group clarified that they used these exploits for gaining initial access to corporate networks rather than stealing archived documents.

## 2.1.2 Categories

In this section, it will include discussion of all categories in ransomware. This section will compare each type of categories to understand its behavior. As shown in the framework based on Figure 2.1, ransomware can be in the category of Locker, Crypto, Scareware, Leakware and RaaS.

### 2.1.2.1 Locker

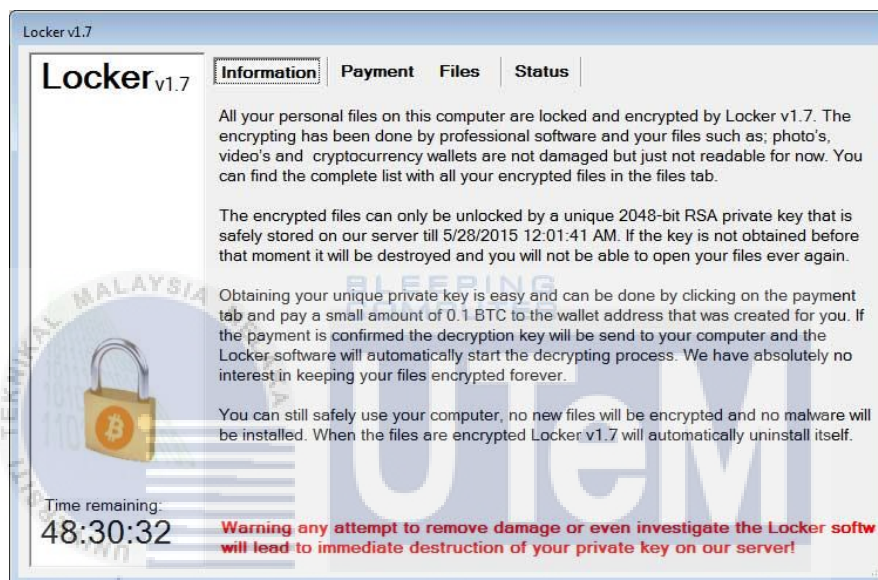


Figure 2.3 Locker ransom note shows at the infected device (Avast, 2022)

Figure 2.3 shows the information of the ransom note by Locker v1.7 as stated by Bleeping Computers. Meanwhile, according to (Horduna et al., 2023) locker ransomware has a low-level risk which blocks access to the computing resources. While the interface is locked, the victim can only use the mouse and keyboards to pay the ransom. On the other hand, (Abbasi, 2023) defines that locker ransomware will lock the computer in which the victim has limit access or even no functionality.

Usually, the malicious attacker will leave a ransom note on the victim's device lock screen. This type of ransomware is also referred as screen-lockers. (Kharraz et al., 2018) states that type of ransomware only block access to the victim's system without using encryption method.

### 2.1.2.2 Crypto



Figure 2.4 Payment method used for retrieving the private key due to the *CryptoLocker* (Avast, 2022)

As stated by (Abbasi, 2023) crypto ransomware will use encryption methods in order to lock the victim's data. The author notes that, some of the attacker use symmetric cryptography, however most of the modern ransomware will use hybrid cryptography. Hybrid cryptography approach means the attacker will encrypt the victim's file using symmetric cryptography (single key to both encryption and decryption data), and then asymmetric cryptography will be used to encrypt the symmetric session keys used for file encryption.

By implementing this approach, the attacker can send the encrypted symmetric session keys to the victim while keeping the private key used for encryption secret. In addition, (Kok et al., 2020) has emphasize that, crypto ransomware is considered to be most damaging type of ransomware. This is because many organisations have been obliged to pay in return for the decryption key, owing to the fact that encrypted files will remain unavailable even after the ransomware has been completely removed. Figure 2.4 above shows the payment procedure for the victim that has been infected with *CryptoLocker* ransomware.



### **2.1.2.3 Scareware**

As the name indicates, scareware is a type of malicious software in the category of ransomware that's designed to scare the user. According to (Horduna et al., 2023), it usually uses social engineering to trick the victim into downloading the malware. For example, a pop-up message that seems legitimate claims that the user's computer is infected with a virus. This action will intimidate the victim and encourage them to take further action such as purchasing the fake software.

On the other hand, (Abbasi, 2023) define scareware is a type of ransomware that make use of people's fear and demands purchase to be made in order to avoid worse consequences. Unlike crypto ransomware that encrypt the user system, scareware does not do anything to the victim data which make it one of the low-level risk ransomware.

### **2.1.2.4 Leakware**

According to (Horduna et al., 2023) leakware is designed not only to encrypt the victim's file but it also threatens to release the victim's data publicly. The author highlights that leakware is one of the high-risk level ransomwares because the impact can be severe for organization. (Moussaileb, 2020) states that data is the most valuable asset as it can lead to millions of dollars in terms of losses if it's released publicly. In accordance with that matter, the attacker usually uses leakware or also known as Doxware to target critical infrastructure such as the bank and other institution that work with confidential and important data .

### **2.1.2.5 RaaS**

According to (Salvi, 2019) RaaS which means Ransomware-as-a-service is a type of online subscription that's based on ransomware model. Third-party criminal entrepreneurs provide malicious users a platform for the purpose of using ransomware such as to keep the hostage computer files. The author also highlights that RaaS has become a trend because it enables attackers who lack coding skills to collaborate with ransomware developers who might not wish to carry out assaults themselves.

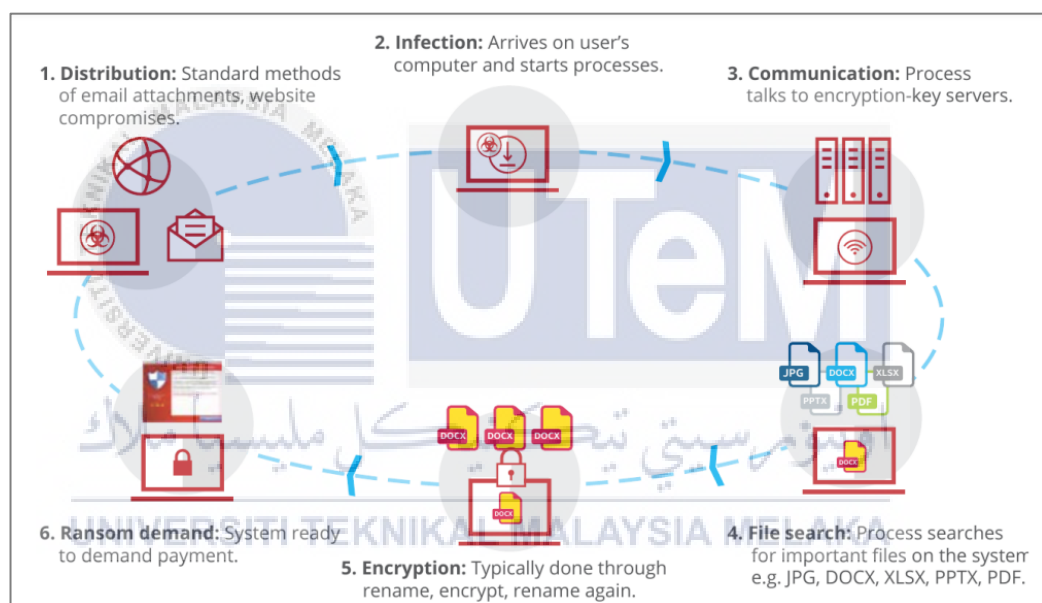
The creators will earn money by writing and adapting the code while the attacker rents the ransomware. In addition, (Horduna et al., 2023) defines RaaS to be an affiliate



scheme that benefits attackers with low technical knowledge about creating ransomware but are a member of the scheme.

In conclusion, this section helps to understand the difference between each ransomware category. Specifically for this project, based on the dataset obtained from official dataset repositories which are Kaggle, UCI Machine Learning Repositories, Resilient Information Systems Security (RISS) Ransomware Dataset, it will focus on the Crypto, Locker, Leakware and Ransomware-as-a-service (Raas) type of ransomware.

### 2.1.3 Lifecycle



*Figure 2.5 Infection phases of ransomware lifecycle (Abbasi, 2023)*

According to some existing studies, such as (Abbasi, 2023) and (Silva, 2019) the ransomware lifecycle also known as the attack kill chain has been proposed. The ransomware lifecycle refers to the series of stages it requires in performing a typical ransomware attack. As shown in Figure 2.5, the research (Abbasi, 2023) has emphasized six stages involved in the ransomware lifecycle which are Distribution, Infection, Communication, File Search, Encryption and Ransom Demand.

**i. Distribution Phase**

This phase involves the attacker finding ways or vulnerability to deliver the ransomware to the victim device or network system. According to (Abbasi, 2023) the distribution of ransomware using standard propagation method such as phishing and drive-by downloads. However, it's not only limited to that method, but some of the sophisticated ransomwares can also take advantage of exploiting different network vulnerabilities. For example, precisely as stated by (Kharraz et al., 2018) the WannaCry attacks has exploited the EternalBlue vulnerability contributing as one of the largest ransomware attacks.

**ii. Infection Phase**

Once the ransomware has been delivered, it can start to infect and compromise victim system by performing task such as collection the infected device information and disabling the anti-malware programs (Abbasi, 2023).

**iii. Communication Phase**

The attacker will communicate using command and control (C&C) server in order to fetch information and send instructions to the infected device (Cusack, 2018) On the other hand, (Abbasi, 2023) has stated that the command and control (C&C) server also can be used to get the session keys information that's used for the victim's files encryption process.

**iv. File Search Phase**

According to (Abbasi, 2023) in this phase the ransomware searches for files to be encrypted. However, different ransomware families implement different techniques for searching files to be encrypted. Traversing for files can start from randomly chosen directory, root directory or user directory (C:/Users/). As stated by (Lee, 2017) there are two methods, first is by enumerates all the file types (.ppt, .pptx, .txt, .doc, docx, etc) after scanning all the mounted file systems and the encryption will be performed. Second method is immediately encrypting the file when it detects a certain file extension.

**v. Encryption Phase**

In this phase, it will use encryption algorithm to encrypt the files. In addition to that, (Muslim, 2019) states that aside from encrypting the files it can seek and damage folder even those that's hidden and contain backup files. The files are intentionally damaged in order to impede computer users from conducting backup restore.

**vi. Ransom Demand Phase**

After encrypting the files or system, the attacker will demand for payment by displaying the ransom note in the victim device. According to (Muslim, 2019) if the ransom is not paid within given time, the price of ransom will increase. In addition to that, according to (Kharraz et al, 2018) the ransom note can be generated in different ways. For example, calls to API functions such as `CreateDesktop()` will made to create a new desktop as default configuration to displaying the ransom note while locking the victim. Other method include using HTML to create a persistent desktop message of the ransom note (Kharraz et al, 2018).

In conclusion, the above lifecycle are explained to help us in understanding the behaviour of the ransomware. This is because later in the attribute selection phase we need to understand the relationship between the API calls according to its behaviour. For example during the encryption process, one of the API calls such as “`CryptAcquireContext`”. This is because the ransomware might use this function to generate encryption keys or perform cryptographic operations during the encryption process.

## **2.1.4 Type of Attacks**

This section will explain about different types of attack vector that contribute to ransomware attacks. The attacker can carry out the ransomware attacks through various types of attack as explained below.

### **2.1.4.1 Exploitable Software Vulnerabilities**

This type of attack vector takes advantage of the flaws or weakness in the system for the attacker to gain unauthorized access. According to (Horduna et al, 2023) potential software vulnerabilities can be buffer overflows, invalidated input and remote desktop (RDP) servers. (Ilascu, 2022) observes that, in the first quarter of 2022, there was 157 software vulnerabilities was exploited due to ransomware attack.

### **2.1.4.2 Brute-Force Credential attacks**

According to Kaspersky Brute-Force Credential attacks involve the attacker attempts by forcing which means using all possible combination until the correct credential is identified. The purpose of this attack is to act as an entry point for the malicious attacker to get into the victim system or network. Once successful, the attacker can use ransomware to encrypt the data on the victim's device.

### **2.1.4.3 Phishing emails**

(Horduna et al., 2023) states that one of the most common ways for ransomware to spread is through emails. The attacker can achieve this objective by sending emails that look legitimate which contain files attachments that's secretly embedded with malicious executable files. In addition to that, (Humayun et al, 2021) emphasize that TorrentLocker and Cryptowall which is one of the family of ransomware commonly spread through spam emails.

### **2.1.4.4 Remote Desktop Protocol Attack**

Remote Desktop Protocol also known as RDP is a popular two-way communication protocol. (Van, 2023) RDP works by transferring client's keyboard and mouse input to the server and transferring the server's screen output to the client. Moreover, RDP commonly accept connection requests on port 3389, the attacker can use port-scanners

to search for devices with open ports. Thus, exploiting any security vulnerability they can find in the target device. (Van, 2023)

In conclusion, the above explanation help us to understand the attack vector of the ransomware. Based on the dataset there are two primary source which are related with ransomware that's targeting android mobile device and targeting Bitcoin transactions. Therefore, possible attack vector according to the source of ransomware sample are gathered are Exploitable Software Vulnerabilities, Brute-Force Credential Attack and Phishing Emails.

## 2.2 Machine Learning

This section will address the definition of machine learning, the different types of machine learning and available tools based on the research paper that has been collected and reviewed.

### 2.2.1 Definition

According to (Abbasi, 2023) machine leaning is a subset artificial intelligence (AI). It enables computer systems to learn new information and make decisions based on the developed intelligence (Microsoft Azure, 2023). In accordance with that, (Horduna, 2023) defines machine learning as a process of training algorithms to identify and learn patterns from collected data. Therefore, based on these identified patterns, it can use the insights to predict and improve detection of malware.

On the other hand, machine learning described by (Sen, 2021) as a system's capacity to autonomously collect, integrate, and then create knowledge from enormous volumes of data by discovering new information without having to be taught to do so. For this reason, it can be a reliable method for the problem of distinguishing new variants of malware without relying only on traditional methods such as the signature-based techniques (Gagulic et al., 2023). In addition to that, as stated by (Gagulic et al., 2023)

Random Forest, support vector machines (SVM) and k-nearest neighbors' algorithm (KNN) are some of the algorithms that can be used for ransomware detection with good performance.

### 2.2.2 Types

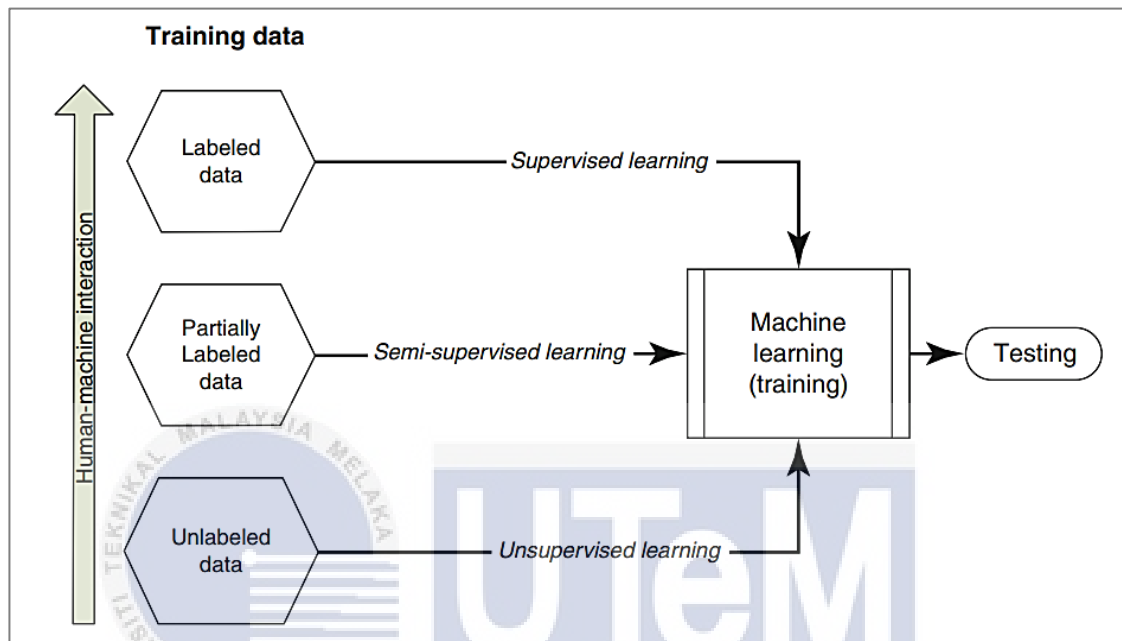


Figure 2.6 Categories of machine learning (Naqa et al., 2015)

Figure 2.6 shows the types of machine learning according to (Naqa et al., 2015). Machine learning can be divided into supervised learning, unsupervised learning and semi supervised. Based on the existing research, this section will review all of the three types of machine learning.

#### 2.2.2.1 Supervised

According to (Chumachenko, 2017) supervised learning is based on labelled data. It requires an initial datasets with labels to mapped it to the correct results. The author also points out that some of the problems that use supervised learning include regression and classification.

In accordance with that, (Abbasi, 2023) states that supervised learning main goal is to generate a model using the training data to accurately predict the outcome. In addition

to that, (Naqa et al., 2015) has similar understanding of supervised learning compared to the other research paper mentioned. The author notes that, supervised learning deals with pattern recognition in which it's programmed to learn and distinguish the data based on its repeated learning experience.

#### **2.2.2.2 Unsupervised**

In contrast with supervised learning, the author (Chumachenko, 2017) has states that unsupervised learning does not deal with labeling of data. The author of that research paper notes the goal of unsupervised learning is to identify patterns in the unsorted data rather than predicting the value. In addition to that, (Sen, 2021) states that algorithms are taught based on unlabelled, unclassified, and uncategorized data.

The author also mentioned that unsupervised learning task are usually anomaly detection, clustering, feature learning and finding association rules. In addition to that, (Abbasi, 2023) also mentioned about clustering in his paper. The author explains that clustering as creating groups of input data based on differences and similarities identified in the data. The primary goal of using unsupervised learning in clustering is to group data with similar attributes.

#### **2.2.2.3 Semi-supervised**

According to (Naqa et al., 2015) semi-supervised learning makes use of labelled and unlabelled data. Similar with the research paper by (Abbasi, 2023), the author states that semi-supervised learning is used when the data class labels are only partially available. The model is build based on supervised and unsupervised learning model in order to train the model using both label and unlabelled data.

(Abbasi, 2023) also emphasizes in semi-supervised learning typically, the labelled data is smaller than the unlabelled data. Text document categorization is one example of this, as it is almost hard to obtain numerous labelled text documents. Therefore, we can say that semi-supervised learning learns from small number of labelled instances while classifying large number of unlabelled instances in training data.

#### 2.2.2.4 Summary of type of machine learning

In conclusion, based on the reviewed research papers as discussed above this project will be using supervised learning. Supervised learning is the most suitable type because this project will involve in training the model using a labelled dataset. The datasets can be obtained from official dataset repositories such as Kaggle, UCI Machine Learning Repositories, Resilient Information Systems Security (RISS) Ransomware Dataset. Other than that, the supervised learning can also help in identifying to distinguish between ransomware and benign software. The training model can learn to recognize the patterns that contributes to ransomware and therefore detecting it.

#### 2.2.3 Tools

This section will cover all the available tools used in various research papers that relate to machine learning. The review is important to determine which tools will be using in this project.

##### 2.2.3.1 Weka

According to (Thampi et al., 2020) WEKA also known as Waikato Environment for Knowledge Analysis is a popular machine learning tool developed at University of Waikato, New Zealand. The author proposed using WEKA to classify benign and malicious categories since the tools contain machine learning algorithms for tasks such as classification, regression, and clustering. On the other hand (Sharma, 2018) proposed research on detection of advanced malware using WEKA to study and compare each classifier. The classifier used such as Random Forest, Logistic Model Tree (LMT), Naïve Bayes and others. In addition to that, (Norouzi et al., 2016) presents research on classification methodologies for detecting malware behavior using WEKA. The author also highlights the research used classification algorithms such as NaiveBayse, BayseNet, IB1, J48, and regression algorithms. After obtaining the evaluation result, the author analyzed that regression algorithms have the best performance for classification method.



### 2.2.3.2 Orange

Orange is an open-source software used for machine learning algorithms. According to (Mahajan et al., 2019) Orange uses python-based libraries which give optimized output. Similar with WEKA, orange tools can analyse various classification techniques such as Random Forest, Naive Bayes, Support Vector Machine. In addition to that, (Padmavaty et al., 2020) states that Orange tool has multiple components known as widgets which makes it easier for user to understand the navigation of the software. Other than that, in the researcher's paper (Mahajan et al., 2019) they analysed tools such as Knime, Orange and RapidMiner. Based on the researcher study, it's found that Orange presented better result compared to the others.

### 2.2.3.3 TensorFlow

According to (Prakash, 2021) TensorFlow is a free and open-source end-to-end platform for developing and deploying machine learning models. Tensorflow.org states that TensorFlow gives flexibility because it has features such as Keras Functional API and Model Subclassing API to handle large amounts of data and complex models. Other than that, TensorFlow has the ability to work with various machine learning algorithms, such as classification, linear regression, deep learning wiper, deep learning classification, boosted tree classification, and boosted tree regression.

### 2.2.3.4 Azure

Azure Machine Learning (Azure ML) is a cloud-based machine learning released by Microsoft (Jainani, 2021). Azure ML Studio has a number of modules for training, scoring, and validation. Users may obtain up to 10GB of model data storage per account, but they can also link their own Azure storage to the service for bigger models. For creating with Azure services, programmers can utilize either the R or Python programming languages. The machine learning algorithm that are popular include regression, anomaly detection, clustering, and classification.

### 2.2.3.5 Scikit-Learn

Scikit-learn is a Python open-source machine learning package that provides a variety of tools and methods for data mining and analysis. According to (Fabian, 2011) Scikit-Learn based on the three primary Python libraries such as Numpy, Spicy and Cython. It's explain that Numpy is the base data structure used for data and model parameters, while Spicy provides useful algorithms for special functions and basic statistical functions. It can also easily integrate with other numerical packages, making it easy to install and use. On the other hand, Cython is a programming language that allows users to combine C and Python for better performance.



## 2.2.4 Summary of Machine Learning tools feature

Table 2.1 Summarization of each Machine Learning tools feature based on reviewed literatures

Tools	Machine Learning features						Authors
	Usage	GUI/ Command Line	OS	Classification	Regression	Clustering	
<b>WEKA</b>	Data mining tasks	Both	Cross-platform (Windows, Linux, MacOS)	Yes	Yes	Yes	(Qolomany et al., 2019) (Sipra, 2021)
<b>TensorFlow</b>	Focus on deep learning neural network	Command line	Cross-platform (Windows, Linux, MacOS)	No	No	No	(Qolomany et al., 2019)
<b>Orange</b>	Visual programming tool for data mining and machine learning	GUI	Cross-platform (Windows, Linux, MacOS)	Yes	Yes	Yes	(Qolomany et al., 2019) (Padmavaty, 2020)
<b>Azure</b>	Open-source data analytics platform	GUI	Not Cross-platform. Only Windows and Linux.	Yes	Yes	Yes	(Qolomany et al., 2019) (Jainani, 2021)
<b>Scikit-Learn</b>	Machine learning library for Python	Command line	Cross-platform (Windows, Linux, MacOS)	Yes	Yes	Yes	(Qolomany et al., 2019) (Gupta, 2023)

In conclusion, based on reviews of the existing research articles and summarization as shown in table 2.1, the machine learning tools that will be used in this project are WEKA and Orange. WEKA and Orange are the best options because they provide a user-friendly interface using a GUI (Graphical User Interface) using widgets as navigation. On the other hand, other tools such as Scikit-learn and TensorFlow use command lines so it might be challenging for users that's beginners in machine learning tools and not expert in programming language such as Python. In comparison with Azure that's not a cross-platform, both WEKA and Orange supported various Operating System such as Microsoft Windows, Linux and MacOS. Therefore, WEKA and Orange are considered the best options as they also provide features to apply many different algorithms to the dataset to determine which one will give the best results.



## **2.3 Techniques**

This section will go through several techniques and how they will be used in this project. There are a lot of classification techniques available in machine learning and this section will only discuss techniques such as Decision Tree, Random Forest and Naive Bayes.

### **2.3.1 Definition**

According to the Cambridge Dictionary, the technique is defined as a way of performing activity or tasks. Technique can also mean the basic method for making or doing something, such as an artistic work or scientific. Therefore, in the context of machine learning techniques refer to the method that will be used for the classification task. Below is the explanation about definition of classification techniques in detail.

### **2.3.2 Classification Techniques**

According to (Brownlee, 2020), classification is a job that involves employing machine learning algorithms to figure out how to assign a class label to problem domain occurrences. The author, (Abbasi, 2023), states that classification in machine learning uses the information gained during the training phase, when training examples with associated class labels are mapped, to predict the class labels of a collection of test instances in the data. There are many different types of classification tasks that can be applied in machine learning.

### 2.3.2.1 Decision Tree

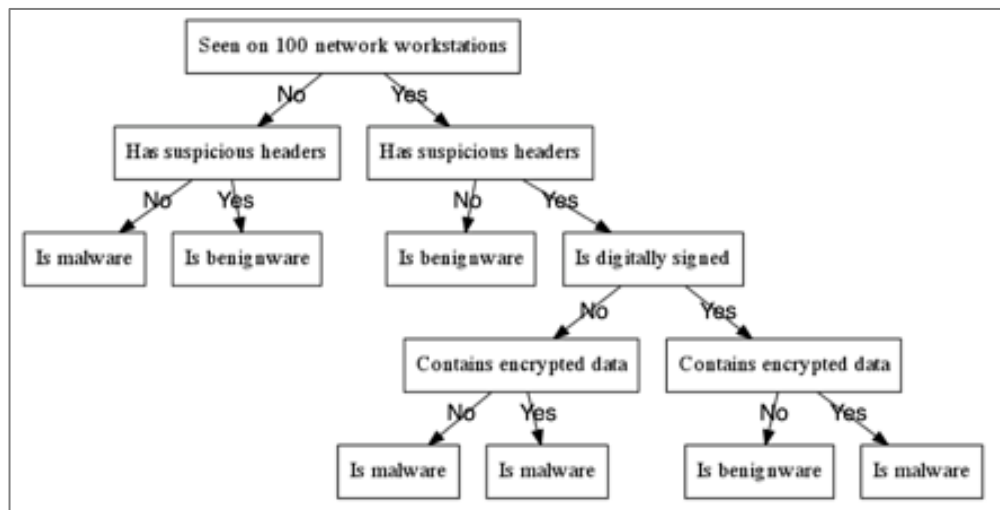


Figure 2.7 Example of decision tree (SecurityExperts.it)

Figure 2.7 illustration the search process in the decision tree. As the name indicates, decision trees are data structures with a tree structure. According to (Chumachenko, 2017), the purpose of decision trees is to produce the most accurate outcome with the fewest amount of decisions. Furthermore, the author says that it may be utilised for classification and regression issues. According to (Horduna, et al., 2023) the most used algorithm in decision tree classification technique is ID3 and it uses the concepts of Information Gain and Entropy.

According to research of (Ahmed, 2020), the author proposed a behaviour-based ransomware detection method using Windows API calls. The call sequences were split into N-grams. Based on his research, their method achieved 98.10% accuracy on average using Decision Tree classification algorithm.

### 2.3.2.2 Random Forest

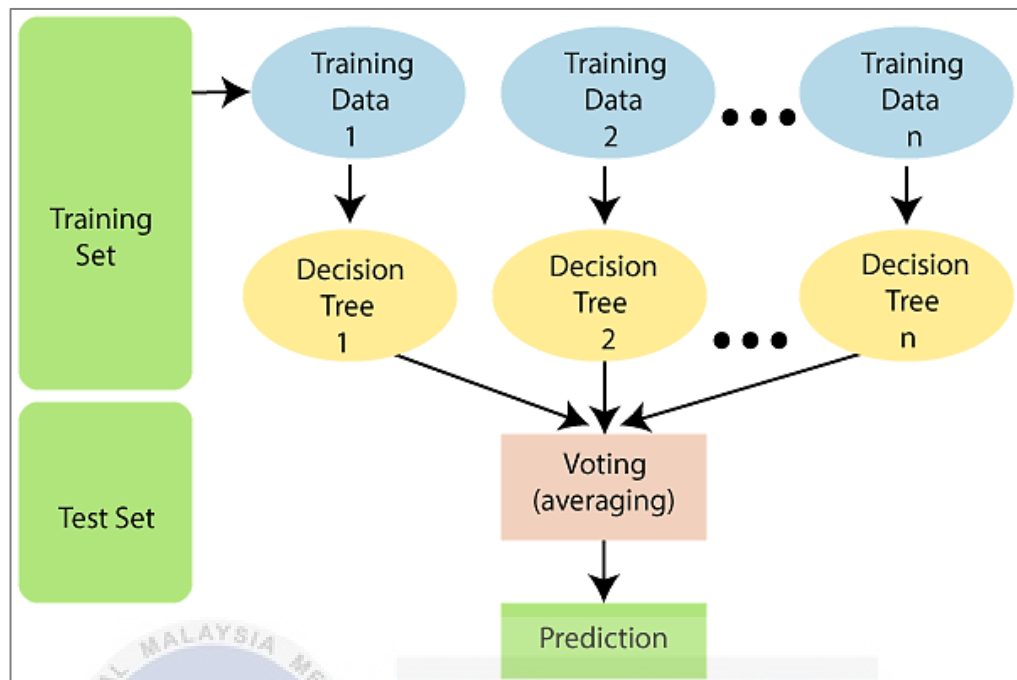


Figure 2.8 Example of Random Forest (javapoint)

Figure 2.8 shows the visualization of random forest techniques by javapoint. According to (Gagulic et al., 2023), the Random Forest model is an addition to the conventional decision tree approach that entails iteratively splitting the dataset at each decision node until a leaf node with the proper label is obtained. Other than that, the author also mentioned his opinion on Decision Tree compared with Random Forest. It's said by (Gagulic et al., 2023) decision trees are extremely sensitive to their training data, which can lead to large variation. Random Forest, on the other hand, is a collection of multiple random decision trees, making it less susceptible to training data and greatly improving the accuracy.

### 2.3.2.3 Support Vector Machine

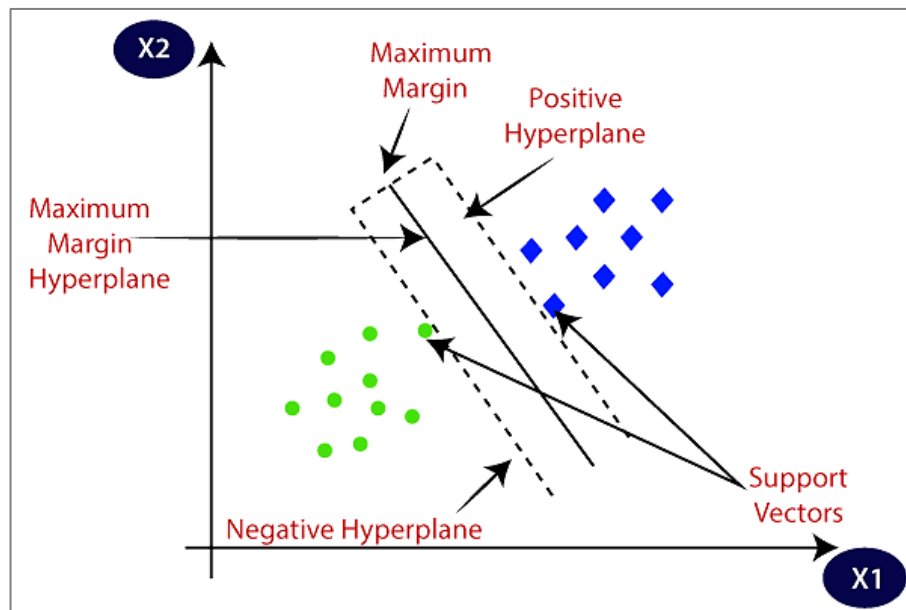


Figure 2.9 Example of Support Vector Machine (javatpoint,2023)

Figure 2.9 shows the visualization of Support Vector Machine by javatpoint to understand about the theory of hyperplane in details. According to (Horduna et al., 2023) Support Vector Machine is one of the well-known machine learning algorithms that can be used in binary classification. For example, detection of benign software or ransomware. It implements the concept of a hyper-plane that divides the points in an n-dimensional space, representing the data into two different groups. In addition to that, (Chumachenko, 2017) describes the purpose of Support Vector Machines (SVM) as finding the optimum hyperplane to split the classes.

(Chumachenko, 2017) also highlights that, the term 'support vectors' refers to points closest to the hyperplane that would change its position if removed, and the distance between the support vector and the hyperplane is known as the margin. Intuitively, the further away from the hyperplane our classes lie, the more accurate predictions we can make.



### 2.3.2.4 Naïve Bayes

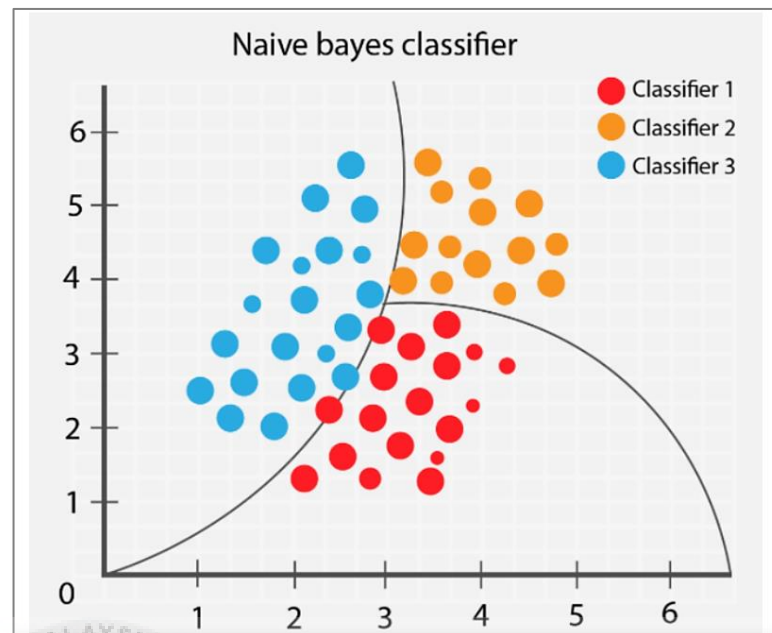


Figure 2.10 Example of Naïve Bayes (Chaudhuri, 2022)

Figure 2.10 shows the illustration of Naïve Bayes classifier according to (Chaudhuri, 2022). According to (Herrera-Silva et al., 2023) this algorithm produces probabilistic models based on target variables. It assumes that input characteristics are independent without pairwise correlation, however this is not always true. This method is labelled "naive" because of the assumption of uncorrelated characteristics. This is because real-world problems often have a correlation between features. In addition, the term Bayes is derived from the well-known probabilistic theory upon which this method generates the probabilistic model. Similar with research by (Chumachenko, K. 2017) defines Naive Bayes is classification based on the Bayes Theorem. The author notes that it predicts the probability of each feature without relations.

### 2.3.2.5 Summary of machine learning algorithms

Table 2.2 Summarization of each Machine Learning algorithms based on previous study

Included (✓), Not Included (X)

No	Research	DT	RF	SVM	NB	KNN	LR	RLR	SNN	JRip
1	(Khalil et al., 2022)	X	✓	✓	✓	✓	✓	X	X	X
2	(Kok et al., 2020)	X	✓	X	X	X	X	X	X	X
3	(Ibrahim, et al., 2020)	✓	X	X	✓	X	X	X	X	✓
4	(Khammas, 2020)	X	✓	X	X	X	X	X	X	X
5	(Abbasi, 2023)	✓	✓	✓	X	✓	X	✓	X	X

6	(Al-Haija et al., 2021)	✓	X	X	X	X	X	✓	✓	X
7	(Chumachenko, K. 2017)	✓	✓	✓	✓	✓	X	X	X	X
Occurrences		4	5	3	3	3	1	2	1	1

Table 2.3 Indicator to represent the machine learning algorithms

Indicator			
DT	Decision Tree (J48)	KNN	K-Nearest Neighbors
RF	Random Forest	LR	Logistic Regression
SVM	Support Vector Machine	RLR	Regularized Logistic Regression
NB	Naïve Bayes	SNN	Shallow Neural Networks

Table 2.4 Summarization of each Machine Learning algorithms

Included (✓), Not Included (X)

Tools	Machine Learning Algorithms			
	Decision Tree	Random Forest	Support Vector Machine	Naïve Bayes
WEKA	✓	✓	✓	✓
TensorFlow	X	✓	X	X
Orange	✓	✓	✓	✓
Azure	✓	✓	✓	✓
Scikit-Learn	✓	✓	✓	✓

In conclusion, after conducting the review of existing literature, the machine learning algorithms that will be used with the high number of occurrences are Decision Tree (J48), Random Forest, Support Vector Machine (SVM) and Naïve Bayes. Although K-Nearest Neighbors (KNN) has the same number of occurrences as Naïve Bayes and Support Vector Machine, according (Alalousi et al., 2016) K-Nearest Neighbors (KNN) are mainly used as the unsupervised classifier. The author also states that its not suitable for smaller dataset as its susceptible to overfitting because of noise in the training data. Therefore, we will not be using KNN since our scope focus on the supervised learning with accordance to the type of dataset that we acquired.

### 2.3.3 Analysis

This section will cover the parameters that will be implemented in this project. This research will also compare the three datasets and use only five parameters to analyze the data. The explanations of parameters, dataset, metric, and critical review are presented below.

#### 2.3.3.1 Datasets

According to Javatpoint a dataset is a collection of data that is organized. In the case of tabular data, a dataset refers to one or more database tables, with each row referring to a specific record in the relevant data set and each column referring to a single variable. A dataset can include any type of data, from a collection of arrays to a database table. "Comma Separated File," or CSV, is the most often used file format for tabular datasets.

(Maigida al., 2019) states that most issue in analysis is acquiring the dataset. Most research articles also faced lack of recent dataset for their research. Indirectly this can affect the training model because they are not using an updated dataset which can affect the result. In accordance with the issues mentioned above, in this project we have gathered an updated open-source ransomware dataset obtained from official dataset repositories such as Kaggle, UCI Machine Learning Repositories, File System Behavior Ransomware Dataset which are collected and analyzed from the Cuckoo sandbox.

#### **a) Dataset 1 Ransomware: UCI BitcoinHeistRansomwareAddressDataset Data Set**

The third dataset is "UCI BitcoinHeistRansomwareAddressDataset Data Set". The dataset is taken from UC Irvine Machine Learning Repository. The dataset consists of 2916697 samples or instances in total with 10 attributes. The types of ransomware family in the sample includes montrealCryptoLocker,

princetonCerber and paduaCryptoWall. The BitcoinHeist dataset was collected from the Bitcoin transaction that are associated with the ransomware payments.

*Table 2.5 Ransomware of families of dataset 3 mapped to its category*

*Included (✓), Not Included (X)*

Family	Locker	Crypto	Scareware	Leakware	RaaS
montrealCryptoLocker	✓	X	X	X	X
princetonCerber	X	✓	X	X	X
paduaCryptoWall	X	✓	X	X	X

**b) Dataset 2 Ransomware: Kaggle Dataset Android Ransomware Detection**

The first dataset that will be used, which is “Kaggle Dataset Android Ransomware Detection”, consist of 203,556 samples and 85 columns, encompassing 10 different types of Android Ransomware and Benign traffic. The type of Ransomware includes such as SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker, Jisut, Lockerpin and Pletor.

*Table 2.6 Ransomware families of dataset 1 mapped to its category*

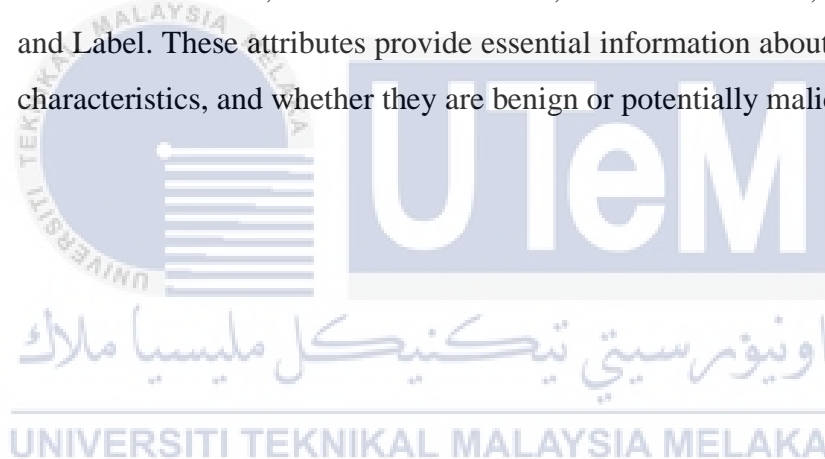
*Included (✓), Not Included (X)*

Family	Locker	Crypto	Scareware	Leakware	RaaS
WannaLocker	✓	X	X	X	X
Simplocker	✓	X	X	X	X
SVpeng	X	X	✓	X	X
PornDroid	X	X	X	✓	X
Koler,	X	X	✓	X	X
RansomBO	✓	X	X	X	X
Charger	✓	X	X	X	X
Jisut	✓	X	X	X	X
Lockerpin	✓	X	X	X	X

Pletor	✓	X	X	X	X
--------	---	---	---	---	---

**c) Dataset 3 Ransomware: Kaggle Ransomware Detection File System Behavior**

Dataset III, authored by (Bensalah, 2022), is a ransomware detection dataset designed for research purposes. It consists of 62,485 samples, with 27,119 classified as benign and the remaining samples labeled as ransomware. This dataset encompasses various columns, including FileName, md5Hash, Machine, DebugSize, DebugRVA, MajorImageVersion, MajorOSVersion, ExportRVA, ExportSize, IatVRA, MajorLinkerVersion, MinorLinkerVersion, NumberOfSections, SizeOfStackReserve, DllCharacteristics, ResourceSize, and Label. These attributes provide essential information about the files, their characteristics, and whether they are benign or potentially malicious.



**2.3.3.2 Parameters**

According to (Kizito ,2022) parameters or attributes are learned or estimated solely from the data during the training process, as the algorithm aims to understand the relationship between the input features and the corresponding labels or targets. As a

result, choosing the optimal parameter values is crucial since it has a direct impact on the model's performance when used during model training.

*Table 2.7 Parameters used in UCI BitcoinHeistRansomwareAddressDataset Dataset*

<b>Parameters</b>	<b>Description</b>
<b>ADDRESS</b>	Addresses are used for sending and receiving Bitcoin transactions.
<b>YEAR</b>	Value that indicates the year in which the transaction related to the Bitcoin address occurred.
<b>DAY</b>	Representing the day of the year
<b>LENGTH</b>	Length attribute of the transaction
<b>WEIGHT</b>	Represents information on the amount
<b>COUNT</b>	Information on the number of transactions
<b>LOOPED</b>	Number of transactions i) separate their coins; ii) transport these coins through the network via multiple pathways; and eventually, iii) combine them into a single address
<b>INCOME</b>	Amount of Bitcoin associated with the transaction.
<b>LABEL</b>	Name of ransomware family/ white (goodware)

*Table 2.8 Parameters used in Kaggle Dataset Android Ransomware Detection*

<b>Parameters</b>	<b>Description</b>
<b>Flow ID</b>	Identifier for the network flow



<b>Source IP</b>	IP address of the source (sender) of the flow
<b>Source Port</b>	Port number used by the source IP
<b>Destination IP</b>	IP address of the destination (receiver) of the flow
<b>Destination Port</b>	Port number used by the destination IP
<b>Protocol</b>	Network protocol used in the flow (e.g., TCP, UDP)
<b>Timestamp</b>	Time when the flow occurred
<b>Flow Duration</b>	Flow duration
<b>Total Fwd Packets</b>	Total number of packets sent in the forward direction
<b>Total Backward Packets</b>	Total number of packets sent in the backward direction
<b>Total Length of Fwd Packets</b>	Total length of packets in the forward direction
<b>Total Length of Bwd Packets</b>	Total length of packets in the backward direction
<b>Flow Bytes/s</b>	Data transfer rate in bytes per second
<b>Flow Packets/s</b>	Packet transfer rate in packets per second
<b>Flag counts</b>	Indicating the presence of specific flags (e.g., FIN, SYN, RST, PSH, ACK, URG)
<b>Network parameters and characteristics</b>	Window size, active and idle times

*Table 2.9 Parameters used in Kaggle Ransomware Detection File System Behavior*

<b>Parameters</b>	<b>Description</b>
<b>FILENAME</b>	Represent names of the files including the file extension (e.g., .dll)
<b>MD5HASH</b>	MD5 hash value associated with each file
<b>Machine</b>	Information about the type of machine or architecture for which the file is intended
<b>DebugSize</b>	Size of debug information within the file
<b>DebugRVA</b>	Relative Virtual Address. It contain the RVA associated with debug information within the file. RVAs are used to specify locations within a file's virtual address space.
<b>MajorImageVersion</b>	Major version number associated with the image or executable file to indicate significant updates or changes in the software.
<b>MajorOSVersion</b>	Indicate the major version number of the operating system for which the file is intended.
<b>ExportRVA</b>	Stores the RVA associated with exported functions or symbols within the file.
<b>ExportSize</b>	size of the exported functions or symbols within the file. It provides information about the resources exposed to other programs.
<b>IatVRA</b>	Represent the RVA of the Import Address Table (IAT) within the file. The IAT is essential for

	dynamically linking functions from external libraries or DLLs.
<b>MajorLinkerVersion</b>	Major version of the linker used during the file's compilation or linking process. Linkers combine object files into executables or libraries.
<b>MinorLinkerVersion</b>	Minor version of the linker used during compilation.
<b>NumberOfSections</b>	Stores the number of sections or segments within the file. Sections are distinct parts of an executable, each with specific characteristics and permissions.
<b>SizeOfStackReserve</b>	Size of the stack reserved for the file. The stack is a memory region used for managing function calls.
<b>DllCharacteristics</b>	Store characteristics or flags associated with the file, indicating its behavior as a dynamic link library (DLL). DLLs are shared libraries used by programs.
<b>ResourceSize</b>	Information about the size of resources embedded within the file. Resources can include data such as images, icons, or strings.
<b>Label</b>	Specifies the classification label for each file whether the file is benign (not malicious) or categorized as ransomware (malicious).

### 2.3.3.3 Train and Test Ratio

According to the author (Khalil et al., 2022) focuses on static analysis for detecting and classifying ransomware utilizing five machine learning algorithms which are Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR) and Naive Bayes (NB). However, the result may be

inaccurate because in his research, the author stated that the dataset was split into 50:50 ratio to train and test the detection model. This is not a good method of splitting the dataset. According to the research by (Dobbin and Simon, 2011) the optimal fraction of data splitting is 2/3 which can be applied to any dataset. Therefore, in our research we will expand the testing by evaluating various train and test ratios which are 50:50, 70:30 and 90:10. Purpose of expanding test is to observe behavior classification algorithms under several conditions.

### 2.3.3.4 Evaluation Techniques

According to (Khalil, 2022) True Positive refers to the ransomware instances that are accurately identified as ransomware (TP). The total amount of benign files is categorized accurately as benign (True Negative, or TN). False Positive (FP): a small number of benign files were mistakenly detected as ransomware. False Negative (FN): the number of ransomwares that are incorrectly categorized as benign.

#### 1) Accuracy

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

Figure 2.11 Accuracy Formula (Kok et al., 2020)

Figure 2.11 shows the calculation of accuracy. According to (Kok et al., 2020) accuracy indicates the number of correctly classified samples over the total samples on dataset.

#### 2) True Positive Rate (TPR)

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

*Figure 2.12 True Positive Rate (TPR)*

*Formula (Kok et al., 2020)*

Figure 2.12 shows the formula for True Positive Rate (TPR). The TPR is the ratio of accurately anticipated positive conditions to the total number of actual positive conditions. This statistic measures how successfully the predictive model predicts positive values.

### 3) False Positive Rate (FPR)

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

*Figure 2.13 False Positive Rate (FPR)*

*Formula (Kok et al., 2020)*

Figure 2.13 shows the formula for False Positive Rate (FPR). The FPR is the ratio of mistakenly projected positive predictions to the total number of actual negative circumstances. This indicator determines the extent to which the predictive model predicts positive values inaccurately.

### 4) True Negative Rate (TNR)

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

*Figure 2.14 True Negative Rate (TPR)*

*Formula (Kok et al., 2020)*

Figure 2.14 shows the formula for True Negative Rate (TNR). The TNR is the ratio of accurately anticipated negative conditions to the total number of actual negative conditions. This statistic measures how well the predictive model predicts negative values.

### 5) False Negative Rate (FNR)

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

Figure 2.15 False Negative Rate (FNR)

Formula (Kok et al., 2020)

Figure 2.15 shows the formula for the False Negative Rate (FNR). The FNR is the ratio of mistakenly anticipated negative conditions to the total number of actual positive conditions. This indicator determines the extent to which the predictive model predicts negative values inaccurately.

### 6) Precision

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

Figure 2.16 Precision Formula (Kok et al., 2020)

Figure 2.16 shows the formula for the precision. The precision measure represents the proportion of positive predictions that were accurately anticipated over the total number of positive forecasts. When the forecast is positive, this indicator determines how much the predictive model may be believed.

### 7) Recall

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

Figure 2.17 Recall Formula (Kok et al., 2020)

Figure 2.17 shows the formula for Recall. Recall, also known as the true positive rate, is an estimate to measure the detection rate of the positive class.

### 8) F-measure

$$F - measure = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$$

Figure 2.18 F-measure Formula (Kok et al., 2020)

Figure 2.18 shows the formula for F-measure. The F-measure, often known as the F-score or F1-score, is the average of TPR and accuracy. This statistic measures how effectively the predictive model predicts positive values while taking both into account.





6	(Al-Haija et al., 2021)	✓	X	X	X	X	✓	✓	✓	✓	X	X	X	X	X	X	X	X
7	(Chumachenko, K. 2017)	✓	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Occurrences		5	3	2	2	2	5	4	5	1	1	2	2	2	2	1	1	1

*Table 2.12 Indicator to represent the evaluation metrics.*

Indicator	
Acc	Accuracy
MCC	Matthews Correlation Coefficient
TPR	True Positive Rate (also known as Sensitivity, Recall, or Hit Rate)
PLR	Positive Likelihood Ratio
FPR	False Positive Rate
NLR	Negative likelihood ratio
TNR	True Negative Rate (also known as Specificity)
DOR	Diagnostic odds ratio
FNR	False Negative Rate
J	Youden's index
Prec	Precision
NND	Number needed to diagnose
Recall	Recall
NNM	Number needed to misdiagnose
F-m	F-measure (also known as F1 Score)
NB	Net benefit
AUC	Area Under the ROC Curve

In conclusion, after conducting a thorough review of existing literature the evaluation metrics that will be used in this project are Accuracy, TPR (True Positive Rate), Precision, Recall and F-measure. These evaluation metrics are chosen based on the number of occurrences as shown in table 2.9 with accordance to the reviewed literature reviews.

## 2.4 Critical Review

*Table 2.13 Summary of critical review for previous research articles*

Research	Type of Tools	Type of Algorithms	Type of Techniques	Problems + Objective
(Khalil et al., 2022)	<ul style="list-style-type: none"> <li>• WEKA</li> <li>• MATLAB</li> </ul>	<ul style="list-style-type: none"> <li>• Support Vector Machines (SVM)</li> <li>• K-Nearest Neighbors (KNN)</li> <li>• Random Forest (RF)</li> <li>• Logistic Regression (LR)</li> <li>• Naive Bayes (NB)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> 1) How can static analysis be used to overcome the constraints of dynamic analysis in order to construct a detection model?</p> <p><b>O:</b> 1) To propose a new technique based on static analysis for detecting and classifying ransomware utilizing five machine learning algorithms.</p>

(Kok et al., 2020)	<ul style="list-style-type: none"> <li>• PEDA (Pre-encryption detection algorithm (PEDA) Cuckoo for generating datasets</li> </ul>	<ul style="list-style-type: none"> <li>• Random Forest</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> 1) How to detect presence of crypto-ransomware before any encryption occurs?</p> <p><b>O:</b> 1) To propose development of pre-encryption detection algorithm (PEDA) for early detection of crypto-ransomware.</p> <p>2) To propose new metrics for the evaluation of a predictive model used in ransomware detection.</p>
(Ibrahim, et al., 2020)	<ul style="list-style-type: none"> <li>• WEKA</li> <li>• Orange</li> <li>• Scikit</li> </ul>	<ul style="list-style-type: none"> <li>• NaiveBayse</li> <li>• JRip</li> <li>• Decision Tree</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How to address the challenges and problems associated with ransomware behavior detection and classification.</p> <p><b>O:</b> To produce solutions for feature selection in machine learning for drive-by download problem</p>
(Khammas, 2020)	<ul style="list-style-type: none"> <li>• WEKA</li> </ul>	<ul style="list-style-type: none"> <li>• Random Forest (RF)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How can we overcome the issue of complicated disassemble process when detecting ransomware attacks?</p> <p><b>O:</b> To propose a new method of ransomware detection using Random Forest technique based on static analysis.</p>

(Abbasi, 2023)	<ul style="list-style-type: none"> <li>• Cuckoo Sandbox</li> <li>• Tensorflow</li> </ul>	<ul style="list-style-type: none"> <li>• Regularized Logistic Regression (RLR)</li> <li>• Random Forest (RF)</li> <li>• Decision Tree (DT)</li> <li>• Support Vector Machines (SVM)</li> <li>• k-Nearest Neighbors (KNN)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How can the challenges of high-dimensional data and time-intensive manual inspection in behavior-based ransomware detection be overcome?</p> <p><b>O:</b> To propose a new representation of API call sequences, for early ransomware detection.</p>
(Al-Haija et al., 2021)	<ul style="list-style-type: none"> <li>• MATLAB</li> </ul>	<ul style="list-style-type: none"> <li>• shallow neural networks (SNNs)</li> <li>• Decision Tree (DT)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How to identify and detect ransomware attacks in early detection of bitcoin transaction.</p> <p><b>O:</b> To develop a predictive system that can classify ransomware payments for heterogeneous bitcoin networks.</p>

Table 2.11 above shows the summary of the critical review. The table is generated from reviews based on the existing research papers. The author (Khalil et al., 2022) proposed a new technique based on static analysis for detecting and classifying ransomware utilizing five machine learning algorithms which are Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR), Naive Bayes (NB). According to the experimental result, the Random Forest achieved the highest detection accuracy. However, the result may be inaccurate because in his research, the author stated that the dataset was split into 50:50 ratio to train and test the detection model. This is not a good method of splitting the dataset. According to the research by (Dobbin and Simon, 2011) the optimal fraction of data splitting is  $2/3$  which can be applied to any dataset.

On the other hand, the research by (Kok et al., 2020) proposes development of pre-encryption detection algorithm (PEDA) for early detection of crypto-ransomware. In the author research (Kok et al., 2020) produces six new metrics for the evaluation of a predictive model used in ransomware detection. The metrics are Likelihood ratio(LR), Diagnostic odds ratio (DOR), Youden's index (J), Number needed to diagnose (NND), Number needed to misdiagnose (NNM) and Net benefit (NB). According to the result and discussion of the research, the author stated that proposed metrics include PLR, NNM, DOR, J Index, and NND is difficult to represent into one graph, as proposed metrics (DOR,PLR) may have an infinite value. This may result in a misunderstanding of the true metric value and the new metrics has not yet been tested for various research domain.

In the research work by (Abbasi, 2023) the author propose a new representation of API call sequences, for early ransomware detection. The research has proof that identifying critical call arguments alongside API call names in sequences can help improve the classification performance. However, there are limitation in this work because the scope of the analysis environment is limited. The research excludes ransomware that targets multiple operating systems such as Linux and Mac, as well as devices such as mobile phones. The findings are limited to ransomware that's compatible with Windows 7 PCs only.

Therefore, for our research we will only focus on the common evaluation metrics as provided by the WEKA program to calculate the accuracy of the classifier

performance. Other than that, we will be using a cross validation method using a 2/3 fraction to split the dataset as recommended by (Dobbin and Simon, 2011) in our research to improve the previous research work. In addition to that, to improve the previous issues of not using ransomware sample of a bigger environment because the author only use ransomware that's compatible with Windows 7 PCs. We will be evaluating ransomware that's targeted the mobile devices as well by using the dataset provided by Kaggle.

## **2.5 Proposed solution**

Based on the reviews of existing literature and a comprehensive evaluation for the critical analysis, this project will be using supervised learning. Supervised learning is the most suitable type because this project will involve training the model using a labelled dataset. Dataset will be downloaded from Kaggle, UCI Machine Learning Repositories and (RISS) Ransomware Dataset. It's chosen because they are the most often utilized open-source sites by researchers for getting datasets for their research on ransomware detection. In addition to that, the machine learning tools that will be used in this project are WEKA and Orange. WEKA and Orange are the best options because they provide a user-friendly interface. On the other hand, other tools such as Scikit-learn use command lines so it might be challenging for users that's beginners in machine learning tools. It also provides feature to apply many different algorithms to the dataset. According to the critical review, the classification technique that will be used with the most occurrence is Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes. Other than that, to evaluate the performance of the experiments conducted in this study, the chosen evaluation metric is Accuracy, TPR (True Positive Rate), Precision, Recall and F-measure.

## **2.6 Summary**

In conclusion, the project will focus on analysis of ransomware detection using machine learning. Based on the discussion above the classification techniques that will be use are Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes. The evaluation metric are Accuracy, TPR (True Positive Rate), Precision,

Recall and F-measure. The next chapter will focus on the methodology that will be implemented.

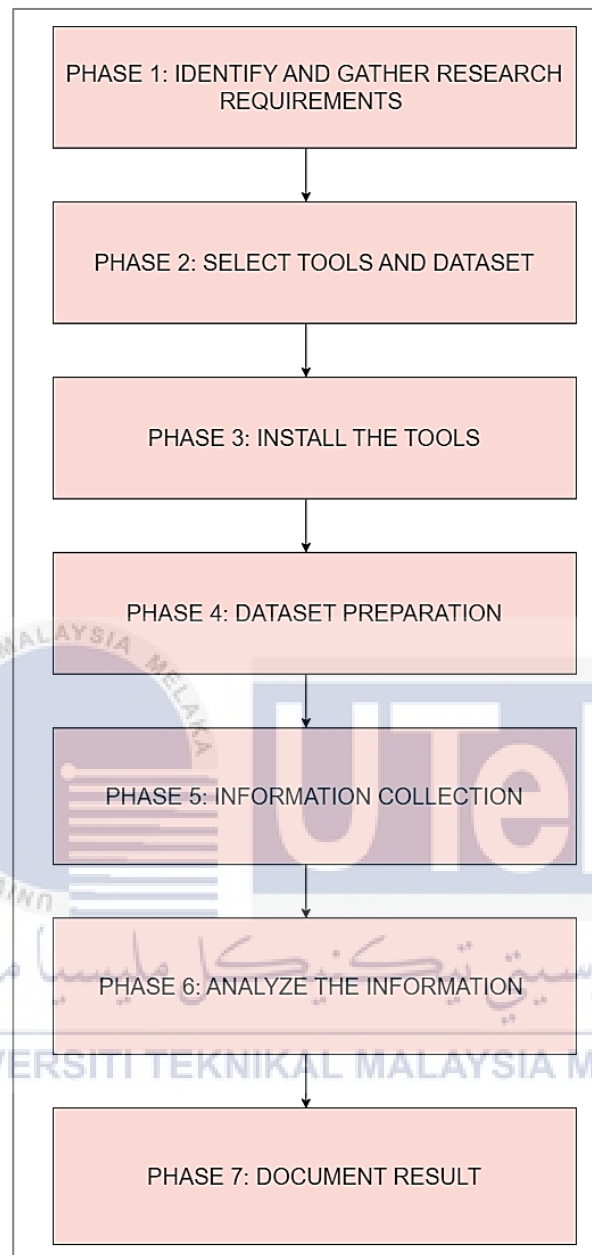
## CHAPTER 3: METHODOLOGY

### 3.0 Introduction

The research methodology used to carry out the study are defined in this chapter. This chapter describes the process used to gather, present, and analyses the data and information required to address the research purpose and question. Additionally, this chapter explicitly defines the project's phase, timetable, and milestones, as well as the project approach. The project methodology for this project is explained as follows.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### 3.1 Project Methodology



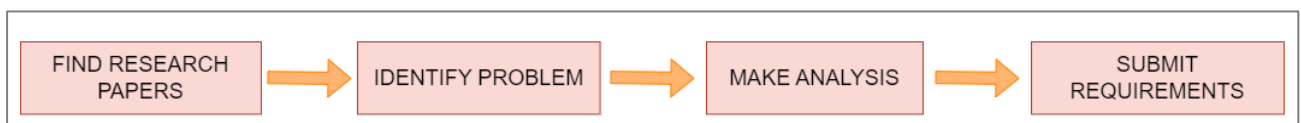
*Figure 3.1 Project methodology*

This section explains the project methodology based on Figure 3.1. The methodology that will be implemented in this project is inspired by (Yusof et al., 2019) and additional steps were added to improve the previous methodology used by the author. Figure 3.1 shows the analysis methodology for ransomware detection which consists of 7 phases. The phases include Phase 1 of this project which is identifying and gathering project requirements.



It involves justifying problem statements based on the reviews of research papers and submission of the project requirements. Phase 2 involves choosing the tools and dataset and Phase 3 involves installing the tools that will be used in the project. In addition to that, Phase 4 involves dataset preparation, Phase 5 involves information collection, Phase 6 involves information analysis, and Phase 7 involves documenting the outcomes.

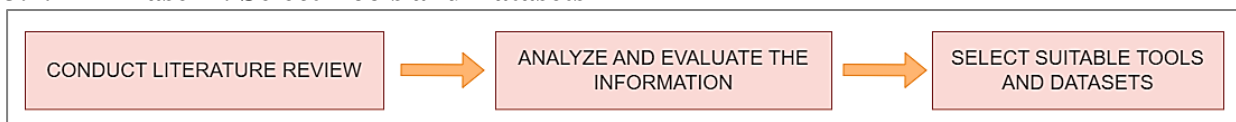
### 3.1.1 Phase I: Identify and Gather Project Requirements



*Figure 3.2 Process to Identify and gather project requirement*

Figure 3.2 shows the activity to conduct Phase 1 which is identifying and gather project requirements. Initially, before the research can begin the first step of the research is to identify common problems in the ransomware domain. This can be done by analyzing existing research papers to gain insight into current challenges in the targeted domain. This step is important for us to gain a clear understanding of the issue that we aim to address throughout the entire research is conducted. The project requirement will be documented in the proposal, which will outline the problem statement, research objectives, methodology and expected outcomes. The proposal is then submitted and approved to proceed to the next step.

### 3.1.2 Phase II: Select Tools and Datasets

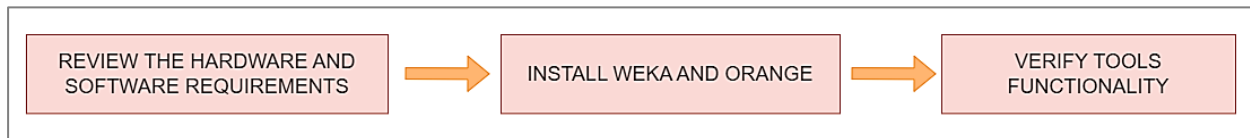


*Figure 3.3 Process of selecting tools and dataset*

Figure 3.3 shows the process of selecting tools and dataset. A comprehensive literature review is conducted so that we can identify the most suitable machine learning tools that will be used in the project for ransomware detection. Based on the gathered information of available machine learning tools, the features, its usage, type of operating system that its compatible and task that it performs will be evaluated to

identify the best tool. In our case, based on the literature review in the Chapter 2, the machine learning tools that will be used in this research are WEKA and Orange.

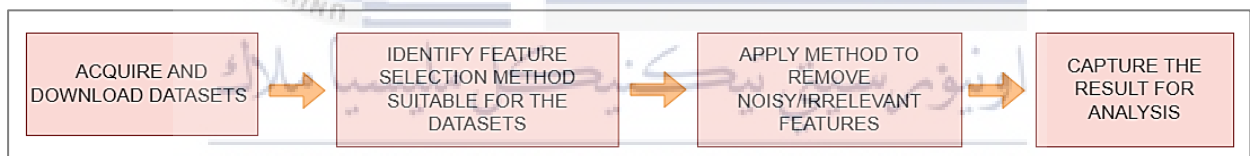
### 3.1.3 Phase III: Installing the Tools



*Figure 3.4 Process of installing the selected machine learning tools*

Figure 3.4 shows the process of installing the chosen machine learning tools. Before the installation can begin, first the hardware and software specifications will be reviewed to ensure our system meets the necessary specifications. The selected machine learning tools which are WEKA and Orange are downloaded and installed according to the installation guide at their official website. After the installation is completed, the tools will be verified to ensure the functionalities.

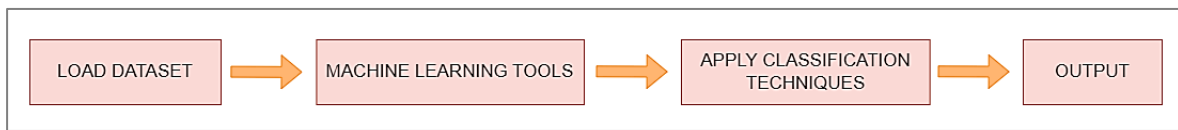
### 3.1.4 Phase IV: Dataset Preparation



*Figure 3.5 Process of Dataset Preparation*

Figure 3.5 shows the process of Dataset preparation. In this step, the dataset will be acquired and downloaded from official datasets repositories. In our case, the dataset will be acquired from Kaggle and UCI Machine Learning Repositories. Additionally, in this phase it involves identifying a suitable feature selection method. This is important because by applying the chosen feature selection methods, it can remove noisy or irrelevant features from the datasets. Once it's chosen we will apply method to remove the irrelevant features. Sub-sampling will also be done using Jupyter Notebook.

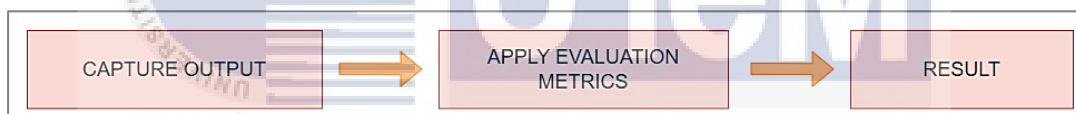
### 3.1.5 Phase V: Information Collection



*Figure 3.6 Process of Information collections*

Figure 3.6 shows the process of information collection. Once the datasets has been processed as explained in the previous step, then we can proceed with training the ransomware detection model using various machine learning classification techniques. According to the literature review, the selected classification technique that will be used with the most occurrence is Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes. After applying the classification techniques, we will collect the information generated from the machine learning tool.

### 3.1.6 Phase VI: Analyze the Information



*Figure 3.7 Process of analyze the information*

Figure 3.7 shows the process of information analysis. This step includes analyzing the information generated from the ransomware detection model. It involves capturing the output that will serve to evaluate the performance of the model. Various evaluation metrics will be applied to evaluate the accuracy result of the machine learning classification techniques. This is a crucial step because these evaluation metrics will provide information of different aspect of the classifier's performance such as its ability to correctly classify ransomware instances and its ability to minimize false positives or false negatives. To be specific, the research will apply evaluation metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Precision and accuracy.

### 3.1.7 Phase VII: Document Result



*Figure 3.8 Process of document result*

Figure 3.8 shows the process documenting the result. Based on the accuracy result of classification techniques using different evaluation metrics tool, comparison will be made. Therefore, we can identify the best classification techniques based on the accuracy. Lastly the result and analysis will be documented.

### 3.2 Research Milestone

*Table 3.1 Summary of Research Milestone*

Start Week	End Week	Activities
W1	W2	<ul style="list-style-type: none"> <li>Identify Project Requirements (Research Problems and Objectives)</li> <li>Proposal Assessment and Project Consultation</li> </ul>
W2	W2	<ul style="list-style-type: none"> <li>Gather Project Requirements.</li> <li>Proposal Improvement</li> </ul>
W3	W4	<ul style="list-style-type: none"> <li>Introduction Of Report Writing</li> </ul>
W4	W6	<ul style="list-style-type: none"> <li>Select Machine Learning Tools and Datasets</li> <li>Conduct Literature Review</li> </ul>
W6	W7	<ul style="list-style-type: none"> <li>Construct Methodology of Project</li> </ul>
W8	W9	<ul style="list-style-type: none"> <li>Design Project Architecture</li> <li>Installing Tools (Weka and Orange)</li> </ul>
W10	W12	<ul style="list-style-type: none"> <li>Datasets Preparation</li> </ul>
W13	W15	<ul style="list-style-type: none"> <li>Information Collections</li> <li>Testing Features in The Machine Learning Tools</li> </ul>
W15	W17	<ul style="list-style-type: none"> <li>Train The Machine Learning Classifier</li> </ul>
W18	W24	<ul style="list-style-type: none"> <li>Information Analysis</li> <li>Applying Evaluation Metric to The Classifier</li> <li>Identify Best Performance</li> </ul>
W25	W26	<ul style="list-style-type: none"> <li>Documenting Result</li> </ul>



### 3.4 Summary

This chapter has addressed the methodology that will be used in the research. As discussed above the phases include Phase 1 identifying and gathering project requirements, Phase 2 involves choosing the tools and dataset and Phase 3 involves installing the tools that will be used in the project, Phase 4 involves dataset preparation, Phase 5 involves information collection, Phase 6 involves information analysis, and Phase 7 involves documenting the outcomes. This section also outlines the research milestones and Gantt chart. The next section will discuss further about the analysis and design according to the phases of the methodology that will be implemented for this research.

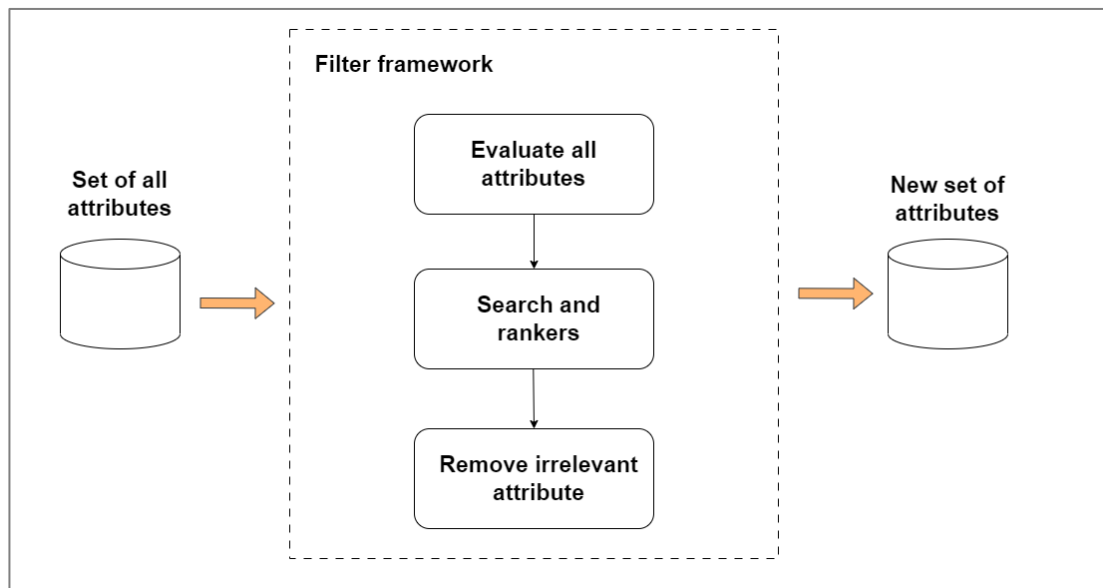


## CHAPTER 4: ANALYSIS AND DESIGN

### 4.0 Introduction

This chapter focuses on the analysis and design phase of this project. The analysis phase involves understanding the research workflow. It includes identifying problem domain, reviewing existing literature reviews, requirement identification and submission. In addition to that, the design phase will focus on choosing the suitable dataset, machine learning tools, algorithms, and evaluation metrics. The design phase is essential for establishing the procedures that must be taken to analyze the ransomware dataset. This is to ensure the project goal will be achieved, thus the recommended design will be thoroughly analyzed and studied. Below are the details for the research workflow, project requirement analysis, architecture analysis, and parameter measurement. The section of project requirements analysis will include both software and hardware requirements that will be followed for conducting the project in order to ensure smooth project execution.

#### 4.1 Research Workflow



*Figure 4.1 Workflow for preparation of dataset for Phase III of the research*

Based on the previous chapter, we have discussed the methodology that will be implemented in the research. In this section, we will identify and design the structure of the workflow to conduct the dataset preparation phase. According to (Mazumdar, 2023) it's important to process the dataset to prevent data dimensionality. This is because by identifying a subset of new attributes we can effectively minimize the impact of noisy or irrelevant attributes.

As shown in Figure 4.1, the datasets will be filtered using filter method that's based on the measure of degree association between each feature and target variable. (Mazumdar, 2023) states that, the higher the F-value the more important the feature is for the task. In addition to that, (Brownlee, 2020) also mentioned that the best method to choose feature selection method for numerical input variables associated with categorical output variable is using Information Gain.



	3N	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	
1	Avg	Bwd Avg	Bwd Avg	Bwd Avg	Subflow F	Subflow F	Subflow F	Subflow F	Init_Win	Init_Win	act_data	min_seg	Active Me	Active Str	Active M	Active M	Idle Mean	Idle Std	Idle Max	Idle Min	Label	
2	0	0	0	0	6	1076	8	4575	65535	353	3	32	0	0	0	0	0	0	0	0	0	Benign
3	0	0	0	0	2	23	0	0	1594	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
4	0	0	0	0	2	23	0	0	1486	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
5	0	0	0	0	1	31	1	0	1548	391	0	32	0	0	0	0	0	0	0	0	0	Benign
6	0	0	0	0	6	1313	7	307	65535	352	3	32	0	0	0	0	0	0	0	0	0	Benign
7	0	0	0	0	2	23	0	0	1403	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
8	0	0	0	0	1	0	2	31	1550	122	0	32	0	0	0	0	0	0	0	0	0	Benign
9	0	0	0	0	2	0	0	0	1547	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
10	0	0	0	0	2	0	0	0	1547	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
11	0	0	0	0	1	0	1	0	1594	350	0	32	0	0	0	0	0	0	0	0	0	Benign
12	0	0	0	0	1	0	1	0	1726	357	0	32	0	0	0	0	0	0	0	0	0	Benign
13	0	0	0	0	1	0	1	0	1637	349	0	32	0	0	0	0	0	0	0	0	0	Benign
14	0	0	0	0	2	0	0	0	1547	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
15	0	0	0	0	2	55	0	0	362	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
16	0	0	0	0	3	61	0	0	1909	-1	1	32	0	0	0	0	0	0	0	0	0	Benign
17	0	0	0	0	4	372	4	489	65535	59	1	32	0	0	0	0	0	0	0	0	0	Benign
18	0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	37456032	4017699	40296974	34615089	40296974	Benign
19	0	0	0	0	2	0	0	0	1547	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
20	0	0	0	0	2	0	0	0	1386	-1	0	32	0	0	0	0	0	0	0	0	0	Benign
21	0	0	0	0	4	217	6	257	65535	114	1	32	32889	127.2792	32979	32799	35508719	41704811	64998473	6018964	Benign	

Figure 4.2 Dataset II “Android Ransomware” from Kaggle

Figure 4.2 shows the content of the Android Ransomware dataset that will be used in this research. We can see the input variables consist of numerical values while the target variable which is the “label” is defined as categorical output.

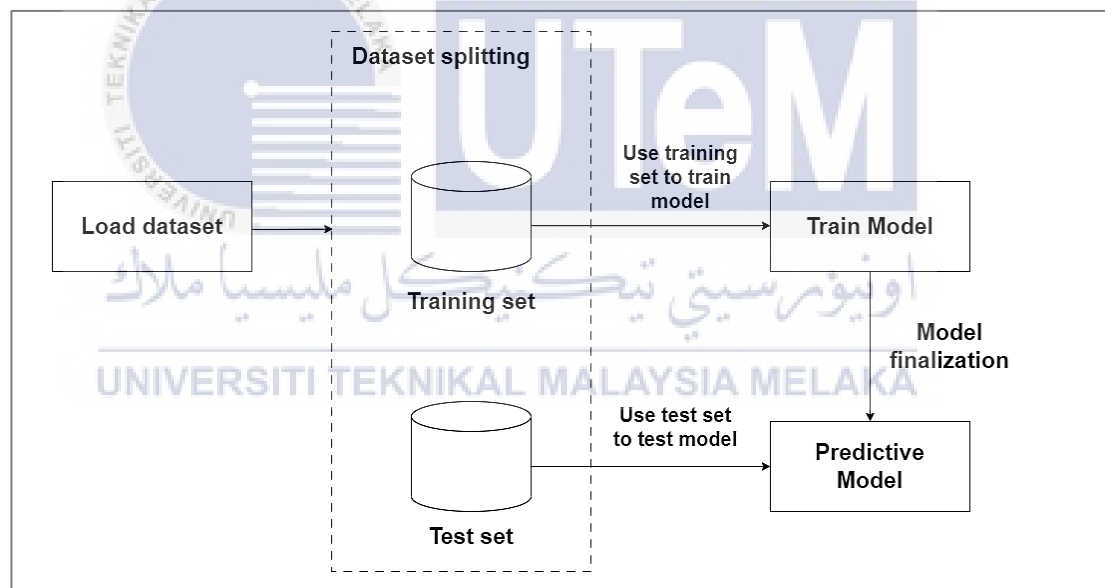


Figure 4.3 Workflow of the training and testing

As discussed in the previous chapter, Phase IV includes information collection. In the information collection phase, it involves training the machine learning classifiers. Figure 4.3 shows the illustration of how the model can be trained. (Dobbin and Simon, 2011) states that the recommended amount to split the dataset is 2/3 fraction split. Once the data is split as shown in Figure 4.2, the model will be trained and we will test the model using the test set to evaluate new samples of ransomware for the predictive model.

## 4.2 Project Requirements Analysis

The project requirement is a specific requirement that is followed for conducting the project in order to ensure a smooth functioning of the project. This section contains a reference to the project specification, which will be implemented throughout the project.

### 4.2.1 Hardware Requirements

This section focuses on determining the project's hardware prerequisites. Hardware requirements in the context of ransomware detection using machine learning using tools like WEKA and Orange will include the specifications of the computer systems and hardware requirements to run the program. This might include aspects such as the minimum CPU speed, RAM capacity, storage capacity, and network connectivity necessary to use machine learning tools efficiently. Below are the details of the hardware requirements:

*Table 4.1 Summary of hardware requirements*

Requirements	Details
Processor	Intel Dual-Core Processor or advance
RAM	A minimum of 8GB of RAM is recommended. (For larger dataset more RAM will be required)
Type of storage	Solid-state drives (SSDs)
Graphic Card	NVIDIA's / Intel's GPU

### 4.2.2 Software Requirements

Software requirements include the version numbers and specific configurations of the tools, such as WEKA and Orange, that will be used in the project. Additionally, this section will specify the operating system requirements or dependencies necessary for the software to function properly. Below are the details of the software requirements:

*Table 4.2 Summary of software requirements*

Requirements	Details
Operating System	Windows 7/8/10
Java Version	8 or latest
Python Version	3.4 or latest
Miniconda Version	miniconda3 v4.12.0
Machine Learning tool	WEKA Orange

**a) WEKA**

WEKA is a well-known open-source software package for data mining and machine learning. It offers a set of machine learning tools and methods that enable users to carry out operations including feature selection, grouping, regression, and data preparation. WEKA supports a number of file types for data entry and has an intuitive graphical user interface. For data analysis and model creation, it is widely utilised by academics and practitioners in the field of machine learning.

**b) Orange**

Orange is an open-source software which also for data mining and machine learning. Both specialists and non-experts in the area can use it because it is designed to be user-friendly and aesthetically pleasing. Classification, regression, clustering, and association rule mining are just a few of the data visualisation, preprocessing, and modelling methods that Orange provides. Additionally, it provides interactive data exploration and features a visual programming interface that enables users to build processes for data analysis without writing any code.

**c) Java**

Java is a general-purpose programming language with robust libraries and frameworks for machine learning. It provides implementations of machine learning algorithms and utilities for data processing and model building. Java is known for its platform independence, scalability, and performance.

#### d) Python

Python is a versatile programming language widely used for data analysis and machine learning. It has a rich ecosystem of libraries and frameworks dedicated to these domains. It offers tools for data manipulation, statistical analysis, visualization, and building machine learning models.

#### e) Miniconda

Miniconda is a lightweight distribution of the Python programming language. It allows users to easily create and manage Python environments for specific project requirements. It simplifies package management and is useful for setting up customized Python environments.

### 4.3 Architecture Analysis

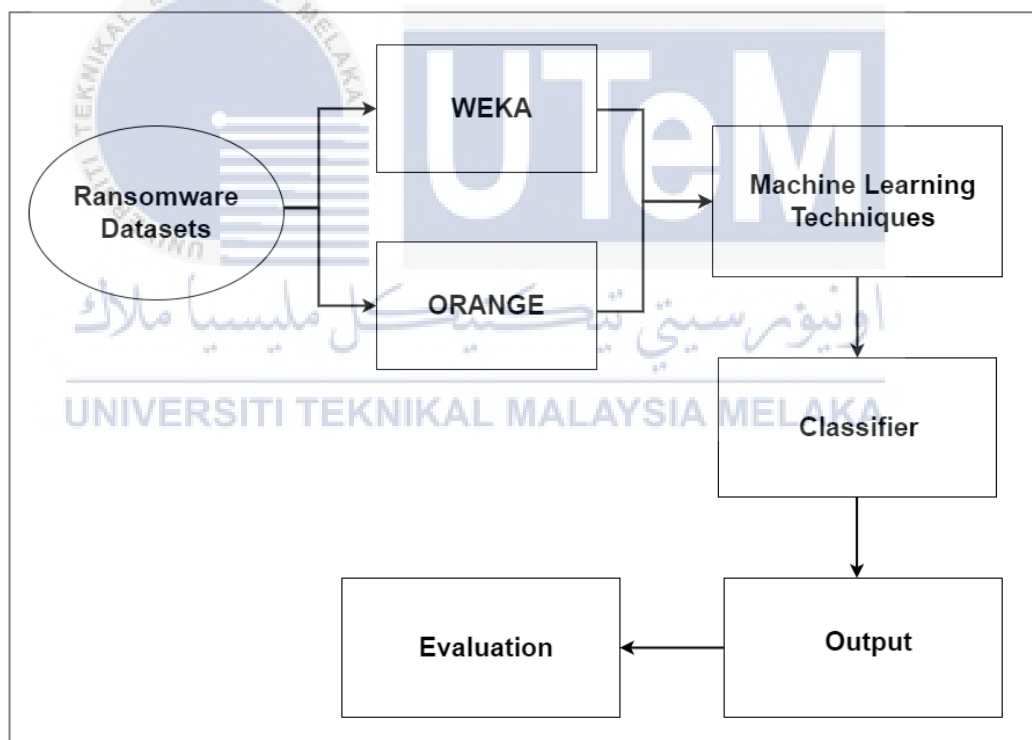


Figure 4.4 Overview of architecture analysis of this project

Figure 4.4 shows the overview architecture analysis for this project that's developed based on the proposed solution as stated in Chapter 2 of this report. The architecture analysis briefly consists of four major crucial components which are the ransomware dataset, machine learning tools, the classification techniques that will be used and evaluation of those classifiers.

#### 4.4 Proposed Research Design

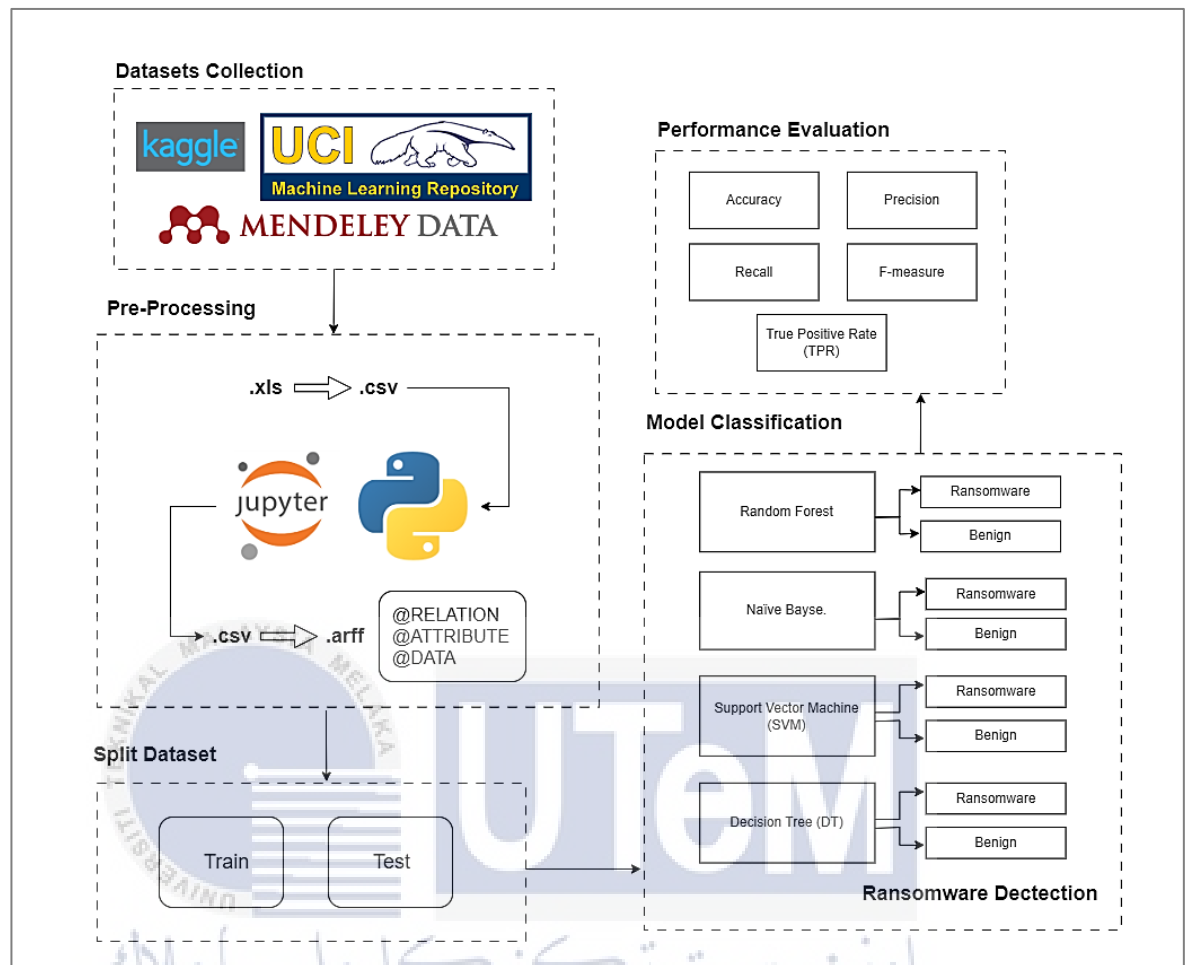


Figure 4.5 Proposed research design for ransomware detection model

Figure 4.5 visualizes the proposed research design for our ransomware detection model. The dataset will be obtained from official dataset repositories in this case such as the Kaggle Dataset Android Ransomware Detection. The dataset will then pre-process to ensure the dataset file can be loaded into the WEKA program. Pre-process in this context means defining the attributes and its instances and ensuring the data are all comma separated.

According to (Srivastava, 2014) preprocessing must include attributes selection which is important because some attribute may be redundant and noisy which can impact the performance of the classifier. Therefore, for this research, we will be using infoGainEvaluation as the ranker for the attribute selection in WEKA. After the dataset is processed and able to be loaded successfully into the WEKA program, the

dataset will be split into training and testing. The training set will be used to train the ransomware detection model whereas the testing set will be used to evaluate the model's performance. This step ensures that the model will be trained on the subset of the data and tested on the unseen data to determine its performance capability for the ransomware detection.

As for the process of model classification, four classification techniques were chosen based on the critical review in Chapter 2. The classification techniques are Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), and Support Vector Machine (SVM) where each technique has a different approach in building the classification model. Random Forest creates a group of decision trees then combines the prediction to make a final decision. Naïve Bayes is a probabilistic classifier that uses Bayes' theorem to calculate the probability of a data point belonging to a particular class. Support Vector Machine (SVM) will separate data points using hyperplanes in high-dimensional space. Decision tree on the other hand,

Lastly for the performance evaluation, the ransomware classification model will be evaluated using the testing set. The evaluation metrics that will be used include Accuracy, Precision, Recall, F-measure, and True Positive Rate (TPR). By analyzing the result of the evaluation metrics, the efficiency and accuracy of the ransomware model performance can be assessed.

#### 4.5 Flowchart Design of Research

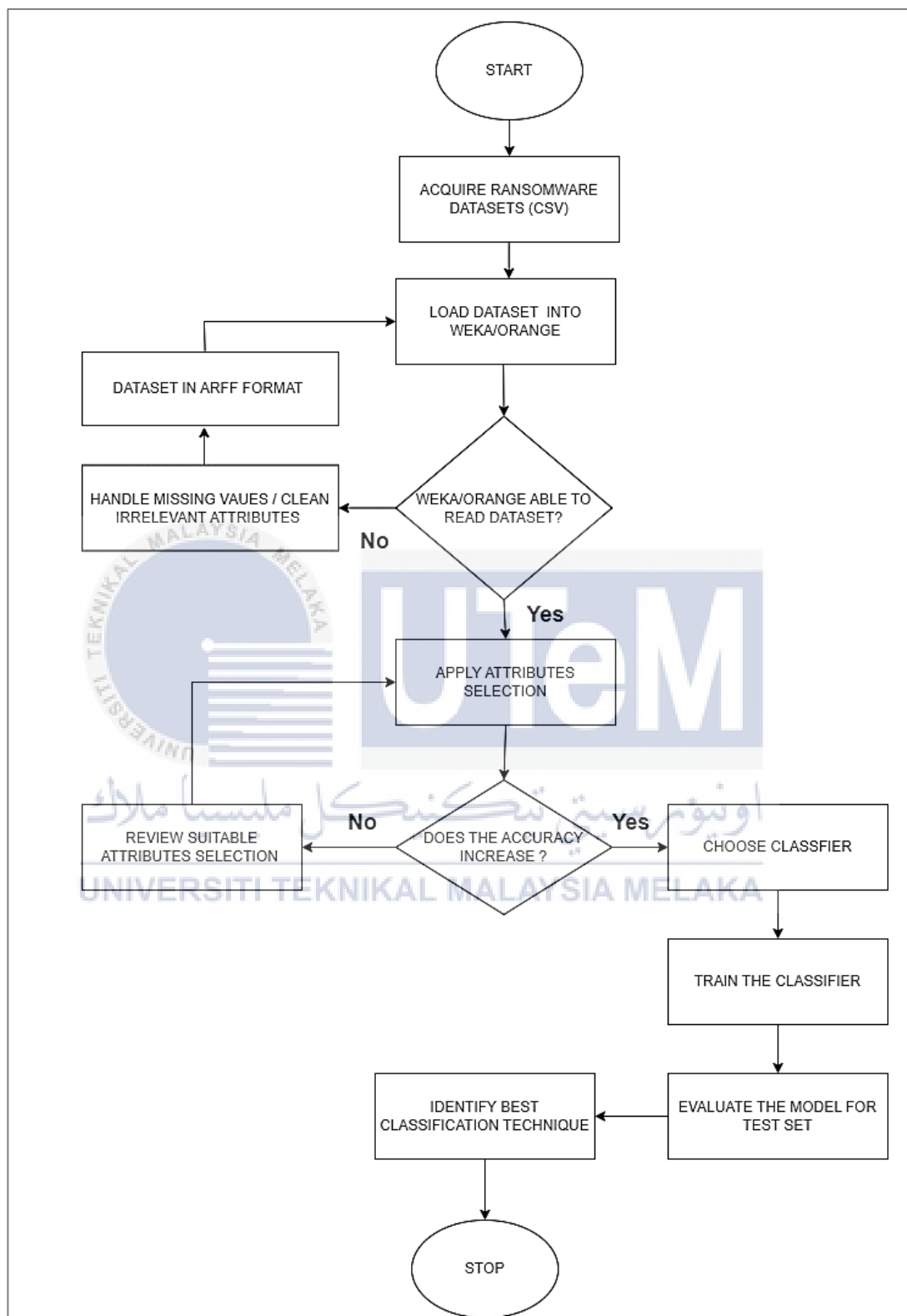


Figure 4.6 Flowchart of WEKA/ORANGE to visualize possible scenario that might occur

Figure 4.6 above shows the flowchart for the machine learning tools. Once the dataset has been downloaded, there can be 2 possibilities. If the dataset cannot be loaded into WEKA program, then we have to check the datasets if there's any missing values. If the dataset is not in the ARFF file format, then we will convert it by defining the attributes and data of the old dataset. Using the same step, which is loading the dataset, when it's successful the attribute selection will be performed to minimize noisy and irrelevant data using the filter method as discussed in the previous chapter.

Once new datasets of best feature are finalized then we will use the dataset for training and testing the model. The same phase will iterate three times since we will evaluate four types of machine learning classifiers which are Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes. The classifiers performance will be evaluated using evaluation metric are Accuracy, TPR (True Positive Rate), FPR (False Positive Rate) Precision, Recall and F-measure. After capturing the result then the WEKA program will be exit.





## 4.6 Tools interface

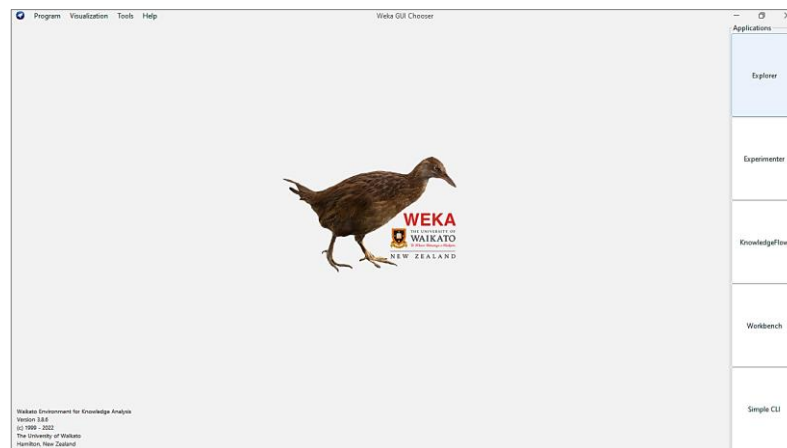


Figure 4.7 WEKA GUI chooser

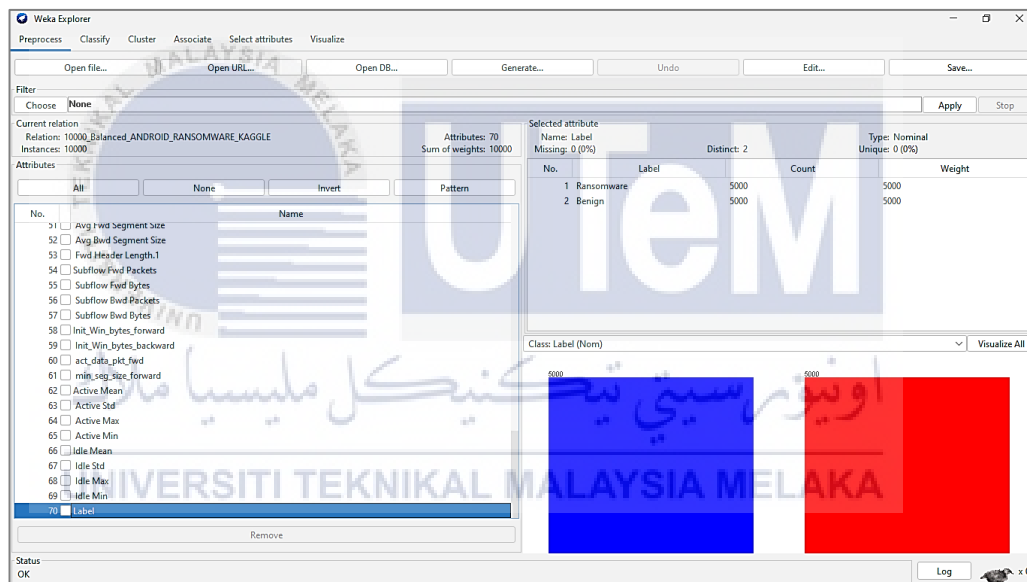


Figure 4.8 Example of result when load the Dataset II Android Ransomware to the WEKA

Figure 4.7 shows the interface for the WEKA application. First launch the WEKA application then we can see the WEKA GUI chooser which includes option for Explorer, Experimenter, KnowledgeFlow, Workbench and Simple CLI. In addition to that, at the top section it has important features such as downloading a new package manager especially for new algorithms using the “tool”. Second WEKA also can load and transform dataset from CSV to ARFF file format. Figure 4.8 shows the data visualization when we load the Dataset 1 Ransomware: Kaggle Dataset Android Ransomware Detection to the WEKA application.

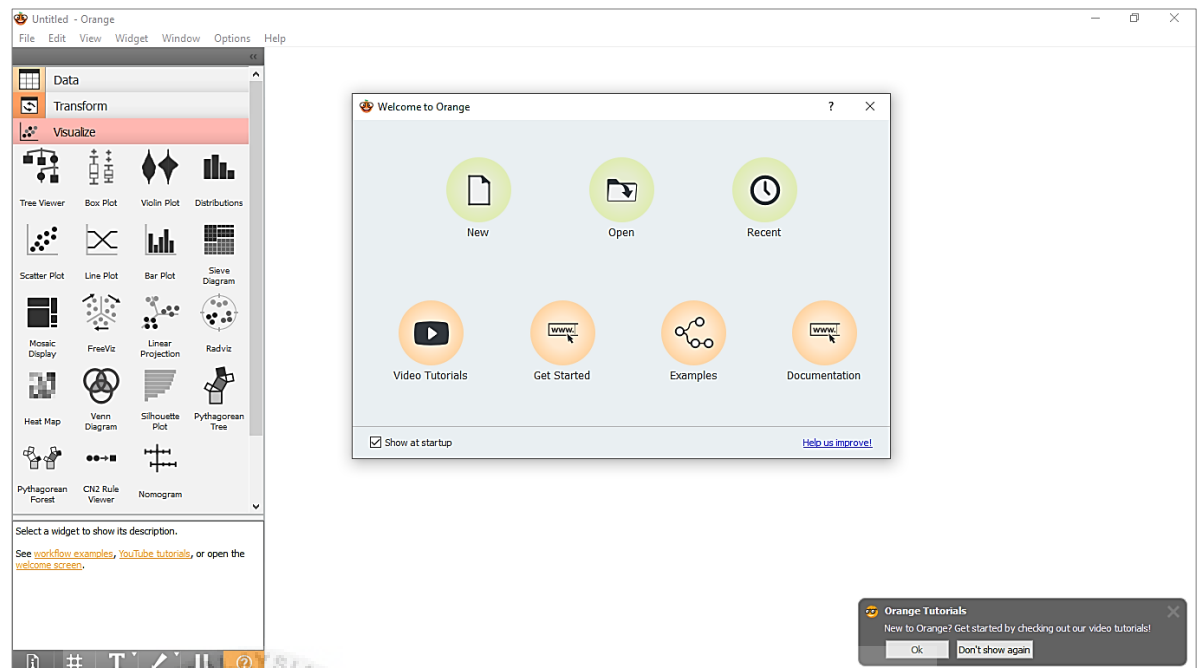


Figure 4.9 Main user interface of Orange

Based on Figure 4.9, the toolbox on the left shows the widgets available while the white canvas on the left is the working area. In order to add the widget to the canvas, we can either drag, double click or right click on the canvas menu. Datasets can be loaded using the file option or from the csv widgets in the toolbox.

## 4.8 Summary

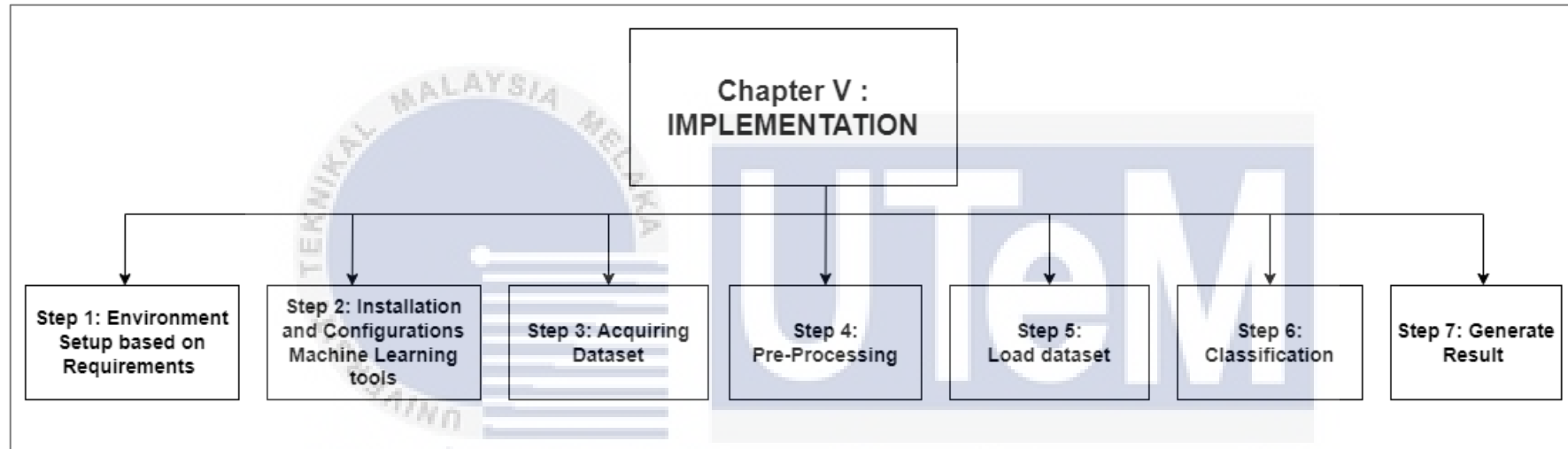
In conclusion, this chapter summarizes the analysis and design of the research. The proposed research design is visualized to illustrate the possible scenario for the project execution that will be conducted in Chapter 4 for implementation phase. This section also identifies the hardware and software requirements which are important to ensure the implementation of the research can be carried out without issues such as compatibility or not enough resources to load the dataset. The next chapter will discuss in detail about the implementation of this project.

## CHAPTER 5: IMPLEMENTATION

### 5.0 Introduction

This chapter focuses on the implementation phase of this project. The implementation of Chapter 5 includes Phase 3, Phase 4 and Phase 5 of project methodology. Based on the design consideration from the previous Chapter 4, this chapter involves applying various classification algorithms to the dataset, each of which is designed to identify patterns that distinguish ransomware samples and benign samples. The implementation process also involves configuring the machine learning tools to align with the designated algorithms, processing the dataset and starting the learning process by building the machine learning model. The implementation phase is important to see the performance of the classification algorithms chosen applied on the dataset. The result of this chapter will be used as the foundation for the next chapter, by evaluating the accuracy result of classification techniques using different evaluation metrics tools.

## 5.1 Research Implementation Activities



*Figure 5.1 Diagram Outlining Research Implementation Activities*

Figure 5.1 shows the activities involved in the research implementation, which is constructed to align and implement the research design in the previous Chapter 4. It involves 7 steps which are, Step 1: Environment Setup based on Requirements, Step 2: Installation and Configurations Machine Learning tools, Step 3: Acquiring Dataset, Step 4: Pre-Processing, Step 5: Load dataset, Step 6: Classification and Step 7: Generate Result. It's very important to outline the research implementation activities to ensure a systematic guide and organized execution of the research project. In addition to that, constructing these steps ensures minimization of error from occurring during the implementation of the research.

## 5.2 Step 1: Environment Setup based on Requirements

The environment setup for this project is a critical factor in ensuring a smooth execution of the research workflow. This section provides an overview of the essential tools, software, and hardware configurations required for the successful execution of the ransomware detection analysis using machine learning. In addition to the tools and software, the operating system is also an important consideration for the environment setup.

### 5.2.1 Hardware

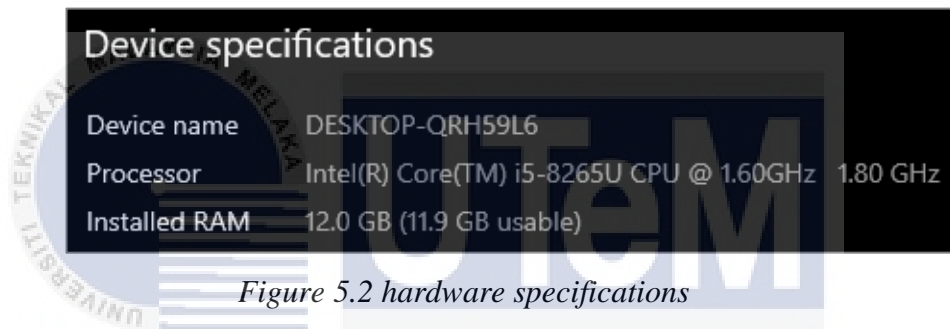


Figure 5.2 shows the device specifications from the hardware aspect. This research will be conducted using a laptop with 12 GB of RAM installed. The processor being used by the laptop is the quad-core processor Intel Core i5-8265U CPU, which operates at a base frequency of 1.60 GHz. This frequency indicates the CPU's standard processing speed. When greater computing power is required, the CPU will dynamically raise its speed up to a maximum turbo frequency of 1.80 GHz. This brings advantages to the device because it can manage tasks efficiently and provide higher performance.

## 5.2.2 Operating System

Windows specifications	
Edition	Windows 10 Home Single Language
Version	22H2
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 5.3 Operating System specifications

Figure 5.3 shows the device specifications from the operating system aspect. The Windows operating system installed on the laptop is Windows 10 Home Single Language. Its specific version is 22H2 and the system type is 64-bit operating system. In addition to that, the system type 64-bit means the computer's processor architecture is capable of handling 64-bit instructions. Therefore, it's beneficial because the device can use more memory than a 32-bit operating system which enables it to run both 32-bit and 64-bit applications efficiently.

## 5.2.3 Software

### 1) WEKA

The screenshot shows the WEKA official website. The main content area is titled 'Stable version #' and provides instructions for downloading and installing Weka. The instructions are organized by operating system and processor type:

- WINDOWS**
  - Click here to download a self-extracting executable for 64-bit Windows that includes Azul's 64-bit OpenJDK Java VM 17 (weka-3-8-6-azul-zulu-windows.exe; 133.2 MB)
  - This executable will install Weka in your Program Menu. Launching via the Program Menu or shortcuts will automatically use the included JVM to run Weka.
- MAC OS - INTEL PROCESSORS**
  - Click here to download a disk image for Mac OS that contains a Mac application including Azul's 64-bit OpenJDK Java VM 17 for Intel Macs. (weka-3-8-6-azul-zulu-osx.dmg; 180.2 MB)
- MAC OS - ARM PROCESSORS**
  - Click here to download a disk image for Mac OS that contains a Mac application including Azul's 64-bit OpenJDK Java VM 17 for ARM Macs. (weka-3-8-6-azul-zulu-arm-osx.dmg; 166.3 MB)
- LINUX**

The right sidebar contains a 'Table of contents' with links to various sections: Snapshots, Stable version, Windows, Mac OS - intel processors, Mac OS - ARM processors, Linux, Other platforms, Developer version, Windows, Mac OS - Intel processors, Mac OS - ARM processors, Linux, Other platforms, Old versions, and Upgrading from Weka 3.7.

Figure 5.4 WEKA official website

Figure 5.4 shows the official website of WEKA and its package installation. Weka is an open-source program created by academics at the University of Waikato in New Zealand. WEKA is an abbreviation for Waikato Environment for Knowledge Analysis. It was created by the international scientific community and is freely available under the GNU GPL license. WEKA is written entirely in Java. It integrates with the SQL database using Java Database connection. It includes various machine learning algorithms for implementing data mining jobs.

## 2) Orange

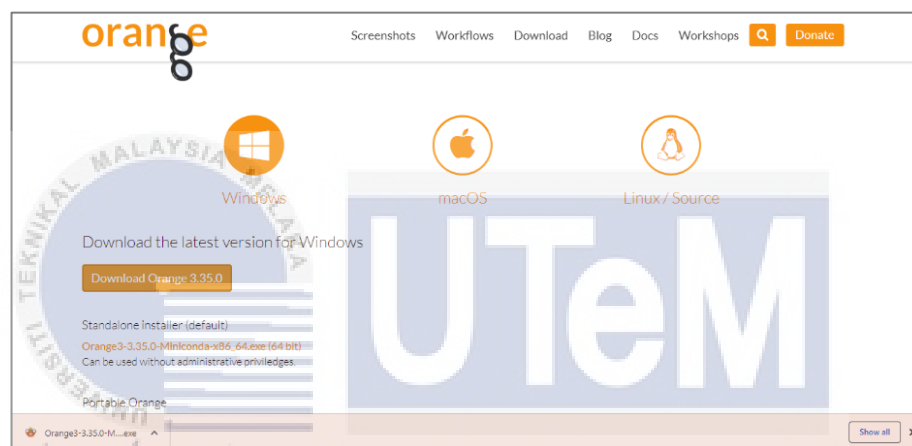


Figure 5.5 Orange official website

Figure 5.5 shows the official website of Orange with three version for Windows, macOS and Linux operating system. Orange is an open-source data mining and machine learning platform. It is intended to be user-friendly so it can be used by both specialists and non-experts in the field. Orange offers a variety of data visualization, preprocessing, and modelling approaches, including classification, regression, clustering, and association rule mining. It also offers interactive data exploration and a visual programming interface.

### 5.3 Step 2: Installation and Configurations Machine Learning Setup

This section will focus on the setup and configuration for both WEKA and ORANGE. This requires aligning the device to its baseline specifications, configuration settings, and operational complexities, assuring consistent and intended performance throughout time. It's to establish an environment which is conducive to achieving the objectives of our study by precisely configuring WEKA and ORANGE. This management strategy protects against any inconsistencies or mistakes caused by insufficient Machine Learning tools setup.

#### 5.3.1 Installation and Configurations WEKA

- 1) Download the WEKA installation file according to the type of operating system. Launch the setup and click “Next” as shown in Figure 5.6 below.



Figure 5.6 Setup Wizard of WEKA



- 2) Read the License Agreement and click “I Agree” as shown in Figure 5.7 below.

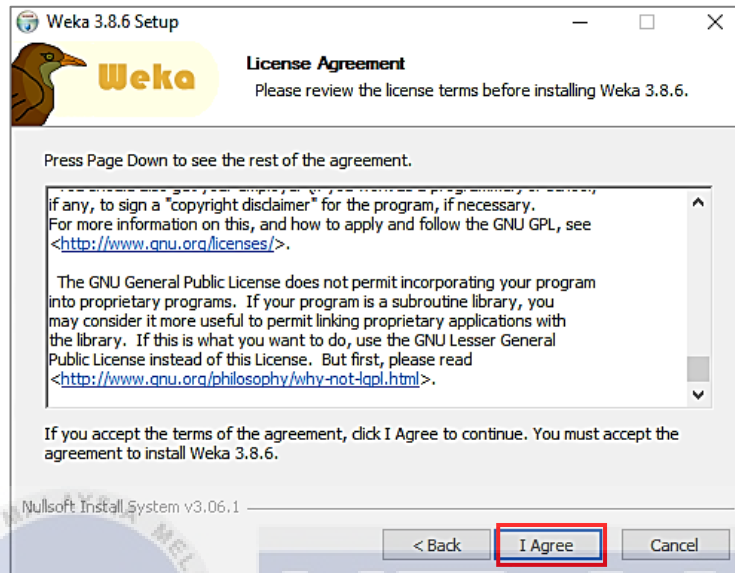


Figure 5.7 WEKA License Agreement

- 3) Select the Associated Files since Full component installation is recommended to prevent issues during the data mining tasks as shown in Figure 5.8 below.

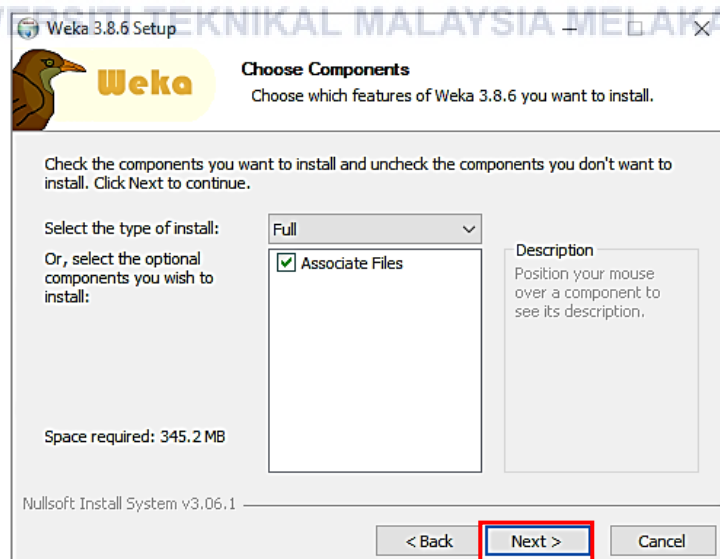
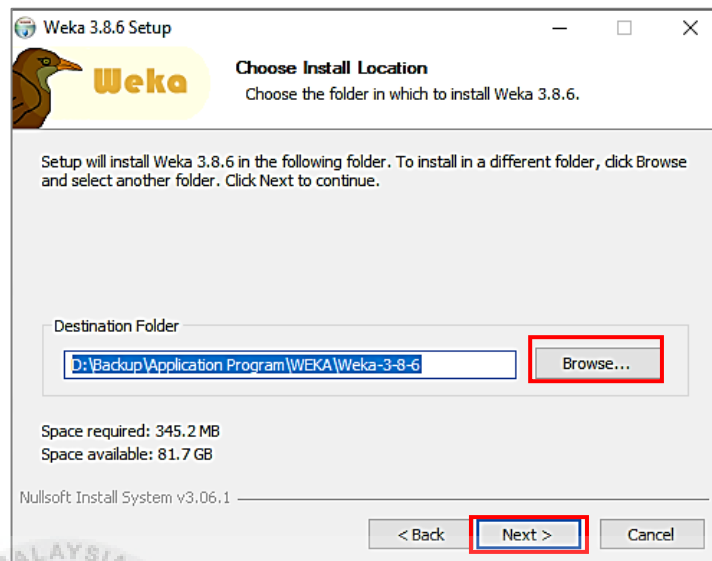


Figure 5.8 WEKA Associate Package Files

- 4) Browse for the installation file location and click Next to start the installation as shown in Figure 5.9 below.



*Figure 5.9 WEKA Installation Location*

- 5) After installation is completed, WEKA will be launched as show in Figure 5.10 below.



*Figure 5.10 WEKA interface*

## 5.4 Step 3: Acquiring Datasets

The dataset will be acquired and downloaded from official datasets repositories. In our case, the dataset will be acquired from Kaggle and UCI Machine Learning Repositories.

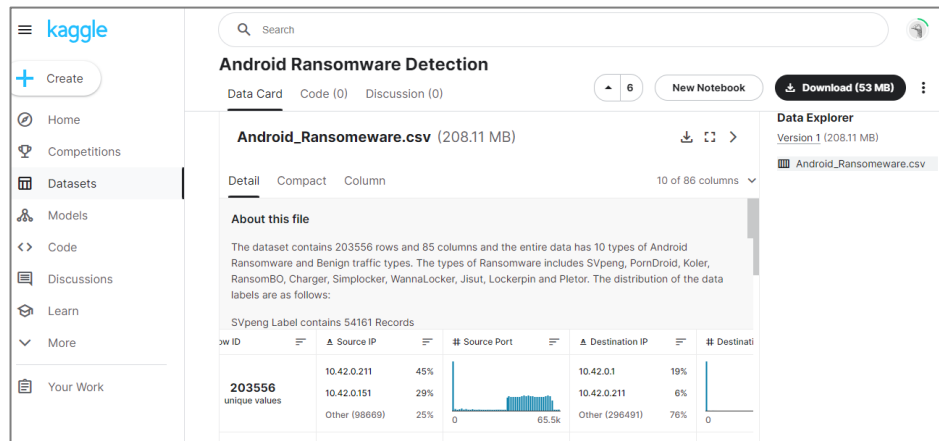


Figure 5.11 Android Ransomware Detection dataset

Figure 5.11 shows the Android Ransomware Detection dataset from Kaggle. It's distributed under the GNU Affero General Public License and serves as a resource for analyzing Android ransomware and benign traffic types.

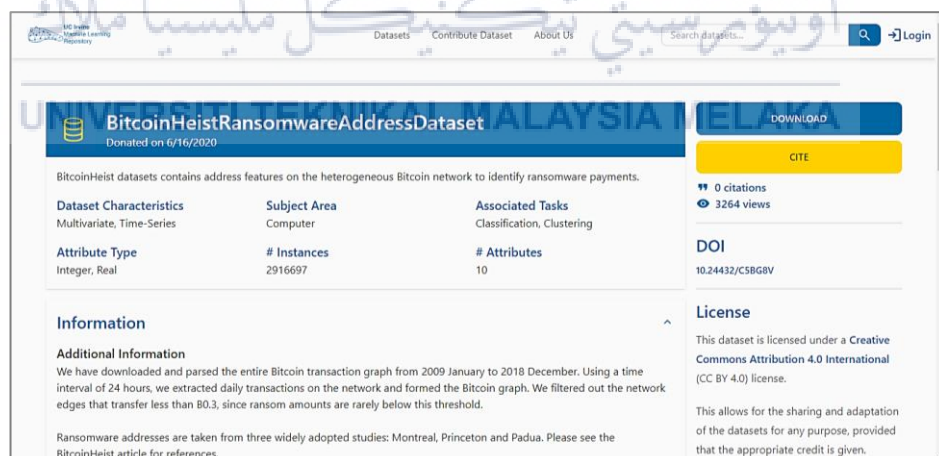


Figure 5.12 BitcoinHeistRansomwareAddress Dataset

Figure 5.12 shows the BitcoinHeistRansomwareAddress Dataset. The dataset is designed for identifying ransomware payments on the Bitcoin network. It contains features related to addresses involved in the Bitcoin network, particularly focusing on detection ransomware-related transactions.

## 5.5 Step 4: Pre-Processing

### 5.5.1 Steps for Pre-Processing

1) Select the csv file of the chosen dataset as shown in Figure 5.13 below.

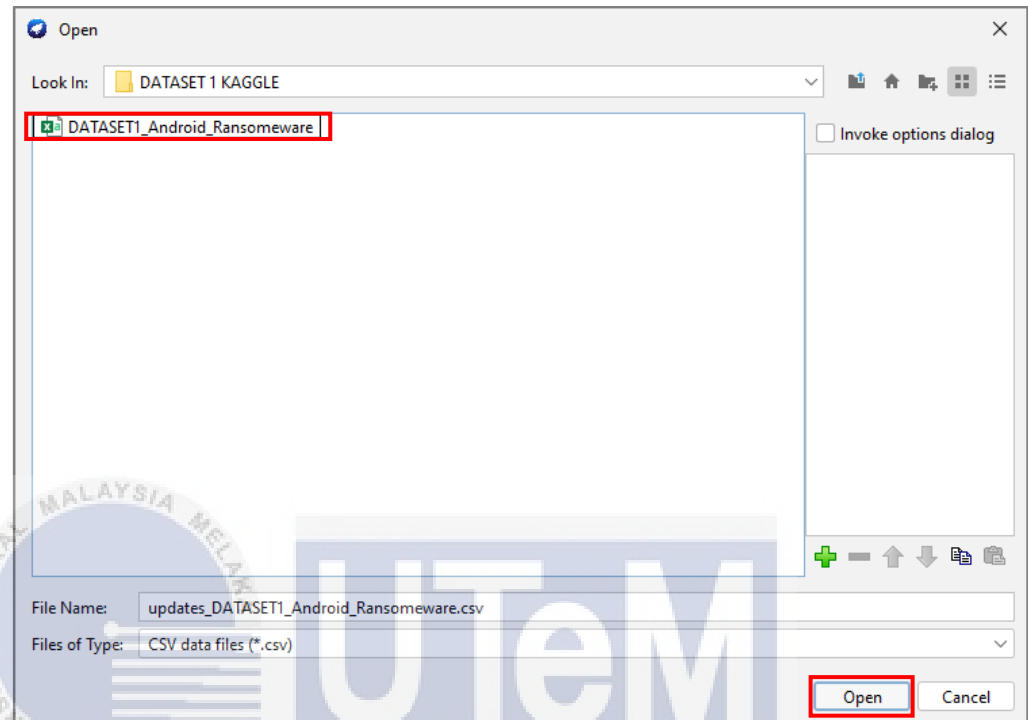


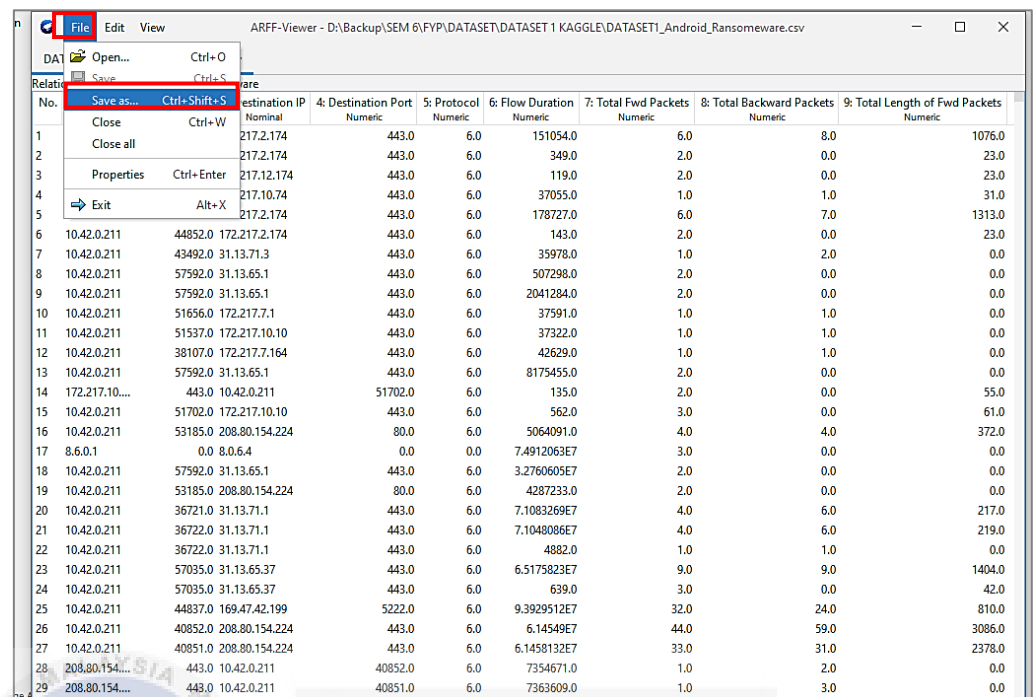
Figure 5.13 selecting the dataset in csv file format

2) The ARFF-viewer tool of WEKA will automatically categorize each attribute based on the sample type whether it's Numerical or Nominal as shown in Figure 5.14 below.

No.	1: Source IP Nominal	2: Source Port Numeric	3: Destination IP Nominal	4: Destination Port Numeric	5: Protocol Numeric	6: Flow Duration Numeric	7: Total Fwd Packets Numeric	8: Total Backward Packets Numeric	9: Total Length of Fwd Packets Numeric
1	10.42.0.211	51023.0	172.217.2.174	443.0	6.0	151054.0	6.0	8.0	1076.0
2	10.42.0.211	51023.0	172.217.2.174	443.0	6.0	349.0	2.0	0.0	23.0
3	10.42.0.211	34259.0	172.217.12.174	443.0	6.0	119.0	2.0	0.0	23.0
4	10.42.0.211	55509.0	172.217.10.74	443.0	6.0	37055.0	1.0	1.0	31.0
5	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	178727.0	6.0	7.0	1313.0
6	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	143.0	2.0	0.0	23.0
7	10.42.0.211	43492.0	31.13.71.3	443.0	6.0	35978.0	1.0	2.0	0.0
8	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	507298.0	2.0	0.0	0.0
9	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	2041284.0	2.0	0.0	0.0
10	10.42.0.211	51656.0	172.217.7.1	443.0	6.0	37591.0	1.0	1.0	0.0
11	10.42.0.211	51537.0	172.217.10.10	443.0	6.0	37322.0	1.0	1.0	0.0
12	10.42.0.211	38107.0	172.217.7.164	443.0	6.0	42629.0	1.0	1.0	0.0
13	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	8175455.0	2.0	0.0	0.0
14	172.217.10....	443.0	10.42.0.211	51702.0	6.0	135.0	2.0	0.0	55.0
15	10.42.0.211	51702.0	172.217.10.10	443.0	6.0	562.0	3.0	0.0	61.0
16	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	5064091.0	4.0	4.0	372.0
17	8.6.0.1	0.0	8.0.6.4	0.0	0.0	7.4912063E7	3.0	0.0	0.0
18	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	3.2760605E7	2.0	0.0	0.0
19	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	4287233.0	2.0	0.0	0.0
20	10.42.0.211	36721.0	31.13.71.1	443.0	6.0	7.1083269E7	4.0	6.0	217.0
21	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	7.1048086E7	4.0	6.0	219.0
22	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	4882.0	1.0	1.0	0.0
23	10.42.0.211	57035.0	31.13.65.37	443.0	6.0	6.5175823E7	9.0	9.0	1404.0

Figure 5.14 Categorization of dataset attributes

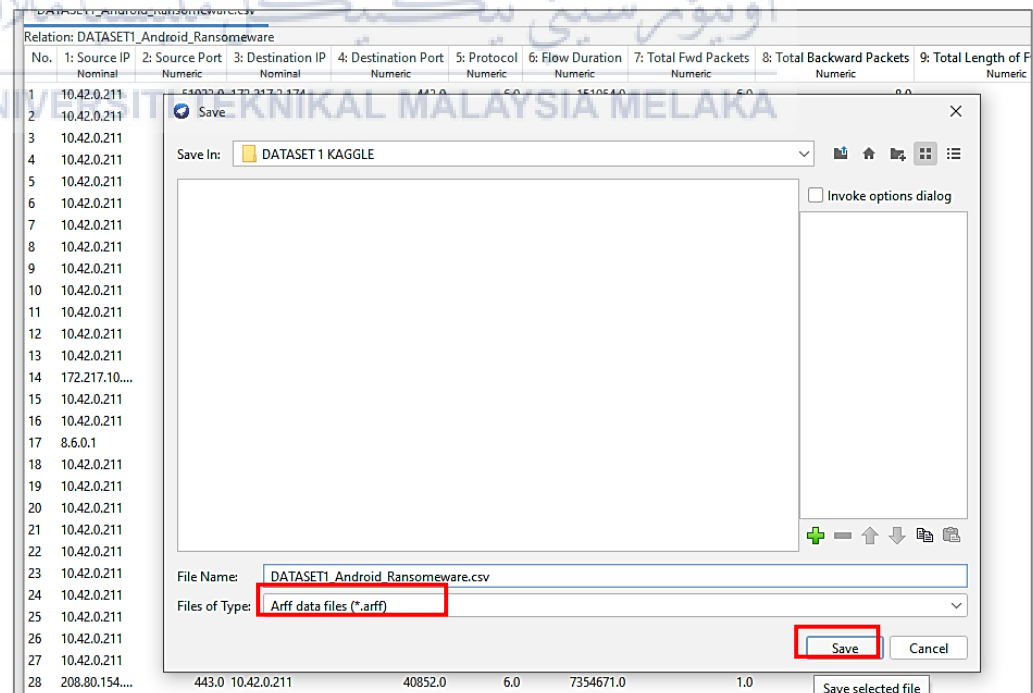
- 3) Navigate to the File tab and click “Save as” shown in Figure 5.15 below.



No.	1: Source IP	2: Source Port	3: Destination IP	4: Destination Port	5: Protocol	6: Flow Duration	7: Total Fwd Packets	8: Total Backward Packets	9: Total Length of Fwd Packets
1	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	151054.0	6.0	8.0	1076.0
2	10.42.0.211	43492.0	31.13.71.3	443.0	6.0	349.0	2.0	0.0	23.0
3	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	119.0	2.0	0.0	23.0
4	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	37055.0	1.0	1.0	31.0
5	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	178727.0	6.0	7.0	1313.0
6	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	143.0	2.0	0.0	23.0
7	10.42.0.211	43492.0	31.13.71.3	443.0	6.0	35978.0	1.0	2.0	0.0
8	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	507298.0	2.0	0.0	0.0
9	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	2041284.0	2.0	0.0	0.0
10	10.42.0.211	51656.0	172.217.7.1	443.0	6.0	37591.0	1.0	1.0	0.0
11	10.42.0.211	51537.0	172.217.10.10	443.0	6.0	37322.0	1.0	1.0	0.0
12	10.42.0.211	38107.0	172.217.7.164	443.0	6.0	42629.0	1.0	1.0	0.0
13	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	8175455.0	2.0	0.0	0.0
14	172.217.10...	443.0	10.42.0.211	51702.0	6.0	135.0	2.0	0.0	55.0
15	10.42.0.211	51702.0	172.217.10.10	443.0	6.0	562.0	3.0	0.0	61.0
16	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	5064091.0	4.0	4.0	372.0
17	8.6.0.1	0.0	8.0.6.4	0.0	0.0	7.4912063E7	3.0	0.0	0.0
18	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	3.2760605E7	2.0	0.0	0.0
19	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	4287233.0	2.0	0.0	0.0
20	10.42.0.211	36721.0	31.13.71.1	443.0	6.0	7.1083269E7	4.0	6.0	217.0
21	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	7.1048086E7	4.0	6.0	219.0
22	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	4882.0	1.0	1.0	0.0
23	10.42.0.211	57035.0	31.13.65.37	443.0	6.0	6.5175823E7	9.0	9.0	1404.0
24	10.42.0.211	57035.0	31.13.65.37	443.0	6.0	639.0	3.0	0.0	42.0
25	10.42.0.211	44837.0	169.47.42.199	5222.0	6.0	9.3929512E7	32.0	24.0	810.0
26	10.42.0.211	40852.0	208.80.154.224	443.0	6.0	6.14549E7	44.0	59.0	3086.0
27	10.42.0.211	40851.0	208.80.154.224	443.0	6.0	6.1458132E7	33.0	31.0	2378.0
28	208.80.154...	443.0	10.42.0.211	40852.0	6.0	7354671.0	1.0	2.0	0.0
29	208.80.154...	443.0	10.42.0.211	40851.0	6.0	7363609.0	1.0	3.0	0.0

Figure 5.15 Categorization of dataset attributes

- 4) Save the previous dataset with extension .csv file to convert it into an .arff file format shown in the Figure 5.16 below. Now the dataset is ready to be loaded and processed into WEKA and Orange.



No.	1: Source IP	2: Source Port	3: Destination IP	4: Destination Port	5: Protocol	6: Flow Duration	7: Total Fwd Packets	8: Total Backward Packets	9: Total Length of Fwd Packets
1	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	151054.0	6.0	8.0	1076.0
2	10.42.0.211	43492.0	31.13.71.3	443.0	6.0	349.0	2.0	0.0	23.0
3	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	119.0	2.0	0.0	23.0
4	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	37055.0	1.0	1.0	31.0
5	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	178727.0	6.0	7.0	1313.0
6	10.42.0.211	44852.0	172.217.2.174	443.0	6.0	143.0	2.0	0.0	23.0
7	10.42.0.211	43492.0	31.13.71.3	443.0	6.0	35978.0	1.0	2.0	0.0
8	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	507298.0	2.0	0.0	0.0
9	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	2041284.0	2.0	0.0	0.0
10	10.42.0.211	51656.0	172.217.7.1	443.0	6.0	37591.0	1.0	1.0	0.0
11	10.42.0.211	51537.0	172.217.10.10	443.0	6.0	37322.0	1.0	1.0	0.0
12	10.42.0.211	38107.0	172.217.7.164	443.0	6.0	42629.0	1.0	1.0	0.0
13	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	8175455.0	2.0	0.0	0.0
14	172.217.10...	443.0	10.42.0.211	51702.0	6.0	135.0	2.0	0.0	55.0
15	10.42.0.211	51702.0	172.217.10.10	443.0	6.0	562.0	3.0	0.0	61.0
16	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	5064091.0	4.0	4.0	372.0
17	8.6.0.1	0.0	8.0.6.4	0.0	0.0	7.4912063E7	3.0	0.0	0.0
18	10.42.0.211	57592.0	31.13.65.1	443.0	6.0	3.2760605E7	2.0	0.0	0.0
19	10.42.0.211	53185.0	208.80.154.224	80.0	6.0	4287233.0	2.0	0.0	0.0
20	10.42.0.211	36721.0	31.13.71.1	443.0	6.0	7.1083269E7	4.0	6.0	217.0
21	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	7.1048086E7	4.0	6.0	219.0
22	10.42.0.211	36722.0	31.13.71.1	443.0	6.0	4882.0	1.0	1.0	0.0
23	10.42.0.211	57035.0	31.13.65.37	443.0	6.0	6.5175823E7	9.0	9.0	1404.0
24	10.42.0.211	57035.0	31.13.65.37	443.0	6.0	639.0	3.0	0.0	42.0
25	10.42.0.211	44837.0	169.47.42.199	5222.0	6.0	9.3929512E7	32.0	24.0	810.0
26	10.42.0.211	40852.0	208.80.154.224	443.0	6.0	6.14549E7	44.0	59.0	3086.0
27	10.42.0.211	40851.0	208.80.154.224	443.0	6.0	6.1458132E7	33.0	31.0	2378.0
28	208.80.154...	443.0	10.42.0.211	40852.0	6.0	7354671.0	1.0	2.0	0.0
29	208.80.154...	443.0	10.42.0.211	40851.0	6.0	7363609.0	1.0	3.0	0.0

Figure 5.16 Categorization of dataset attributes

## 5.5.2 Comparison Before and After

Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Total Backward Pa
2.174-10.42.0.211-443-51023-6	10.42.0.211	51023,172.217.2.174,443,6	16/06/2017	03:55:47	151054,6,8	1076.0	4575.0	821.0
2.174-10.42.0.211-443-51023-6	10.42.0.211	51023,172.217.2.174,443,6	16/06/2017	03:55:47	349,2,0	23.0	23.0	0.0
12.174-10.42.0.211-443-34259-6	10.42.0.211	34259,172.217.12.174,443,6	16/06/2017	03:55:52	119,2,0	23.0	23.0	0.0
10.74-10.42.0.211-443-55509-6	10.42.0.211	55509,172.217.10.74,443,6	16/06/2017	03:55:53	37055,1,1	31.0	31.0	0.0
2.174-10.42.0.211-443-44852-6	10.42.0.211	44852,172.217.2.174,443,6	16/06/2017	03:55:58	178727,6,7	1313.0	307.0	753.0
2.174-10.42.0.211-443-44852-6	10.42.0.211	44852,172.217.2.174,443,6	16/06/2017	03:55:58	143,2,0	23.0	23.0	0.0
211-31.13.71.3-43492-443-6	10.42.0.211	43492,31.13.71.3,443,6	16/06/2017	03:56:44	35978,1,2	0.0	31.0	0.0
211-31.13.65.1-57592-443-6	10.42.0.211	57592,31.13.65.1,443,6	16/06/2017	03:56:44	507298,2,0	0.0	0.0	0.0
211-31.13.65.1-57592-443-6	10.42.0.211	57592,31.13.65.1,443,6	16/06/2017	03:56:46	2041284,2,0	0.0	0.0	0.0
7.1-10.42.0.211-443-51656-6	10.42.0.211	51656,172.217.7.1,443,6	16/06/2017	03:56:50	37591,1,1	0.0	0.0	0.0
10.10-10.42.0.211-443-51537-6	10.42.0.211	51537,172.217.10.10,443,6	16/06/2017	03:56:50	37322,1,1	0.0	0.0	0.0
7.164-10.42.0.211-443-38107-6	10.42.0.211	38107,172.217.7.164,443,6	16/06/2017	03:56:50	42629,1,1	0.0	0.0	0.0
211-31.13.65.1-57592-443-6	10.42.0.211	57592,31.13.65.1,443,6	16/06/2017	03:56:52	8175455,2,0	0.0	0.0	0.0
10.10-10.42.0.211-443-51702-6	10.42.0.211	51702,172.217.10.10,443,6	16/06/2017	03:57:25	135,2,0	55.0	55.0	0.0
10.10-10.42.0.211-443-51702-6	10.42.0.211	51702,172.217.10.10,443,6	16/06/2017	03:57:25	562,3,0	61.0	38.0	20.0
154.224-10.42.0.211-80-53185-6	10.42.0.211	53185,208.80.154.224,80,6	16/06/2017	03:57:41	5064091,4,4	372.0	489.0	372.0
-8.6.0.1-0-0-0,8.6.0.1,0,8.0.6.4,0,0	16/06/2017	03:55:40	74912063,3,0	0.0	0.0	0.0	0.0	0.0
211-31.13.65.1-57592-443-6	10.42.0.211	57592,31.13.65.1,443,6	16/06/2017	03:57:16	32760605,2,0	0.0	0.0	0.0
154.224-10.42.0.211-80-53185-6	10.42.0.211	53185,208.80.154.224,80,6	16/06/2017	03:57:47	4287233,2,0	0.0	0.0	0.0
211-31.13.71.1-36721-443-6	10.42.0.211	36721,31.13.71.1,443,6	16/06/2017	03:56:46	71083269,4,6	217.0	257.0	217.0
211-31.13.71.1-36722-443-6	10.42.0.211	36722,31.13.71.1,443,6	16/06/2017	03:56:46	71048086,4,6	219.0	257.0	219.0
211-31.13.71.1-36722-443-6	10.42.0.211	36722,31.13.71.1,443,6	16/06/2017	03:57:57	4882,1,1	0.0	0.0	0.0
211-31.13.65.37-57035-443-6	10.42.0.211	57035,31.13.65.37,443,6	16/06/2017	03:56:54	65175823,9,9	1404.0	745.0	1078.0
211-31.13.65.37-57035-443-6	10.42.0.211	57035,31.13.65.37,443,6	16/06/2017	03:57:59	639,3,0	42.0	42.0	0.0
42.199-10.42.0.211-5222-44837-6	10.42.0.211	44837,169.47.42.199,5222,6	16/06/2017	03:56:15	93929512,32,24	810.0	545.0	226.0
154.224-10.42.0.211-443-40852-6	10.42.0.211	40852,208.80.154.224,443,6	16/06/2017	03:57:42	61454900,44,59	3086.0	70437.0	598.0

Figure 5.17 Dataset 1 before Pre-Processing

Figure 5.17 shows the dataset before pre-processing. The provided data is in CSV format and contains network traffic information. Each row represents a network communication instance. The columns contain various attributes related to communication, such as source IP, source port, destination IP, destination port, protocol and so on. The dataset that is in CSV format, is not a standard format for machine learning datasets. Although Weka can read CSV data, it requires manual conversion. This can be a time-consuming and error-prone process.

```

@relation updates_DATASET1_Android_Ransomware

@attribute Source IP' {10.42.0.211,172.217.10.10,8.6.0.1,208.80.154.224,172.217.2.163,172.217.10.74,74.125.22.10}
@attribute Source Port' numeric
@attribute Destination IP' {172.217.2.174,172.217.12.174,172.217.10.74,31.13.71.3,31.13.65.1,172.217.7.1,172.217.10.10,10.74.10.42}
@attribute Destination Port' numeric
@attribute Protocol' numeric
@attribute Flow Duration' numeric
@attribute Total Fwd Packets' numeric
@attribute Total Backward Packets' numeric
@attribute Total Length of Fwd Packets' numeric
@attribute Total Length of Bwd Packets' numeric
@attribute Fwd Packet Length Max' numeric
@attribute Fwd Packet Length Min' numeric
@attribute Fwd Packet Length Mean' numeric
@attribute Fwd Packet Length Std' numeric
@attribute Bwd Packet Length Max' numeric
@attribute Bwd Packet Length Min' numeric
@attribute Bwd Packet Length Mean' numeric
@attribute Bwd Packet Length Std' numeric
@attribute Flow Bytes' numeric
    
```

Figure 5.18 Dataset 1 with defined relation and attributes

Figure 5.18 shows the dataset is in ARFF (Attribute-Relation File Format), which is a standard file format used to represent datasets in machine learning and data mining.





- 2) Another terminal “WEKA Explorer” will be popup as shown Figure 5.21. Navigate to “open file” and choose the dataset that has been processed as an .arff file format in order to load the dataset. Click open to proceed.

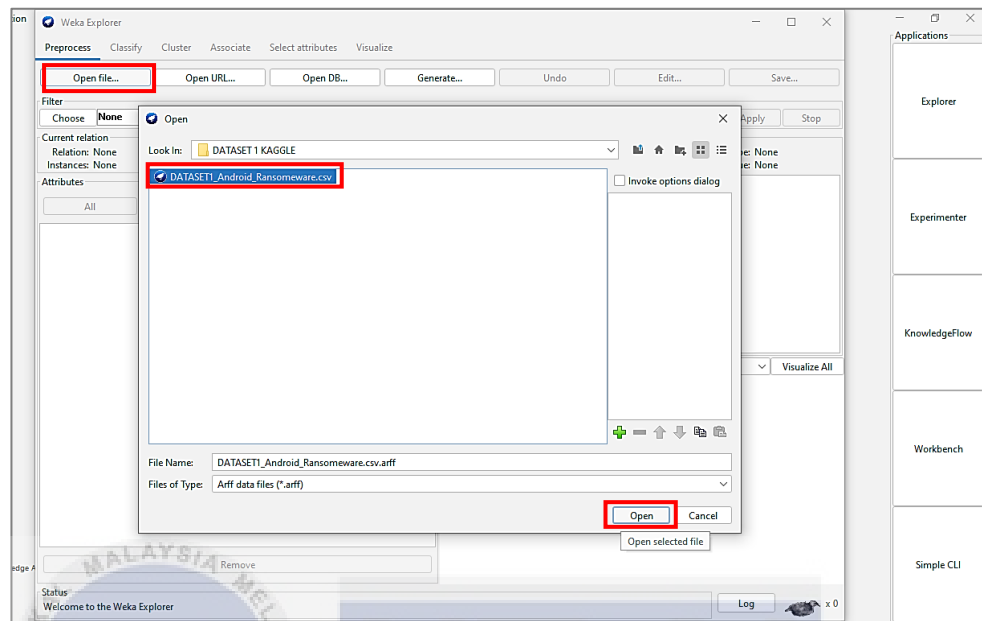


Figure 5.21 Loading the .arff file format dataset into WEKA

- 3) Wait for WEKA to load the dataset as it might take some time especially for dataset with large number of samples/instances. The result is shown as Figure 5.22 below. The graph visualizes the target class which is Label into 2 categories which are Ransomware samples and Benign samples.

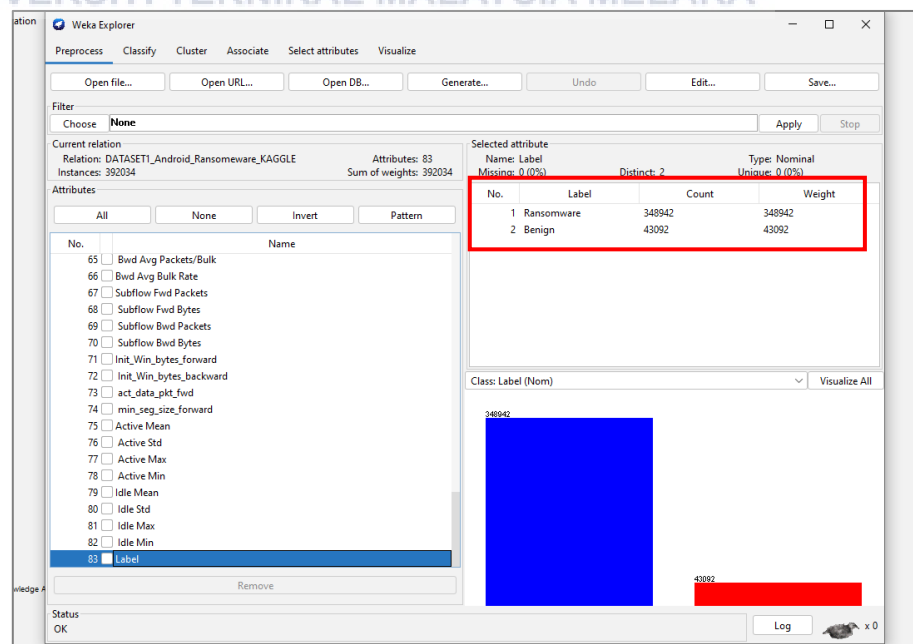


Figure 5.22 graph for visualize the Ransomware and Benign samples



## 5.6.2 Steps to Load Dataset for ORANGE

- 1) Once Orange has been launched, navigate to the menu bar at the left section and choose “File” widget as shown in Figure 5.23 below.

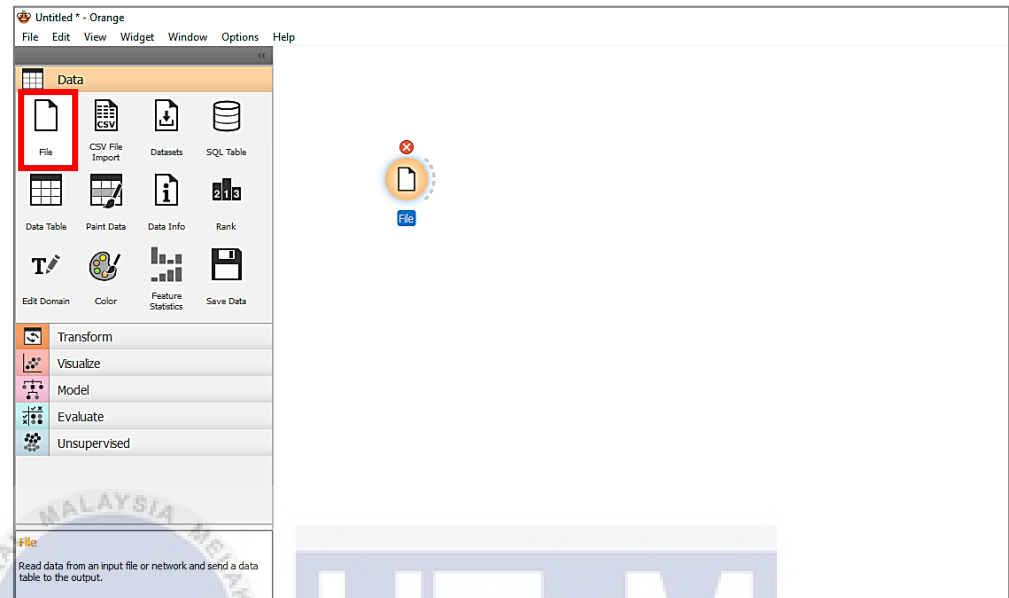


Figure 5.23 Loading dataset into Orange

- 2) Select the dataset that has been processed as an .arff file format from the provided folder. Orange will show all the attributes presented in the chosen dataset and view it using the “Data Table” widget as shown in Figure 5.24 below.

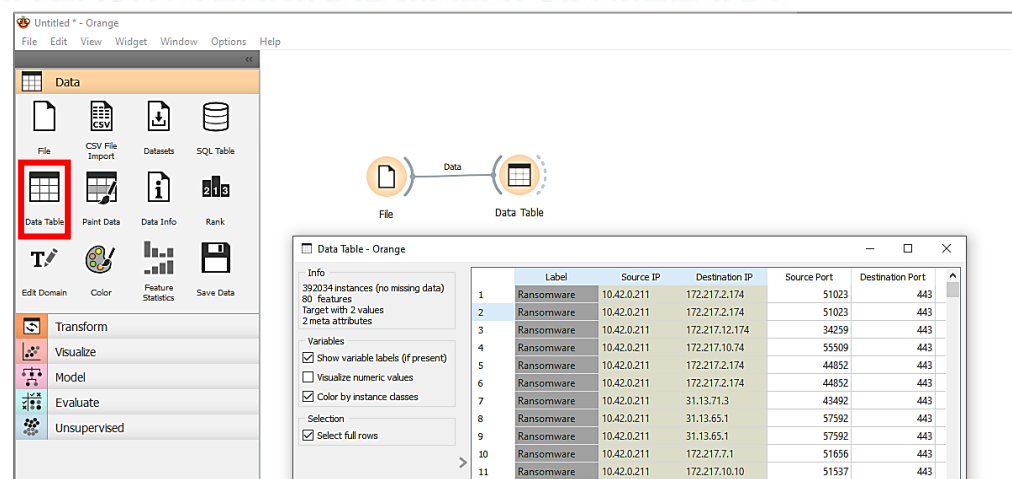


Figure 5.24 Data Table representation in Orange

- 3) In Orange, we have to manually choose which attribute will be our target class. In this case, change the Role value for attribute label as the target class that will be categorized into 2 categories which are Ransomware samples and Benign samples as shown in Figure 5.25 below.

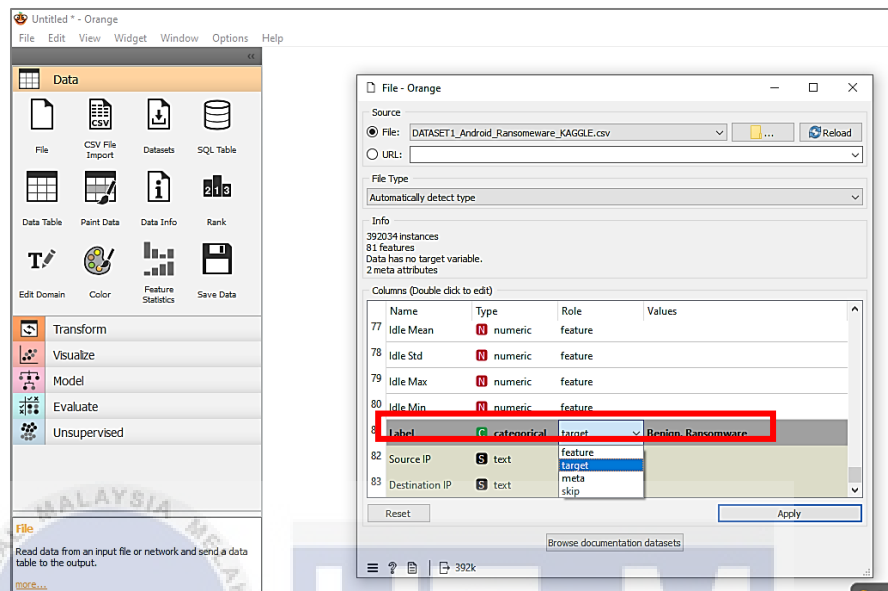


Figure 5.25 Assigning target class in Orange

## 5.7 Step 6: Classification

This step focuses on choosing machine learning classification techniques. According to the literature review in the previous Chapter 3, the selected classification techniques that will be used with the most occurrence are Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes. Below demonstration will show examples of steps for Decision Tree for both WEKA and Orange. Note that the same steps will be applied for the other types of classification techniques. Refer Appendix B for demonstration of Random Forest, Support Vector Machines (SVM) and Naïve Bayes.

### 5.7.1 Steps to classify data in WEKA

- 1) Once the dataset has been loaded, navigate to the “Classify” tab as shown in Figure 5.26 below.

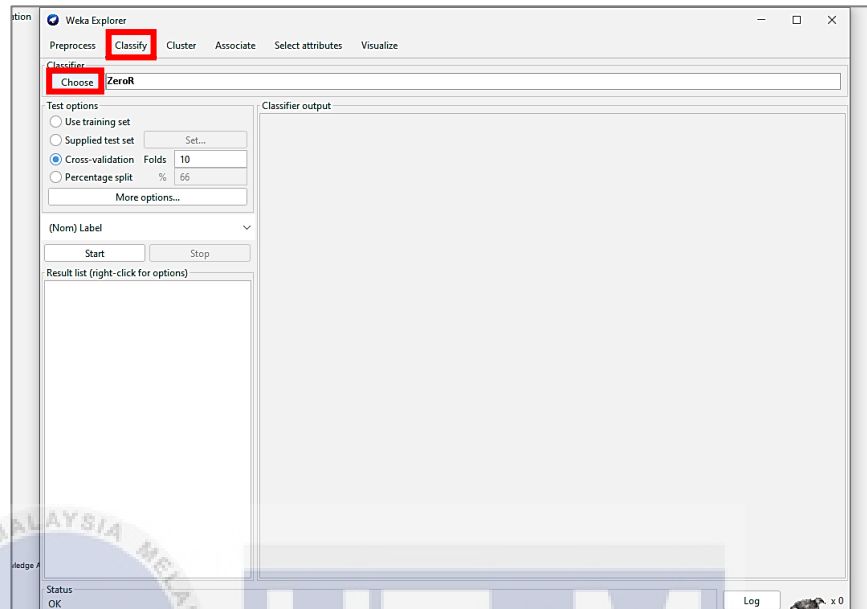


Figure 5.26 WEKA explorer

- 2) Under the “Classifier” section, click choose to select the machine learning classification algorithms as shown in Figure 5.27 below. In this case, we try to classify using one of the decision trees classification algorithms specifically J48.

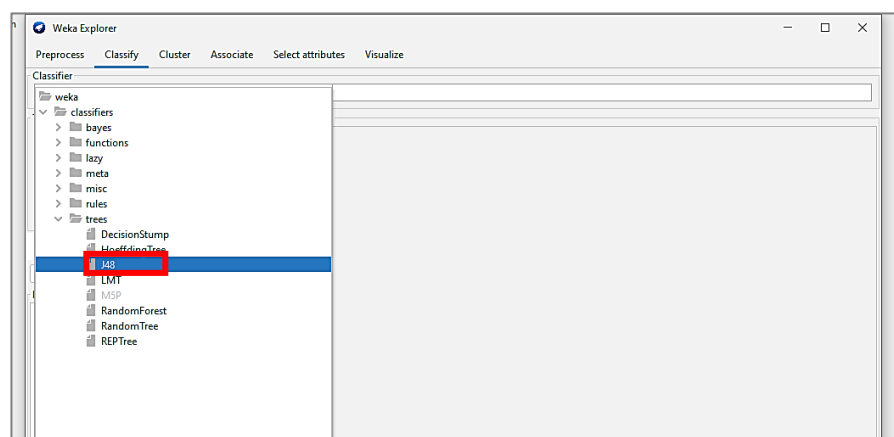


Figure 5.27 Selection of classification algorithms in WEKA

### 5.7.2 Steps to classify data in ORANGE

- 1) To start the classification process on the dataset, navigate to the “Model” section and select the “Tree” widget which represents the Decision Tree classification algorithm as shown in Figure 5.28 below.

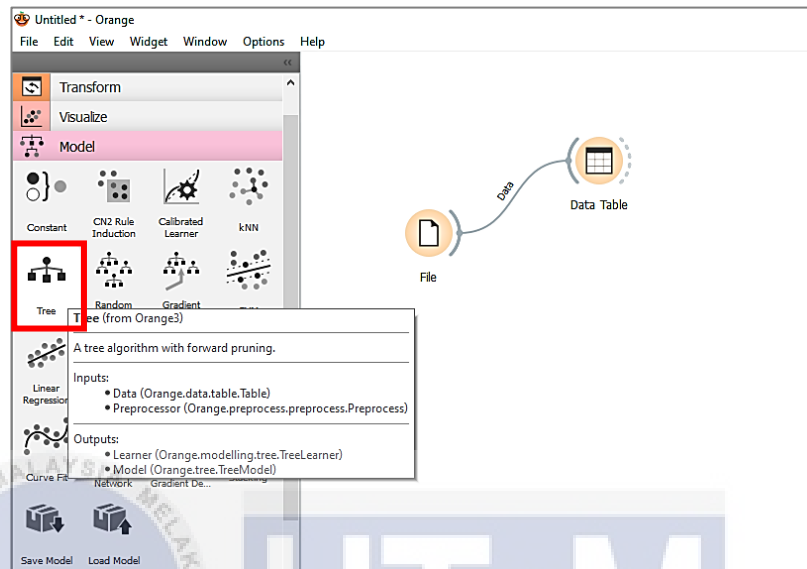


Figure 5.28 Selection of classification algorithms in Orange

- 2) Connect the nodes to start training the model using the Decision Tree classification algorithms as shown in Figure 5.29 below.

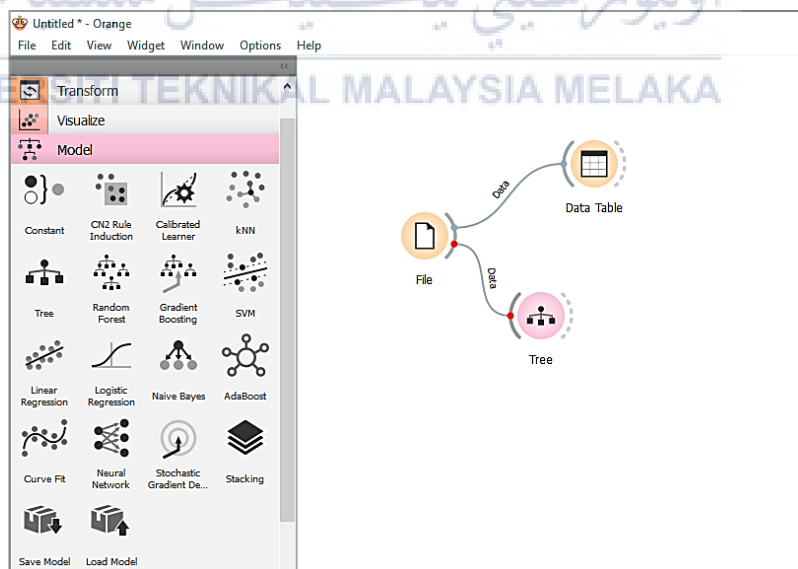


Figure 5.29 Connecting each nodes in Orange

## 5.8 Step 7: Generate Result

This step focuses on generating the result based on the machine learning classification algorithm chosen. The output for this step will be used for the next chapter to evaluate the accuracy result of classification techniques using different evaluation metrics tool.

### 5.8.1 Step 7: Generate Result for WEKA

- 1) Once the classification algorithm has been chosen, for this demonstration we will split the dataset into 70:30 ratio for training and testing the model. Note that the same steps will be repeated for other ratios such as 50:50 and 90:10.

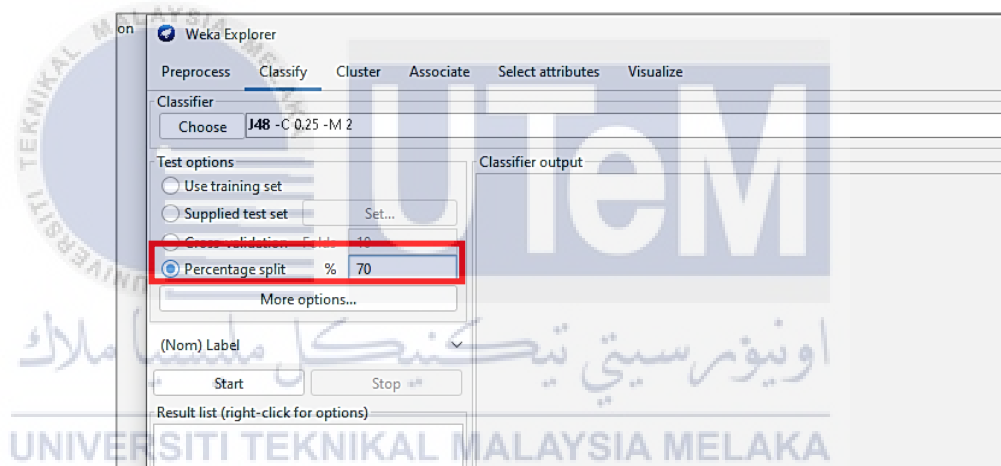


Figure 5.30 Dataset splitting in WEKA

- 2) Click start and wait for WEKA to generate the result as it may take some time. As shown in Figure 5.31 below, the information generated such as the time taken for the model to be build, Summary for each instance, Performance for each evaluation metrics and the Confusion Matrix.

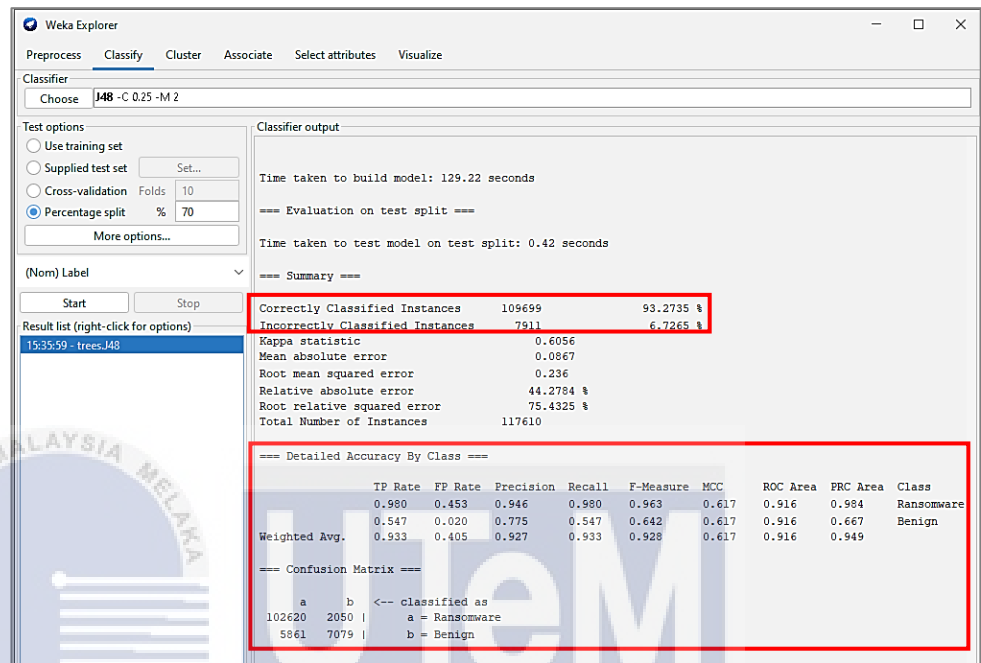


Figure 5.31 Evaluation Metrics result for Decision Tree in WEKA

## 5.8.2 Step 7: Generate Result for ORANGE

- 1) Once the classification algorithm has been chosen, we will evaluate it using the evaluation metrics. Navigate to the “Evaluate” section and connect the node to the “Test and Score” widget as shown in Figure 5.32 below. Click on the Test and Score to edit the configuration of the model and set the training to be 70:30 ratio.

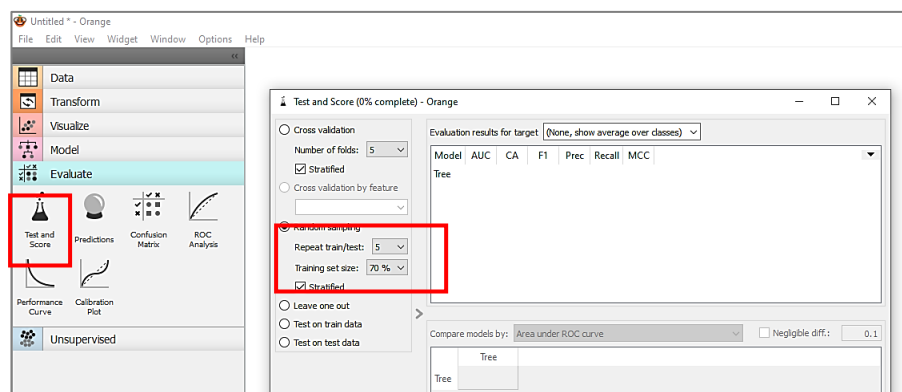


Figure 5.32 Dataset splitting in Orange

- 2) The percentage shows the progress of building and evaluation of the chosen classification model as shown in Figure 5.33 below.

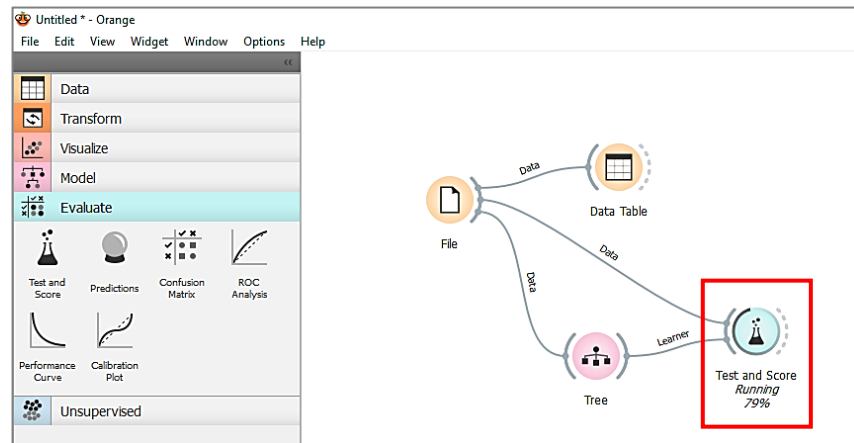


Figure 5.33 Completion percentage of Test and Score process

- 3) After reaching 100% of completion the result will be generated as shown in Figure 5.34 below.

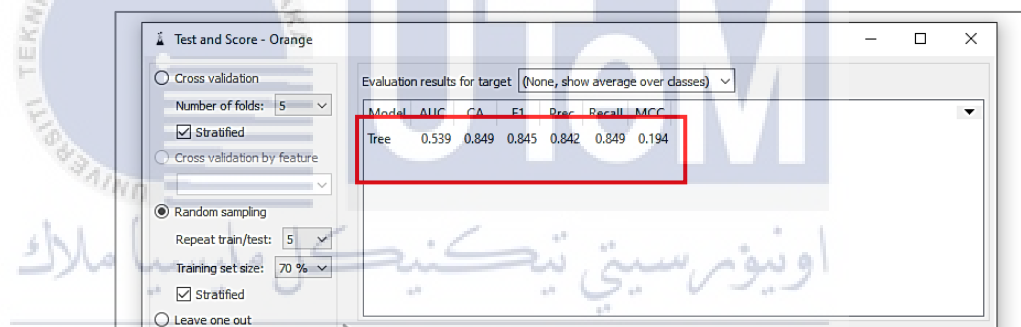


Figure 5.34 Evaluation Metrics result for Decision Tree in Orange

## 5.9 Conclusion

This chapter focuses on the activities to implement the research. Same steps will be taken for other classification algorithm Random Forest, Support Vector Machines (SVM), Naïve Bayes and each dataset splitting ratio 50:50, 70:30, 90:10. The reason for various dataset splitting as discussed in previous Chapter 3 critical review section, which is to improve the previous research work that only using 50:50 ratio. Other than using the method of 2/3 fraction equivalent to 70:30 for splitting the dataset as recommended by (Dobbin and Simon, 2011), we will expand the research by considering various ratios and classification algorithms. Each comparison will be discussed and evaluated in the next Chapter 6.

## CHAPTER 6: TESTING AND EVALUATION

### 6.0 Introduction

The previous chapter focuses on the implementation of the research which involves configuring the machine learning tools to align with the designated algorithms, processing the dataset and starting the learning process by building the machine learning model. This chapter will explain about the testing and analysis after the machine learning model is created. The testing includes a comprehensive evaluation that involves experimenting with diverse dataset testing samples. It's a crucial step as it allows us to analyze the model performance towards different dataset characteristics. Following the experimental results, analysis will be made to evaluate the model performance according to the evaluation metrics. This chapter is crucial to measure the performance of the model and discuss whether different datasets size may influence the accuracy of the model. Chapter 6 activities are mapped based on the research methodology Phase 6 Analyze the information and Phase 7 Documenting Result.



## 6.1 Test Plan Strategy

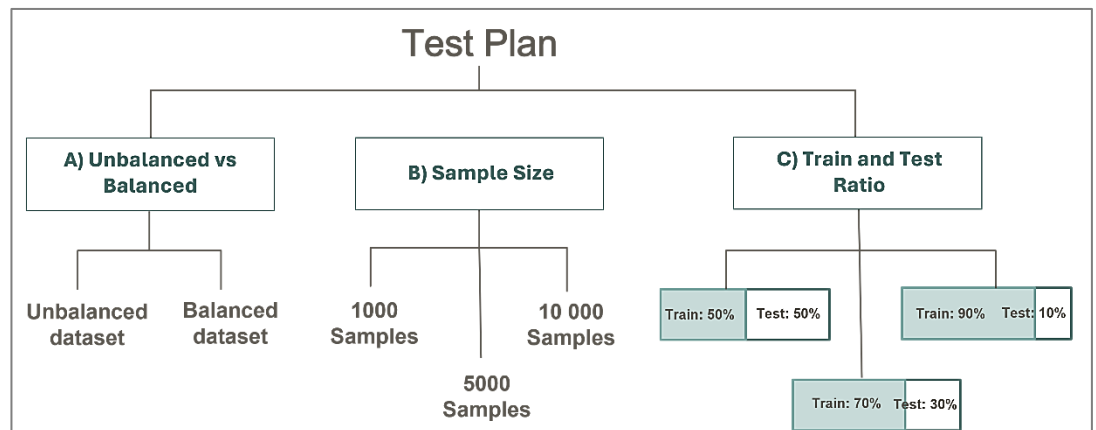


Figure 6.1 Test Plan Strategy

Based on Figure 6.1 we have constructed three test plan with purpose to observe the classification techniques behavior under diverse condition. To be specific, the aspects include testing for a) unbalance vs balanced dataset b) various sample sizes c) various ratio of training and test. Test Plan A is expanded to consider not only the unbalanced dataset but also the balanced dataset which is inspired by (Alsoghyer et al., 2020), (Almomani et al., 2021), and (Mercialdo, 2021). Previous authors only consider 1000 balanced samples precisely 500 Ransomware samples and 500 Benign samples. Therefore, we will improve the previous research by considering to expand the dataset for 1000, 5000 and 10 000 samples. This is because, according to the findings of research by (Ajiboye et al., 2015) in the paper titled "Evaluating The Effect Of Dataset Size On Predictive Model Using Supervised Learning Technique," it is evident that enhancing the results can be achieved through dataset size which motivates our research plan B. For test plan C, as suggested by (Dobbin and Simon, 2011) the recommended amount of training and testing ratio is 70:30. However, an additional ratio was added to the experiment with the purpose of expanding our testing scope.

### 6.1.1 Test Plan Flowchart

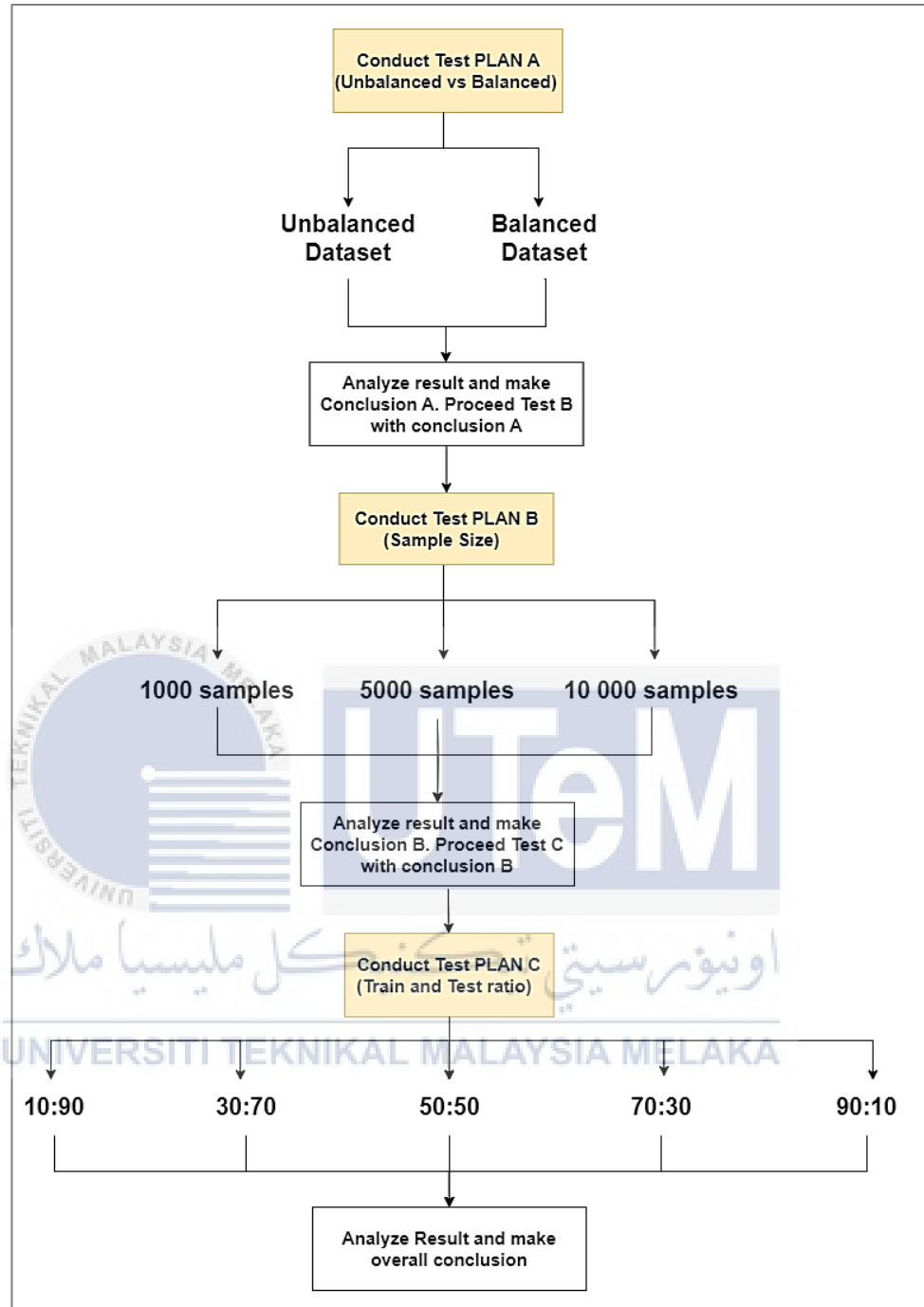
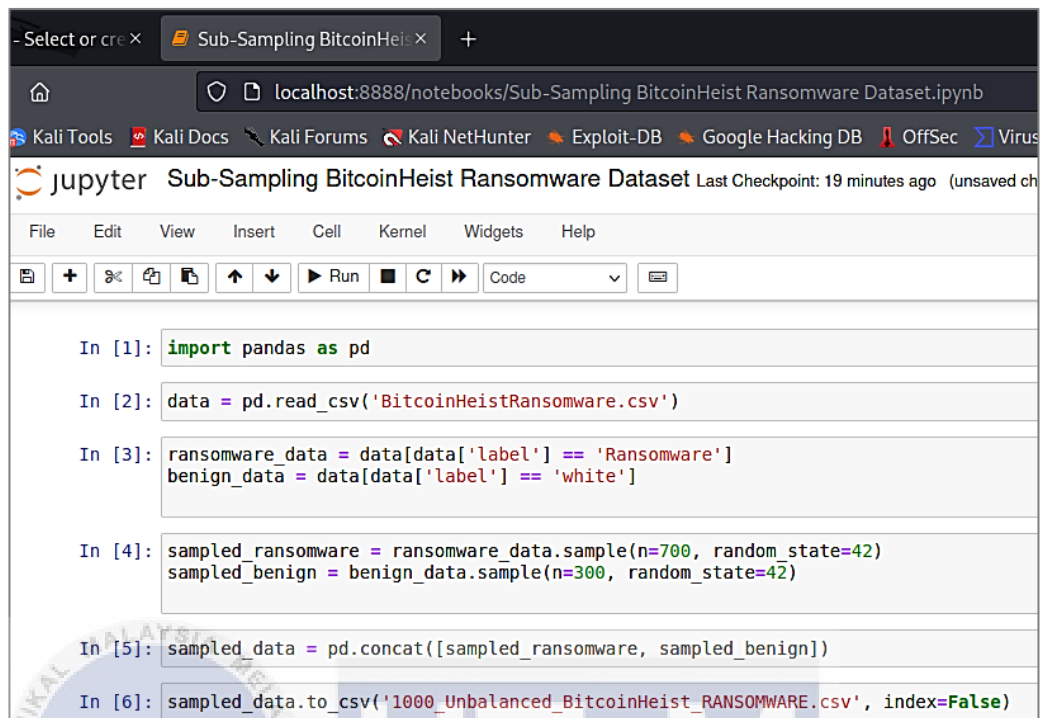


Figure 6.2 Flowchart Conducting Test Plan

The suggested flowchart illustrates an approach for improving the accuracy of machine learning models based on combining ideas from previous research, as discussed in section 6.1. It begins by determining whether a balanced or imbalanced dataset produces greater results and proceed Test Plan B with outcome/conclusion of Test Plan A. Based on conclusion B, we will proceed with Test Plan C and make overall conclusion which is explain in section 6.6 Significant Results.

## 6.1.2 Sub-sampling Ransomware Dataset



```

In [1]: import pandas as pd

In [2]: data = pd.read_csv('BitcoinHeistRansomware.csv')

In [3]: ransomware_data = data[data['label'] == 'Ransomware']
benign_data = data[data['label'] == 'white']

In [4]: sampled_ransomware = ransomware_data.sample(n=700, random_state=42)
sampled_benign = benign_data.sample(n=300, random_state=42)

In [5]: sampled_data = pd.concat([sampled_ransomware, sampled_benign])

In [6]: sampled_data.to_csv('1000_Unbalanced_BitcoinHeist_RANSOMWARE.csv', index=False)

```

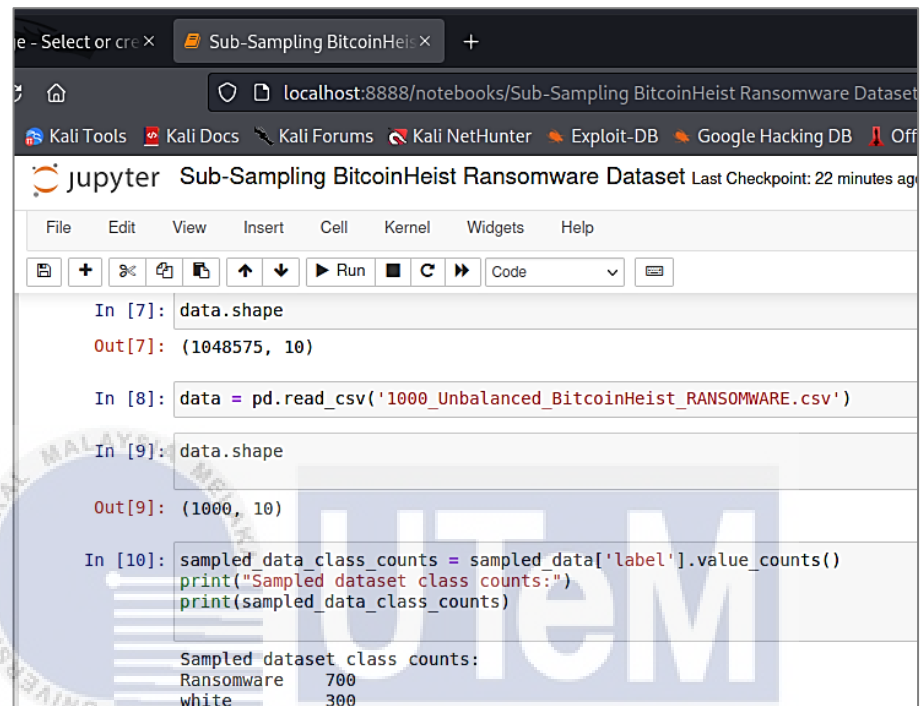
*Figure 6.3 Command for Sub-sampling the Ransomware dataset using Python language in Jupyter Notebook.*

Figure 6.3 shows the command for sub-sampling method for Dataset I. The method is conducted using Python language in Jupyter Notebook. The testing is done in Kali Linux VMWare workstation since it has a better environment for Jupyter Sever configuration setup compared to the Windows environment. In the provided code snippet, the Python panda's library is utilized to handle a dataset named BitcoinHeistRansomware.csv.

This dataset is filtered into two categories: 'Ransomware' and 'white' as the benign sample. Subsequently, a subset is created by randomly sampling 700 instances from the 'Ransomware' class and 300 instances from the 'white' class. The resulting sampled dataset, comprising 1000 instances, is then saved as '1000\_Unbalanced\_BitcoinHeist\_RANSOMWARE\_.csv'. Same code is run by adjuster the parameter for sampled\_ransomware and sample\_benign as the following for Dataset I ( BitcoinHeist Ransomware) :

- a) 1000\_Unbalanced\_BitcoinHeist\_ RANSOMWARE\_.csv
- b) 1000\_Balanced\_BitcoinHeist\_ RANSOMWARE\_.csv

- c) 5000\_Unbalanced\_BitcoinHeist\_RANSOMWARE\_.csv
- d) 5000\_Balanced\_BitcoinHeist\_RANSOMWARE\_.csv
- e) 10 000\_Unbalanced\_BitcoinHeist\_RANSOMWARE\_.csv
- f) 10 000\_Balanced\_BitcoinHeist\_RANSOMWARE\_.csv



```

In [7]: data.shape
Out[7]: (1048575, 10)

In [8]: data = pd.read_csv('1000_Unbalanced_BitcoinHeist_RANSOMWARE.csv')

In [9]: data.shape
Out[9]: (1000, 10)

In [10]: sampled_data_class_counts = sampled_data['label'].value_counts()
print("Sampled dataset class counts:")
print(sampled_data_class_counts)

Sampled dataset class counts:
Ransomware    700
white         300

```

Figure 6.4 Command to show the Ransomware dataset dimension before and after sub-sampling method.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 6.4 shows the command displaying the dimensions for number of rows and columns of the original dataset which comprises of 1048575 million samples with 10 attributes. The purpose of the sub-sampling is to counter against the heap size error when loading the dataset into WEKA. Therefore, this strategy involves working with smaller datasets initially and gradually increasing the size. Line 10 of the Python code shows the result distribution of classes in the newly created balanced subset. Sample of code can be found in Appendix C. Same sub-sampling method is done for all datasets, BitcoinHeist Ransomware dataset (Dataset I), Android Ransomware dataset (Dataset II) and File System Behavior Ransomware Detection dataset (Dataset III). Each parameter has been discussed in the previous Chapter 2 section 2.3.3.2 for further understanding.

## 6.2 Result and Analysis Dataset I

In this section, the results from the project implementation phase are presented. The analysis focuses on identifying the performance of machine learning algorithms by evaluating the accuracy result using different evaluation metrics tools by expanding the test plan. All the results for Dataset I focuses on BitcoinHeist and discussion are shown below. Refer Appendix D for others results.

### 6.2.1 Evaluation Metric Result of 1000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Ransomware Detection

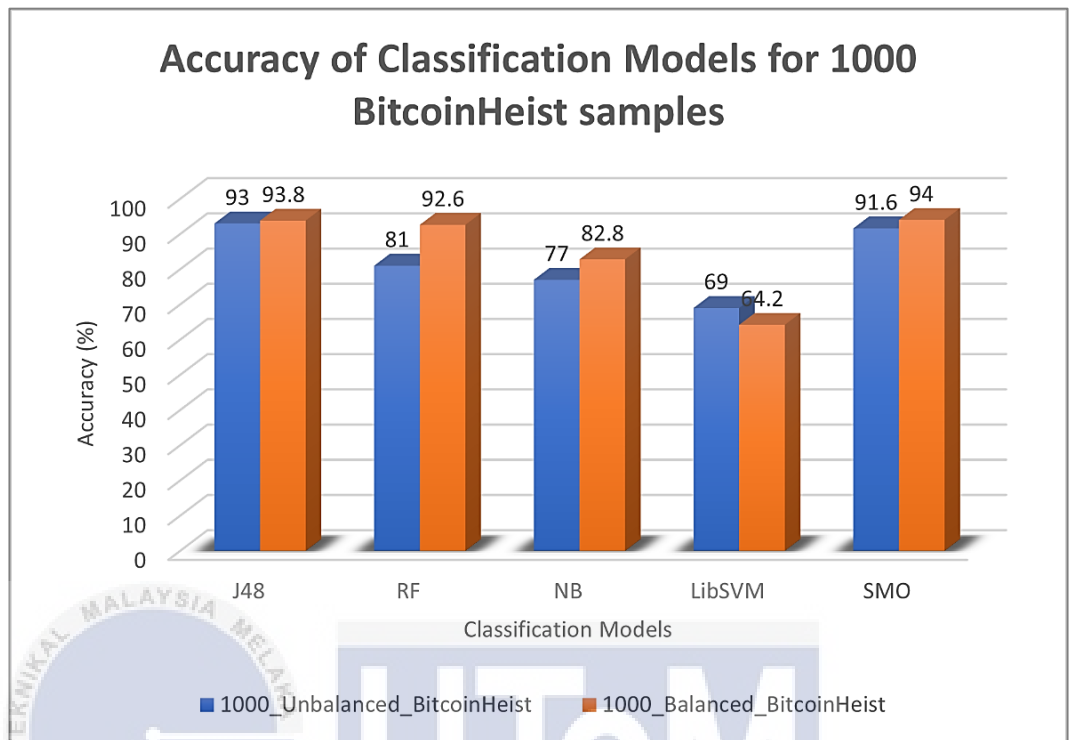
Table 6.1 Evaluation Metrics Result of 1000 BitcoinHeist Samples (50:50)

Dataset	Algorithms	True Positive	False Positive	Precision	Recall	F-Measure	Overall Accuracy
		Rate	Rate				
<b>1000_Unbalanced_BitcoinHeist</b>	Decision Tree	0.930	0.075	0.932	0.930	0.931	93.00%
	Random Forest	0.810	0.424	0.843	0.810	0.781	81.00%
	Naïve Bayes	0.770	0.518	0.820	0.770	0.719	77.00%
	LibSVM	0.690	0.696	0.481	0.690	0.567	69.00%
	SMO	0.916	0.161	0.917	0.916	0.914	91.60%
<b>1000_Balanced_BitcoinHeist</b>	Decision Tree	0.938	0.065	0.940	0.938	0.938	93.80%
	Random Forest	0.926	0.078	0.935	0.926	0.925	92.60%
	Naïve Bayes	0.828	0.165	0.862	0.828	0.825	82.80%
	LibSVM	0.642	0.373	0.730	0.642	0.599	64.20%
	SMO	0.940	0.059	0.941	0.940	0.940	<b>94.00%(Best)</b>

Table 6.1 shows the evaluation metrics result of 1000 BitcoinHeist Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced ransomware samples. 1000\_Unbalanced\_BitcoinHeist Dataset consists of 700 ransomware samples and 300 benign samples (referred to as "white"). Whereas 1000\_Balanced\_BitcoinHeist Dataset consists of 500 ransomware samples and 500 benign samples (referred to as "white"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 1000 samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.1, for the unbalanced scenario, the Decision Tree achieved the highest True Positive Ratio among all classifiers which is 0.930. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.932. The False Positive is relatively lowest among all at 0.075% suggesting that it is classifying non-ransomware (white) instances with a high level of accuracy which is 93.00 %.

For the case of balanced scenario, it shows that the Support Vector Machine with Polynomial Kernel (SMO) is the best classifier with highest accuracy 94.00%. It achieved the highest True Positive Ratio among all classifiers which is 0.940 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.059 suggesting that it is classifying non-ransomware (white) instances with a high level of precision of 0.941.



*Figure 6.5 Graph Accuracy for 1000 BitcoinHeist Samples (50:50)*

Figure 6.5 shows the Graph Accuracy for 1000 BitcoinHeist Samples for both cases. The graph is important to provide a clear visual representation of the model performance between the unbalanced and balanced scenarios. The graph illustrates a noticeable increase in accuracy across four models, specifically Decision Tree J48, Random Forest, Naïve Bayes, and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario. As conclusion machine learning algorithms perform better across multiple metrics in the balanced case, where the class distribution is equal. This is because a balanced dataset gives classifiers a more equal representation of both classes, allowing the model to learn and classify more successfully.

## 6.2.2 Evaluation Metric Result of 5000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Dataset Ransomware Detection

Table 6.2 Evaluation Metrics Result of 5000 BitcoinHeist Samples (50:50)

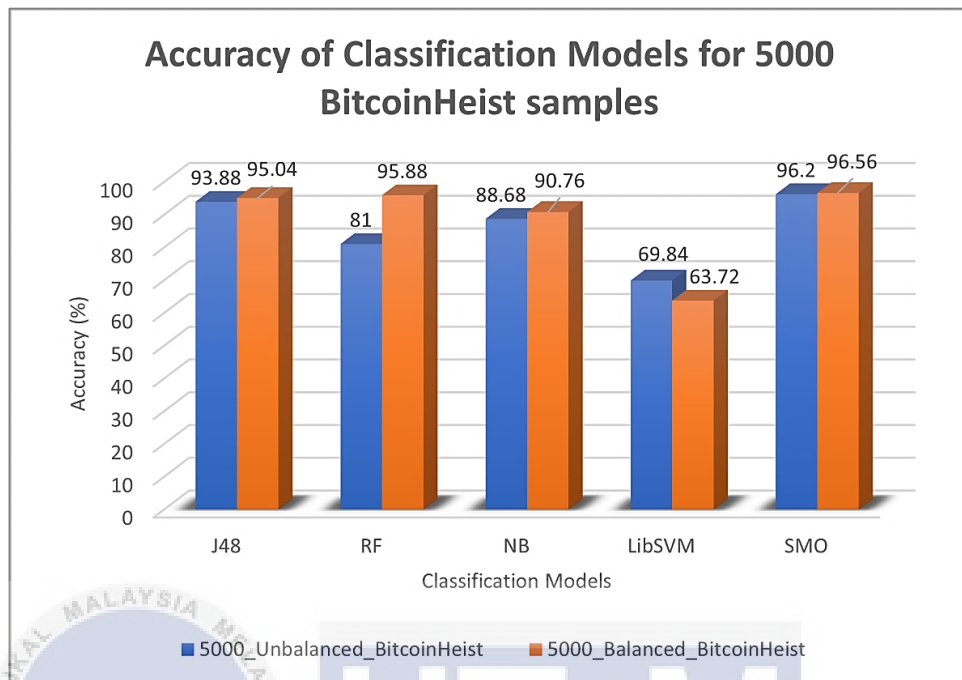
Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>5000_Unbalanced_BitcoinHeist</b>	Decision Tree	0.939	0.045	0.944	0.939	0.940	93.88 %
	Random Forest	0.810	0.430	0.847	0.810	0.779	81.00%
	Naïve Bayes	0.887	0.239	0.895	0.887	0.880	88.68%
	LibSVM	0.698	0.667	0.662	0.698	0.596	69.84%
	SMO	0.962	0.074	0.962	0.962	0.962	96.2%
<b>5000_Balanced_BitcoinHeist</b>	Decision Tree	0.950	0.051	0.952	0.950	0.950	95.04%
	Random Forest	0.959	0.043	0.962	0.959	0.959	95.88%
	Naïve Bayes	0.908	0.092	0.908	0.908	0.908	90.76%
	LibSVM	0.637	0.373	0.691	0.637	0.606	63.72%
	SMO	0.966	0.034	0.966	0.966	0.966	<b>96.56%(Best)</b>



Table 6.2 shows the evaluation metrics result of 5000 BitcoinHeist Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced ransomware samples. 5000\_Unbalanced\_BitcoinHeist Dataset consists of 3500 ransomware samples and 1500 benign samples (referred to as "white"). Whereas 5000\_Balanced\_BitcoinHeist Dataset consists of 2500 ransomware samples and 2500 benign samples (referred to as "white"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 5000 samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.2, for the unbalanced scenario, Support Vector Machine with Polynomial Kernel (SMO) achieved the highest True Positive Ratio among all classifiers which is 0.962. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.962 as well. Despite the False Positive for SMO being the second lowest among all at 0.074, it has the highest Precision, Recall and F-measure value at 0.962 which contributes to its being the highest accuracy of 96.2% among all models.

For the case of the balanced scenario, it is noticeable that the Support Vector Machine with Polynomial Kernel (SMO) maintains its position as the most effective classifier, reaching the accuracy of 96.56. It achieved the highest True Positive Ratio among all classifiers which is 0.966 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.034 suggesting that it is classifying non-ransomware (white) instances with a high level of precision of 0.966.



*Figure 6.6 Graph Accuracy for 5000 BitcoinHeist Samples (50:50)*

Figure 6.6 shows the Graph Accuracy for 5000 BitcoinHeist Samples for both cases. The graph illustrates a noticeable increase in accuracy across four models, specifically Decision Tree J48, Random Forest, Naïve Bayes, and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario. It can be highlight that the performance of Naïve Bayes has significantly increase for 5000 samples (88.68% and 90.76%) compared to the previous 1000 samples (77% and 82.8%). This shows that the Naïve Bayes model performance increases as the sample size increase to 5000 samples. Whereas for three others, Decision Tree J48, Random Forest, SMO maintain its good performance with slight increments in accuracy. As conclusion most of machine learning algorithms which are Decision Tree J48, Random Forest, Naïve Bayes, and SMO perform better across multiple metrics in the balanced case, where the class distribution is equal. This is because a balanced dataset gives classifiers a more equal representation of both classes, allowing the model to learn and classify more successfully.

### 6.2.3 Evaluation Metric Result of 10 000 BitcoinHeist Ransomware Samples for Unbalanced and Balanced Dataset Ransomware Detection

Table 6.3 Evaluation Metrics Result of 10 000 BitcoinHeist Samples (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>10 000_Unbalanced_BitcoinHeist</b>	Decision Tree	0.941	0.032	0.949	0.941	0.942	94.06%
	Random Forest	0.775	0.521	0.830	0.775	0.723	77.46%
	Naïve Bayes	0.912	0.142	0.912	0.912	0.911	91.24%
	LibSVM	0.701	0.665	0.661	0.701	0.603	70.14%
	SMO	0.972	0.053	0.972	0.972	0.972	97.18%
<b>10 000_Balanced_BitcoinHeist</b>	Decision Tree	0.952	0.047	0.954	0.952	0.952	95.2%
	Random Forest	0.965	0.035	0.967	0.965	0.965	96.46%
	Naïve Bayes	0.548	0.460	0.713	0.548	0.436	54.84%
	LibSVM	0.650	0.346	0.704	0.650	0.626	64.98%
	SMO	0.982	0.018	0.982	0.982	0.982	<b>98.16%(Best)</b>

Table 6.3 shows the evaluation metrics result of 10 000 BitcoinHeist Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced ransomware samples. 10 000\_Unbalanced\_BitcoinHeist Dataset consists of 7000 ransomware samples and 3000 benign samples (referred to as "white"). Whereas 10 000\_Balanced\_BitcoinHeist Dataset consists of 5000 ransomware samples and 5000 benign samples (referred to as "white"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 10 000 samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.3, for the unbalanced scenario, Support Vector Machine with Polynomial Kernel (SMO) still maintains its position as the most effective classifier, with the highest True Positive Ratio among all classifiers which is 0.972. The result indicated its effectiveness to correctly identified ransomware samples. Despite the False Positive for SMO being the second lowest among all at 0.053, it has the highest Precision, Recall and F-measure value at 0.972 which contributes to its being the highest accuracy of 97.18% among all models.

For the case of the balanced scenario, it is noticeable that the Support Vector Machine with Polynomial Kernel (SMO) remains its position as the most effective classifier, reaching the greatest accuracy of 98.16%. It achieved the highest True Positive Ratio among all classifiers which is 0.982 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.018 suggesting that it is classifying non-ransomware (white) instances with a highest level of precision at 0.982.

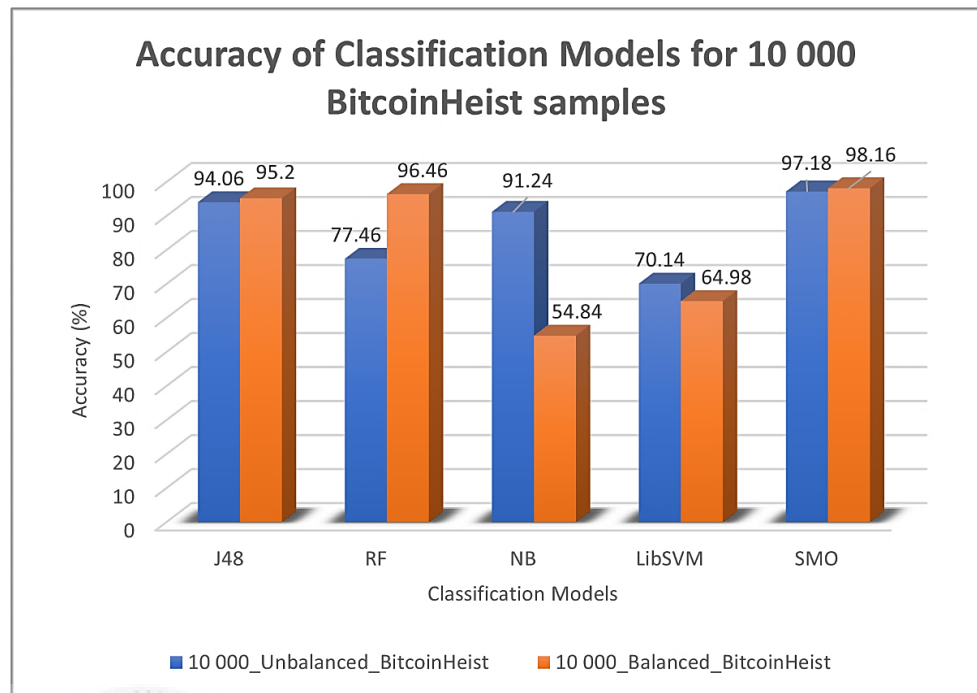


Figure 6.7 Graph Accuracy for 10 000 BitcoinHeist Samples (50:50)

Figure 6.7 shows the Graph Accuracy for 10 000 BitcoinHeist Samples for both cases. The graph illustrates a noticeable increase in accuracy across three models, specifically Decision Tree J48, Random Forest and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario. On the other hand, Naïve Bayes and LibSVM shows inconsistency of result as the sample size increase from 5000 to 10 000 samples. Whereas for three others, Decision Tree J48, Random Forest, SMO maintain its good performance with slight increments in accuracy.

As conclusion most of machine learning algorithms which are Decision Tree J48, Random Forest, Naïve Bayes, and SMO perform better across multiple metrics in the balanced scenario, where the class distribution is equal. This is because a balanced dataset gives classifiers a more equal representation of both classes, allowing the model to learn and classify more successfully. It also can be concluded that, the Support Vector Machine with Polynomial Kernel (SMO) has achieved the best performance for all sample size 1000, 5000 and 10 000 BitcoinHeist samples.

### 6.2.4 Accuracy of Classification Model Across Different Sample Sizes

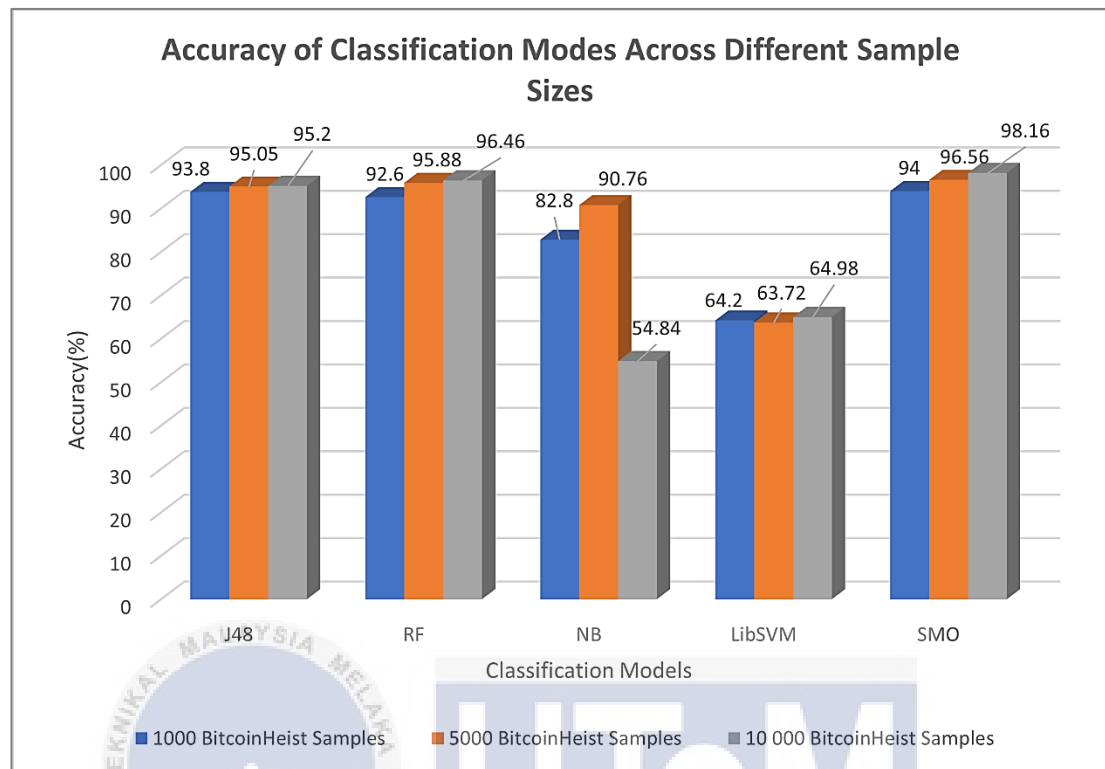
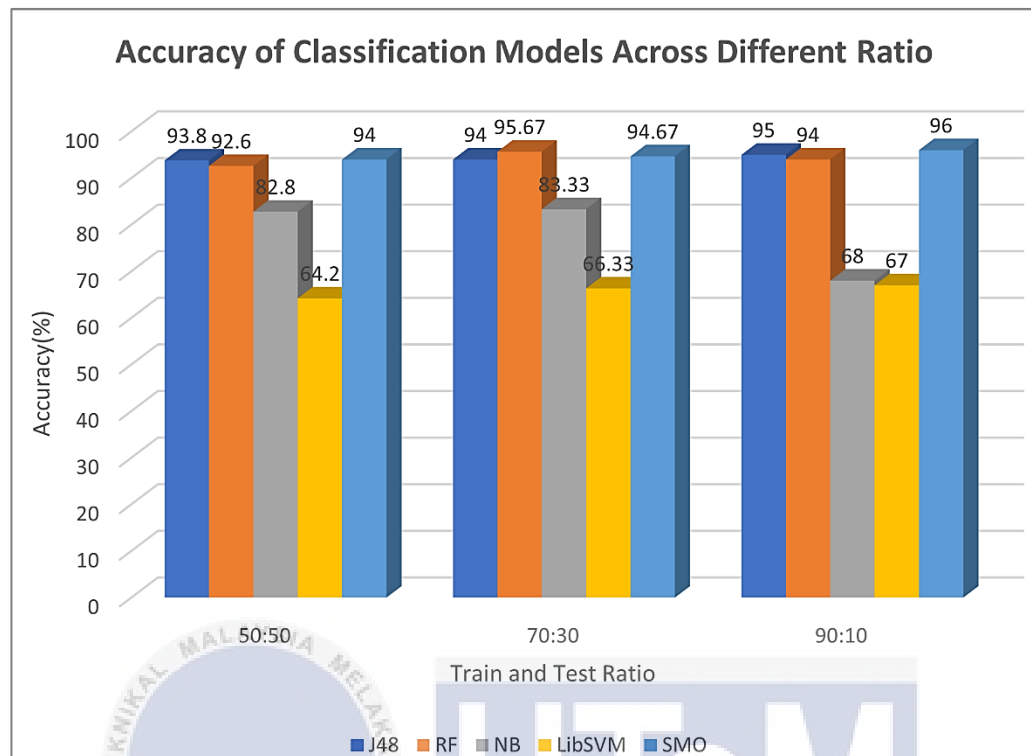


Figure 6.8 Graph Accuracy for Classification Model Across Different Sample Sizes of BitcoinHeist Ransomware (50:50)

Figure 6.8 shows the Graph Accuracy for Classification Model Across Different Sample Sizes of BitcoinHeist Ransomware. The results allow us to visualize the behavior of classification models for the sample of 1000, 5000 and 10 000 by doing the sub-sampling method to prevent heap size issues in WEKA. Based on the results, as the samples increases from 1000 to 5000 and then to 10 000, Decision Tree J48 Random Forest and Support Vector Machine with Polynomial Kernel (SMO) display an incremental rise in accuracy. To be specific Decision Tree J48 shifted from 93.8% up to 95.05% and peaked at 95.20%. Random Forest on the other hand, started with 92.60% increase up to 95.88% and peaked at 96.46%. It can be seen that, Naïve Bayes and LibSVM displays sensitivity to class distributions with poor performance. In this test plan, the Support Vector Machine with Polynomial Kernel (SMO) performed best compared to others starting with 94.00% increase up to 96.56% and peaked at 98.16%.

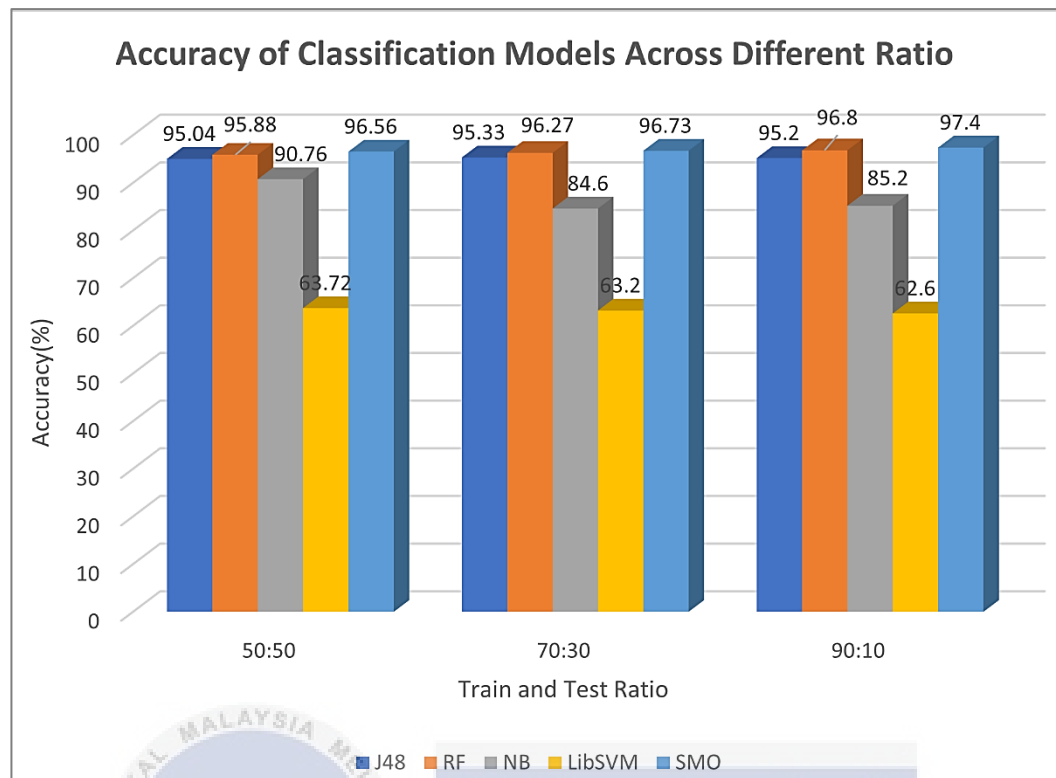
### 6.2.5 Accuracy of Classification Model for Different Ratio



*Figure 6.9 Graph Accuracy for 1000 BitcoinHeist Samples across Different Ratio*

Figure 6.9 shows the Graph Accuracy for 1000 BitcoinHeist samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, both Decision Tree J48 and Support Vector Machine with Polynomial Kernel (SMO) display an incremental rise in accuracy. To be specific Decision Tree J48 shifted from 93.8% up to 94% and peaked at 95%. Whereas SMO experienced a slight increase from 94% up to 94.67% and peaked at 96%. On the other hand, LibSVM displays sensitivity to class distributions with limited improvements as the performance are all below 70% of accuracy. While it's true that both Random Forest (RF) and Naïve Bayes (NB) experience increase in accuracy when transitioning from 50:50 to 70:30 ratio. However, decrease in accuracy can be seen when moving to the 90:10 ratio, which Random Forest and Naïve Bayes scored down to 94% and 68% respectively.

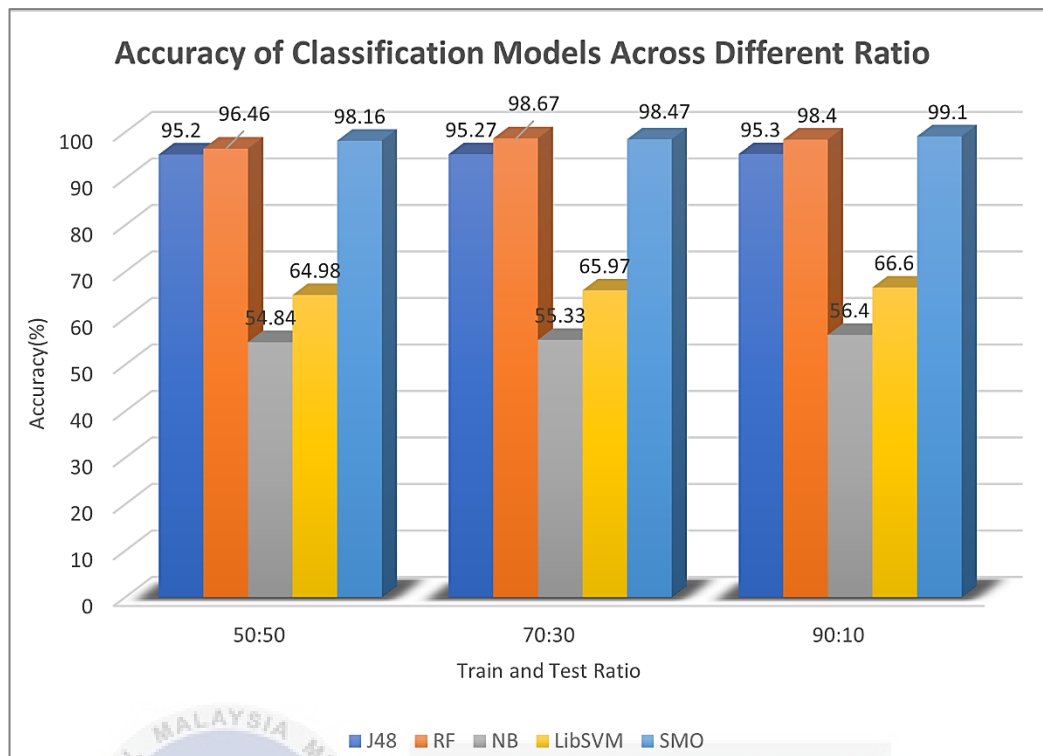




*Figure 6.10 Graph Accuracy for 5000 BitcoinHeist Samples across Different Ratio*

Figure 6.9 shows the Graph Accuracy for 5000 BitcoinHeist samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the result, the SMO model regularly outperforms in all class ratios. It can be highlighted that Support Vector Machine with Polynomial Kernel (SMO) consistently leads with the highest accuracy levels across all ratios. On the other hand, the performance of Naïve Bayes (NB) and LibSVM display sensitivity to class distributions. Their accuracy shows limited improvements with Naïve Bayes decreasing for each ratio and LibSVM performance are all below 70% of accuracy. Whereas both the Decision Tree J48 and Random Forest models improve in accuracy as the 50:50 ratio is changed to 70:30. Decision Tree J48 improved from 95.04% to 95.33%, while Random Forest improved from 95.88% to 96.27%. However, a slight decrease in accuracy can be seen when moving to the 90:10 ratio, with Decision Tree J48 scoring down to 95.20%.





*Figure 6.11 Graph Accuracy for 10 000 BitcoinHeist Samples across Different Ratio*

Figure 6.11 shows the Graph Accuracy for 10 000 BitcoinHeist samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the result, the SMO model regularly outperforms in all class ratios. Specifically, SMO achieves a 98.16% accuracy at a 50:50 ratio. This pattern continues with an accuracy of 98.47% at a 70:30 ratio and peaks at 99.1% at a 90:10 ratio. On the other hand, the performance of Naïve Bayes (NB) and LibSVM display sensitivity to class distributions, with their accuracy showing limited improvements which are all below 70% of accuracy. Both Decision Tree J48 and Random Forest models exhibit an increase in accuracy as the transition progresses from the 50:50 ratio to the 70:30 ratio. However, a slight decrease in accuracy is observed when moving to the 90:10 ratio. Consequently, considering the accuracy trends observed in all samples (1000, 5000 and 10 000) it can be concluded that, most machine learning classification model achieve better performance for testing ratio starting with 70% and above.

## 6.2.6 Comparison between WEKA and Orange

Table 6.4 Summary of results for Dataset I TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.940	0.059	0.940	0.940	0.940	94.00%
	Random Forest	0.957	0.046	0.957	0.957	0.957	<b>95.67% (Best)</b>
	Naïve Bayes	0.833	0.144	0.877	0.833	0.831	83.33%
	LibSVM	0.663	0.381	0.732	0.663	0.626	66.33%
	SMO	0.947	0.048	0.950	0.947	0.947	94.67%
<b>5000</b>	Decision Tree	0.953	0.047	0.955	0.953	0.953	95.33%
	Random Forest	0.963	0.037	0.965	0.963	0.963	96.27%
	Naïve Bayes	0.846	0.154	0.876	0.846	0.843	84.60%
	LibSVM	0.632	0.368	0.680	0.632	0.606	63.20%
	SMO	0.967	0.033	0.968	0.967	0.967	<b>96.73% (Best)</b>
<b>10 000</b>	Decision Tree	0.953	0.048	0.954	0.953	0.953	95.27%
	Random Forest	0.987	0.014	0.987	0.987	0.987	<b>98.67% (Best)</b>
	Naïve Bayes	0.553	0.440	0.716	0.553	0.453	55.33%
	LibSVM	0.660	0.344	0.709	0.660	0.637	65.97%
	SMO	0.985	0.015	0.985	0.985	0.985	98.47%

Table 6.4 shows an in-depth overview of the evaluation's results in WEKA obtained by using balanced BitcoinHeist datasets with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, Random Forest obtains the highest True Positive Rate (TPR) of 0.957 and the lowest False Positive Rate (FPR) of 0.046 with 1000 samples. These numbers represent the algorithm's ability to detect positive cases correctly while minimizing false positives. As the dataset size increases to 5000 samples, Random Forest experienced a slight increase from 95.67% to 96.27% and peaked at 98.67% for 10 000 BitcoinHeist samples as the best classification algorithms.

It can be concluded that, for WEKA Random Forest (RF) approach consistently outperforms other classification algorithms on balanced BitcoinHeist datasets with variable sample sizes and a 70:30 ratio. With accuracy scores of 95.67%, 96.27%, and 98.67% for 1000, 5000, and 10,000 samples, respectively remains the best classification algorithms. Following closely behind, the Support Vector Machine with Polynomial Kernel (SMO) performs successfully, with the highest accuracy of 96.73% for 5000 samples. These findings highlight the reliability of Random Forest and the flexibility of SMO for effective ransomware detection over an extensive selection of dataset sizes.

Table 6.5 Summary of results for Dataset I TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.908	0.088	0.910	0.910	0.910	91.00%
	Random Forest	0.892	0.044	0.926	0.924	0.924	<b>92.40% (Best)</b>
	Naïve Bayes	0.892	0.142	0.875	0.875	0.875	87.50%
	SVM	0.802	0.026	0.900	0.887	0.887	88.70%
<b>5000</b>	Decision Tree	0.940	0.045	0.948	0.948	0.948	94.80%
	Random Forest	0.933	0.030	0.952	0.951	0.951	<b>95.10% (Best)</b>
	Naïve Bayes	0.910	0.185	0.866	0.862	0.862	86.20%
	SVM	0.693	0.089	0.817	0.802	0.800	80.20%
<b>10 000</b>	Decision Tree	0.936	0.046	0.945	0.945	0.945	94.50%
	Random Forest	0.936	0.028	0.955	0.955	0.955	<b>95.50% (Best)</b>
	Naïve Bayes	0.912	0.183	0.868	0.864	0.864	86.40%
	SVM	0.619	0.073	0.802	0.773	0.767	77.30%

Table 6.5 shows an in-depth overview of the evaluation's results in ORANE obtained by using balanced BitcoinHeist datasets with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, the ORANGE examination of balanced BitcoinHeist datasets with a 70:30 ratio across three sample sizes demonstrates the consistent increase of the Random Forest classifier performance. Its accuracy, in particular, gradually rises by around at the beginning at 92.40% for 1000 samples, progressing to 95.10% for 5000 samples, and finally peaked at 95.50% accuracy for 10,000 samples. This rise in accuracy reflects its ability to consistently perform accurate classification as the dataset size grows.

The Decision Tree classifier follows in second best performance in ORANGE closely followed by the Random Forest, and its accuracy shows a similar rising trend across various sample sizes. It begins with an accuracy of 91.00% for 1000 samples, gradually increases to 94.80% for 5000 samples, and then maintains a little lower accuracy of 94.50% for 10,000 samples. This pattern shows a continuous 3.8% increase from the smallest to the largest sample. This development emphasizes the Decision Tree's ability to make accurate predictions. In conclusion, Random Forest performance outperform all others in ORANGE and followed by the Decision Tree.

### 6.3 Result and Analysis Dataset II

In this section, the results from the project implementation phase are presented. All the results for Dataset II focuses on Android Ransomware and discussion are shown below:

#### 6.3.1 Evaluation Metrics Result of 1000 Android Samples for Unbalanced and Balanced Ransomware Detection

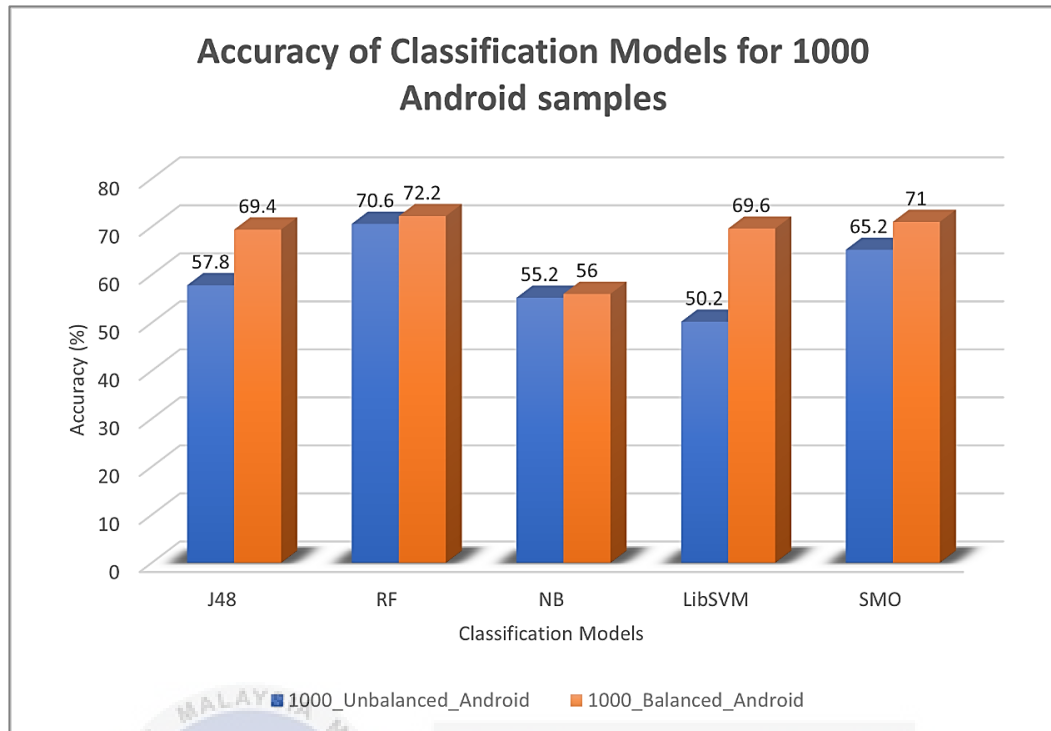
Table 6.6 Evaluation Metrics Result of 1000 Android Samples (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure	Overall Accuracy
<b>1000_Unbalanced_ Android</b>	Decision Tree	0.578	0.437	0.615	0.578	0.532	57.80%
	Random Forest	0.706	0.296	0.706	0.706	0.706	70.60%
	Naïve Bayes	0.552	0.466	0.539	0.552	0.481	55.20%
	LibSVM	0.502	0.476	0.592	0.502	0.369	50.20%
	SMO	0.652	0.352	0.654	0.652	0.649	65.20%
<b>1000_Balanced_ Android</b>	Decision Tree	0.694	0.694	0.694	0.694	0.694	69.40%
	Random Forest	0.722	0.550	0.700	0.722	0.676	<b>72.20% (Best)</b>
	Naïve Bayes	0.560	0.344	0.676	0.560	0.574	56.00%
	LibSVM	0.696	0.671	0.652	0.696	0.590	69.60%
	SMO	0.710	0.614	0.688	0.710	0.637	71.00%

Table 6.6 shows the evaluation metrics result of 1000 Android Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced Android Ransomware samples. 1000\_Unbalanced\_Android Dataset consists of 700 ransomware samples and 300 benign samples (referred to as "Benign"). Whereas 1000\_Balanced\_Android Dataset consists of 500 ransomware samples and 500 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 1000 Android samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.6, for the unbalanced scenario, the Random Forest achieved the highest True Positive Ratio among all classifiers which is 0.706. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.706. The False Positive is relatively lowest among all at 0.296 suggesting that it is classifying non-ransomware (Benign) instances with a high level of accuracy which is 70.60 %.

For the case of balanced scenario, Random Forest maintains its good performance as the best classifier with highest accuracy 72.20%. It achieved the highest True Positive Ratio among all classifiers which is 0.722 which indicated its effectiveness to correctly identified ransomware sample. Despite the False Positive being the second lowest among all at 0.550, it has the highest Precision and Recall values which contributes to its being the highest accuracy of 72.20% among all models.



*Figure 6.12 Graph Accuracy for 1000 Android Samples (50:50)*

Figure 6.12 shows the Graph Accuracy for 1000 Android Samples for both cases. The tabular result has been transformed into a graph which is important to provide a clear visual representation of the model performance between the unbalanced and balanced scenarios. The graph illustrates a noticeable increase in accuracy across all five models, specifically Decision Tree J48, Random Forest, Naïve Bayes, LibSVM and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario.

To be specific, a significant improvement in accuracy can be seen for Decision Tree J48 and LibSVM which are 57.80% to 69.40% and 50.20% to 69.60%. On the other hand, support Vector Machine with Polynomial Kernel (SMO) experience increased in accuracy by 5.8%. Both Random Forest and Naïve Bayes experience slight increase in accuracy 1.6% and 0.8% respectively.

As conclusion machine learning algorithms perform better across multiple metrics in the balanced case, where the class distribution is equal for all classification model of 1000 Android samples. This is because a balanced dataset gives classifiers a more equal representation of both classes, allowing the model to learn and classify more successfully.



### 6.3.2 Evaluation Metric Result of 5000 Android Samples for Unbalanced and Balanced Dataset Ransomware Detection

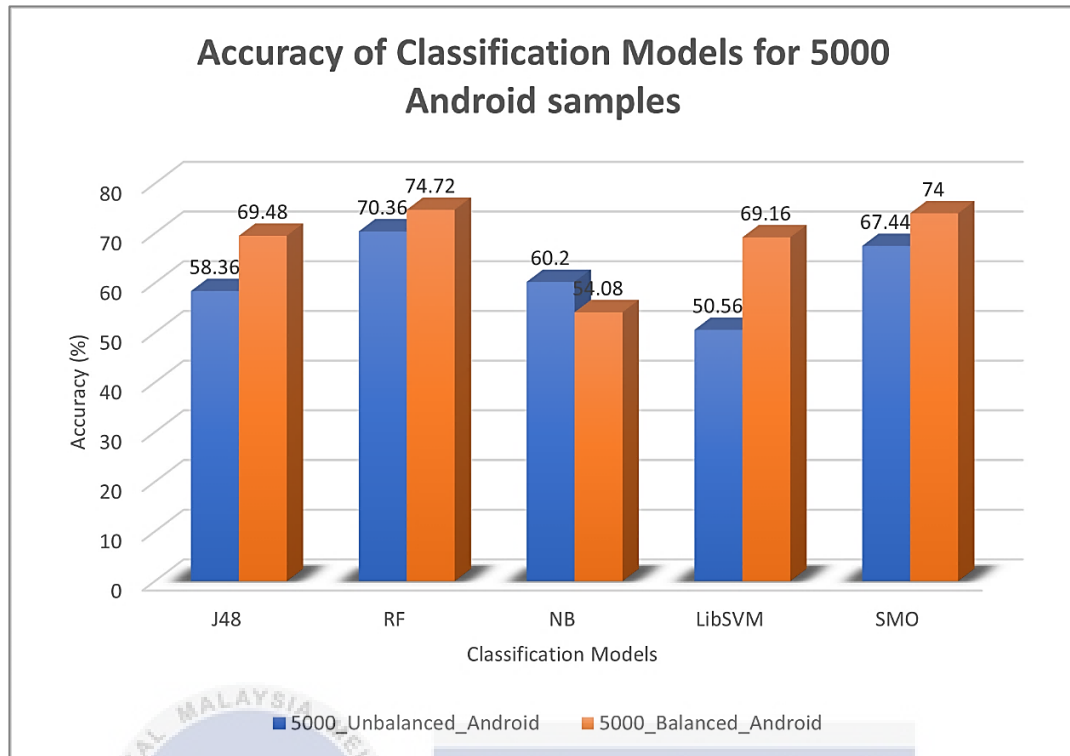
Table 6.7 Evaluation Metrics Result of 5000 Android Samples (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure	Overall Accuracy
<b>5000_Unbalanced_ Android</b>	Decision Tree	0.584	0.421	0.587	0.584	0.577	58.36%
	Random Forest	0.704	0.299	0.706	0.704	0.702	70.36%
	Naïve Bayes	0.602	0.409	0.663	0.602	0.556	60.2%
	LibSVM	0.506	0.476	0.557	0.506	0.390	50.56%
	SMO	0.674	0.328	0.676	0.674	0.673	67.44%
<b>5000_Balanced_ Android</b>	Decision Tree	0.695	0.695	0.695	0.695	0.695	69.48%
	Random Forest	0.747	0.439	0.732	0.747	0.730	<b>74.72% (Best)</b>
	Naïve Bayes	0.541	0.303	0.701	0.541	0.547	54.08%
	LibSVM	0.692	0.689	0.592	0.692	0.575	69.16%
	SMO	0.740	0.489	0.723	0.740	0.711	74.00%

Table 6.7 shows the evaluation metrics result of 5000 Android Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced ransomware samples. 5000\_Unbalanced\_Android Dataset consists of 3500 ransomware samples and 1500 benign samples (referred to as "Benign"). Whereas 5000\_Balanced\_Android Dataset consists of 2500 ransomware samples and 2500 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 5000 samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.7, for the unbalanced scenario, Random Forest achieved the highest True Positive Ratio among all classifiers which is 0.704. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.706 as well. The False Positive is relatively lowest among all at 0.299 suggesting that it is classifying non-ransomware (Benign) instances with a high level of precision of 0.706.

For the case of the balanced scenario, it is noticeable that the Random Forest maintains its position as the most effective classifier, reaching the accuracy of 74.72%. It achieved the highest True Positive Ratio among all classifiers which is 0.747 which indicated its effectiveness to correctly identified ransomware sample. Despite the False Positive being the second lowest among all at 0.439, it has the highest Precision at 0.732 which contributes to its being the highest accuracy of 74.72% among all models.



*Figure 6.13 Graph Accuracy for 5000 Android Samples (50:50)*

Figure 6.13 shows the Graph Accuracy for 5000 Android Samples for both cases. The tabular result has been transformed into a graph which is important to provide a clear visual representation of the model performance between the unbalanced and balanced scenarios. The graph illustrates a noticeable increase in accuracy across four models, specifically Decision Tree J48, Random Forest, LibSVM and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario.

To be specific, Naïve Bayes performance shows inconsistency as it decreases from 60.20% down to 54.08%. On the other hand, an increase in accuracy can be seen for Decision Tree J48 from 58.36% up to 69.48%, Random Forest from 70.36% up to 74.72%, significant increase for LibSVM from 50.56% up to 69.16% and SMO from 67.44% up to 74.00%.

As conclusion machine learning algorithms perform better across multiple metrics in the balanced case, where the class distribution is equal for four classification models specifically Decision Tree J48, Random Forest, LibSVM and Support Vector Machine with Polynomial Kernel (SMO) for 5000 Android Samples.

### 6.3.3 Evaluation Metric Result of 10 000 Android Samples for Unbalanced and Balanced Dataset Ransomware Detection

Table 6.8 Evaluation Metrics Result of 10 000 Android Samples (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure	Overall Accuracy
<b>10000_Unbalanced_ Android</b>	Decision Tree	0.660	0.336	0.711	0.660	0.640	66.00%
	Random Forest	0.726	0.273	0.729	0.726	0.726	72.62%
	Naïve Bayes	0.637	0.358	0.704	0.637	0.606	63.68%
	LibSVM	0.510	0.483	0.537	0.510	0.420	51.02%
	SMO	0.690	0.306	0.730	0.690	0.677	69.00%
<b>10000_Balanced_ Android</b>	Decision Tree	0.698	0.698	0.698	0.698	0.698	69.80%
	Random Forest	0.762	0.398	0.751	0.762	0.751	76.24%
	Naïve Bayes	0.571	0.294	0.713	0.571	0.583	57.14%
	LibSVM	0.697	0.688	0.623	0.697	0.583	69.66%
	SMO	0.765	0.469	0.759	0.765	0.736	<b>76.52% (Best)</b>

Table 6.8 shows the evaluation metrics result of 10 000 Android Samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced ransomware samples. 10 000\_Unbalanced\_Android Dataset consists of 7000 ransomware samples and 3000 benign samples (referred to as "Benign"). Whereas 10 000\_Balanced\_Android Dataset consists of 5000 ransomware samples and 5000 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 10 000 samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.8, for the unbalanced scenario Random Forest still maintains its position as the most effective classifier, with the highest True Positive Ratio among all classifiers which is 0.726. The result indicated its effectiveness to correctly identified ransomware samples. The False Positive is relatively lowest among all at 0.273 suggesting that it is classifying non-ransomware (Benign) instances with a high level of precision of 0.706.

For the case of the balanced scenario, it is noticeable that the Support Vector Machine with Polynomial Kernel (SMO) remains its position as the most effective classifier, reaching the greatest accuracy of 76.52%. It achieved the highest True Positive Ratio among all classifiers which is 0.765 which indicated its effectiveness to correctly identified ransomware sample. It has the highest Precision, Recall and F-measure which contributes to its being the highest accuracy at 76.52%

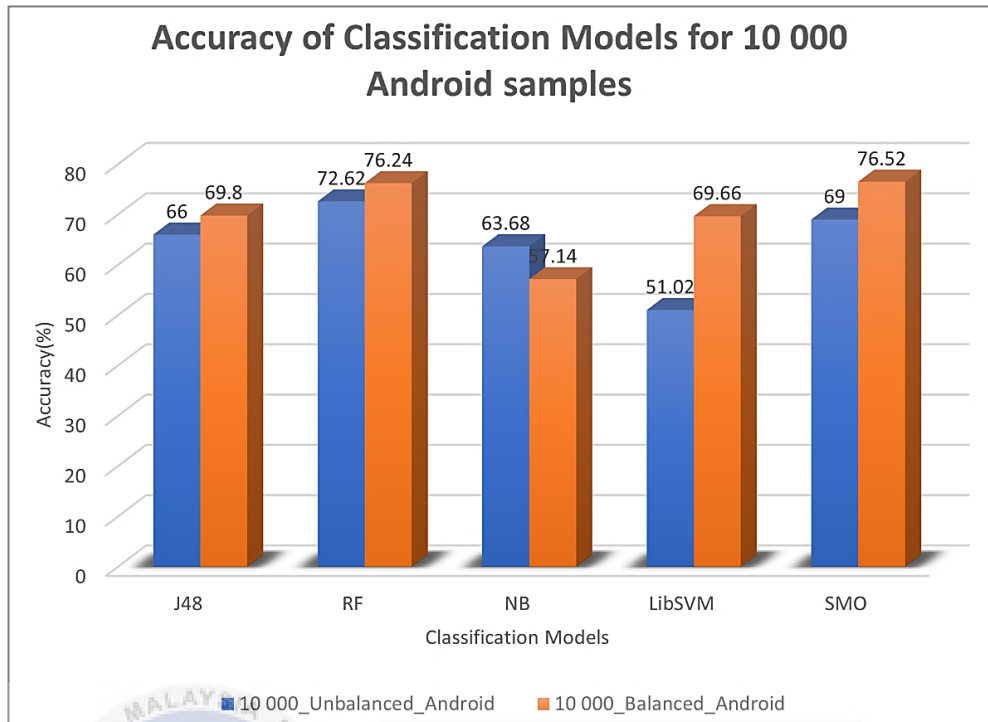


Figure 6.14 Graph Accuracy for 10 000 Android Samples (50:50)

Figure 6.14 shows the Graph Accuracy for 10 000 Android Samples for both cases. The tabular result has been transformed into a graph which is important to provide a clear visual representation of the model performance between the unbalanced and balanced scenarios. The graph illustrates a noticeable increase in accuracy across four models similar with previous test 5000 samples, specifically Decision Tree J48, Random Forest, LibSVM and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario.

To be specific, Naïve Bayes performance shows inconsistency again as it decreases from 63.68% down to 57.14%. On the other hand, an increase in accuracy can be seen for Decision Tree J48 from 66.00% up to 69.8%, Random Forest from 72.62% up to 76.24%, significant increase for LibSVM from 51.02% up to 69.66% and SMO from 69.00% up to 76.52%.

As conclusion machine learning algorithms perform better across multiple metrics in the balanced case, where the class distribution is equal for four classification models specifically Decision Tree J48, Random Forest, LibSVM and Support Vector Machine with Polynomial Kernel (SMO) for 5000 Android Samples.

### 6.3.4 Accuracy of Classification Model Across Different Sample Sizes

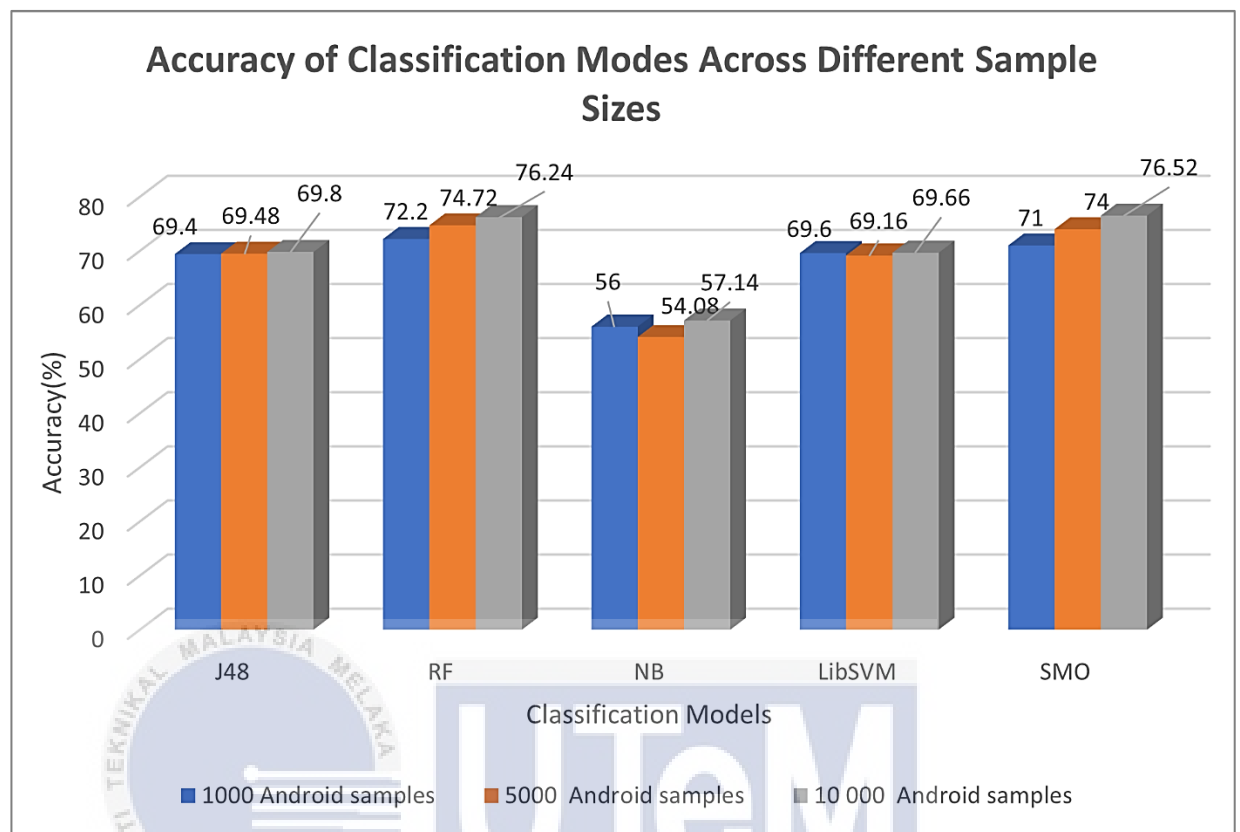


Figure 6.15 Graph Accuracy for Classification Model Across Different Sample Sizes of Android Ransomware (50:50)

Figure 6.15 shows the Graph Accuracy for Classification Model Across Different Sample Sizes of Android Ransomware. The results allow us to visualize the behavior of classification models for the sample of 1000, 5000 and 10 000 by doing the sub-sampling method to prevent heap size issues in WEKA. Based on the results, as the samples increases from 1000 to 5000 and then to 10 000, four out of five model experience increase in accuracy. It includes Decision Tree J48, Random Forest, LibSVM and Support Vector Machine with Polynomial Kernel (SMO). On the other hand, the performance of Naïve Bayes shows inconsistencies as it decrease slightly before increasing up to 57.14%. It can be highlighted that, Support Vector Machine with Polynomial Kernel (SMO) performance is the best out of all starting at 71.00% and increasing to 74.00% and peaked at 76.52% as the sample size increases.

### 6.3.5 Accuracy of Classification Model for Different Ratio

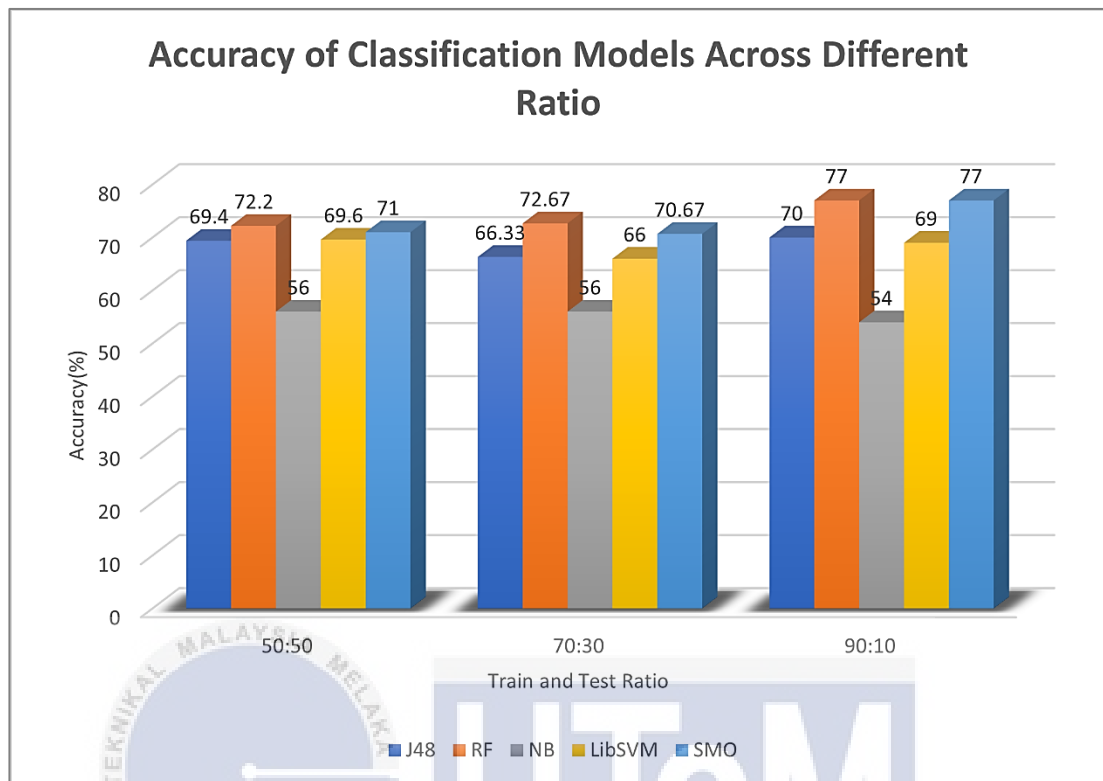
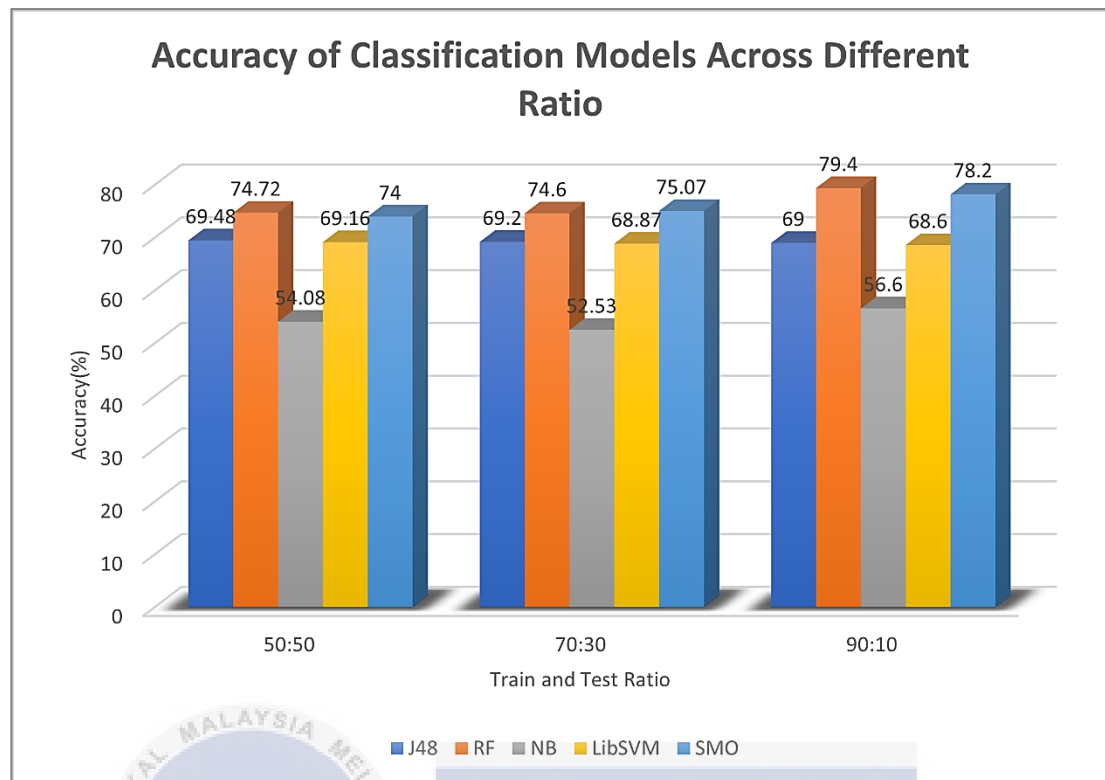


Figure 6.16 Graph Accuracy for 1000 Android Samples across Different Ratio

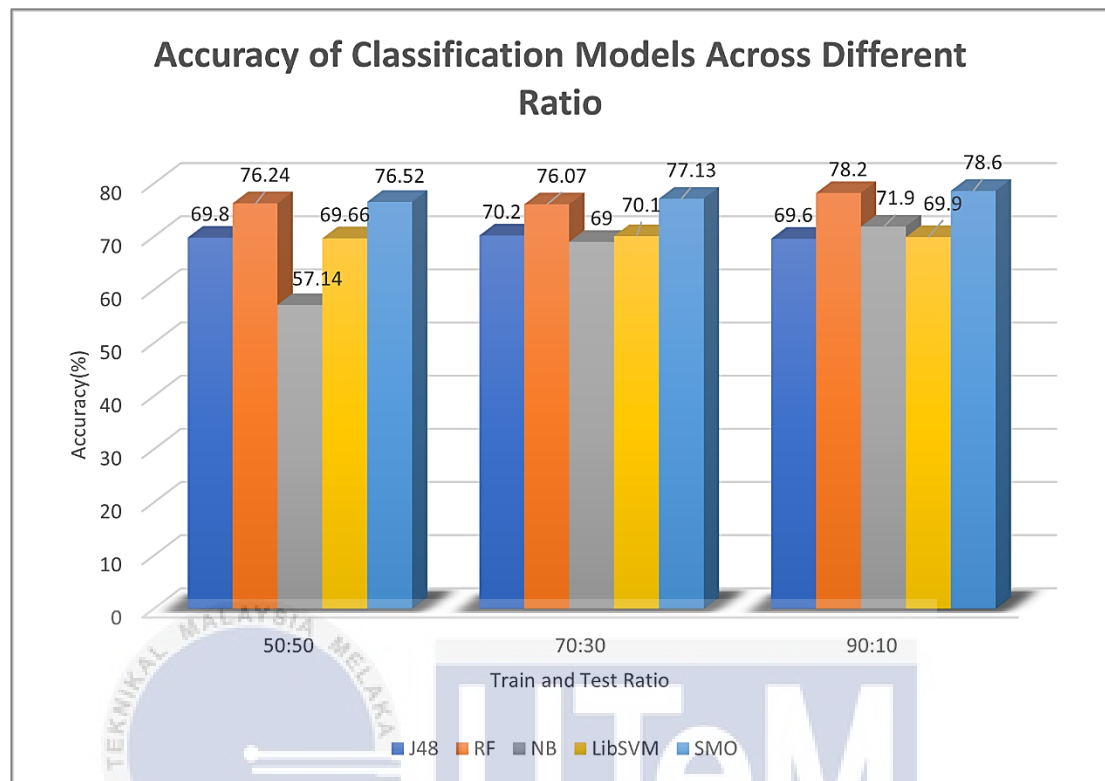
Figure 6.16 shows the Graph Accuracy for 1000 Android samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, Random Forest shows the best performance as it consistently displays an incremental rise in accuracy. To be specific Random Forest shifted from 72.2% up to 72.67% and peaked at 77%. Whereas Decision Tree J48, LibSVM and SMO experienced a slight decrease when transitioning from 50:50 to 70:30 ratio before increasing for 90:10 ratio. On the other hand, Naïve Bayes displays sensitivity to class distributions with limited improvements as the performance is all below 60% of accuracy for all ratio 50:50, 70:30 and 90:30.





*Figure 6.17 Graph Accuracy for 5000 Android Samples across Different Ratio*

Figure 6.17 shows the Graph Accuracy for 5000 Android samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, Support Vector Machine with Polynomial Kernel (SMO), shows the best performance as it consistently displays an incremental rise in accuracy. To be specific SMO shifted from 74.00% up to 75.07% and peaked at 78.20%. Whereas Decision Tree J48, Random Forest experienced a slight decrease 0.28% and 0.12% when transitioning from 50:50 to 70:30 ratio before accuracy increases for 90:10 ratio. It can be highlighted that LibSVM displays constant decrease from 69.16% down to 68.87% and 68.60%. On the other hand, Naïve Bayes still displays sensitivity to class distributions with limited improvements as the performance is all below 60% of accuracy for all ratio 50:50, 70:30 and 90:30.



*Figure 6.18 Graph Accuracy for 10 000 Android Samples across Different Ratio*

Figure 6.18 shows the Graph Accuracy for 10 000 Android samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, Support Vector Machine with Polynomial Kernel (SMO), shows the best performance as it consistently displays an incremental rise in accuracy. To be specific SMO shifted from 76.52% up to 77.13% and peaked at 78.60%. Whereas Random Forest experienced a slight decrease by 0.17% when transitioning from 50:50 to 70:30 ratio before accuracy increases for 90:10 ratio. It can be highlighted that Naïve Bayes finally improves its accuracy compared to previous 1000 and 5000 samples as it displays constant increase from 57.14% up to 69.00% and 69.9%. While it's true that both Decision Tree J48 and LibSVM experience increase in accuracy when transitioning from 50:50 to 70:30 ratio. However, decrease in accuracy can be seen when moving to the 90:10 ratio, which Decision Tree J48 and LibSVM scored down to 69.60% and 69.90% respectively.

### 6.3.6 Comparison between WEKA and Orange

Table 6.9 Summary of results Dataset II for TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.663	0.663	0.663	0.663	0.663	66.33%
	Random Forest	0.727	0.509	0.747	0.727	0.675	<b>72.67% (Best)</b>
	Naïve Bayes	0.560	0.350	0.658	0.560	0.567	56.00%
	LibSVM	0.660	0.646	0.592	0.660	0.550	66.00%
	SMO	0.707	0.549	0.721	0.707	0.642	70.67%
<b>5000</b>	Decision Tree	0.692	0.692	0.692	0.692	0.692	69.20%
	Random Forest	0.746	0.409	0.733	0.746	0.735	74.60%
	Naïve Bayes	0.525	0.301	0.699	0.525	0.527	52.53%
	LibSVM	0.689	0.680	0.606	0.689	0.577	68.87%
	SMO	0.751	0.474	0.740	0.751	0.722	<b>75.07% (Best)</b>
<b>10 000</b>	Decision Tree	0.702	0.702	0.702	0.702	0.702	70.20%
	Random Forest	0.761	0.386	0.750	0.761	0.753	76.07%
	Naïve Bayes	0.690	0.369	0.711	0.690	0.698	69.00%
	LibSVM	0.701	0.683	0.637	0.701	0.596	70.10%
	SMO	0.771	0.462	0.765	0.771	0.744	<b>77.13% (Best)</b>

Table 6.9 shows an in-depth overview of the evaluation's results in WEKA obtained by using balanced Android datasets with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, Random Forest obtains the highest True Positive Rate (TPR) of 0.727 and which contributes to the highest precision of 0.747 with 1000 samples. These numbers represent the algorithm's ability to detect positive cases correctly while minimizing false positives. As the dataset size increases to 5000 samples, it can be seen Support Vector Machine with Polynomial Kernel (SMO) emerge as the best performance from 70.67% to 75.07% and peaked at 77.13% for 10000 Android datasets samples as the best classification algorithms.

It can be concluded that, for WEKA Support Vector Machine with Polynomial Kernel (SMO) approach consistently outperforms other classification algorithms on balanced Android datasets with variable sample sizes and a 70:30 ratio. With accuracy scores of 70.67% to 75.07% up to 77.13% for 1000, 5000, and 10,000 samples, respectively remains the best classification algorithms. Following closely behind, Random Forest performs successfully, with the highest accuracy of 72.67% for 1000 samples. These findings highlight the reliability of Support Vector Machine with Polynomial Kernel (SMO) and Random Forest for effective ransomware detection over an extensive selection of dataset sizes.

Table 6.10 Summary of results for Dataset II for TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.615	0.462	0.574	0.573	0.573	57.30%
	Random Forest	0.630	0.426	0.615	0.615	0.615	<b>61.50% (Best)</b>
	Naïve Bayes	0.513	0.486	0.584	0.584	0.584	58.40%
	SVM	0.375	0.346	0.514	0.511	0.450	51.10%
<b>5000</b>	Decision Tree	0.606	0.457	0.575	0.574	0.574	57.40%
	Random Forest	0.611	0.416	0.598	0.598	0.598	<b>59.80% (Best)</b>
	Naïve Bayes	0.522	0.518	0.502	0.502	0.502	50.20%
	SVM	0.267	0.259	0.505	0.504	0.475	50.40%
<b>10 000</b>	Decision Tree	0.618	0.452	0.583	0.583	0.582	58.30%
	Random Forest	0.627	0.400	0.614	0.614	0.614	<b>61.40% (Best)</b>
	Naïve Bayes	0.555	0.537	0.509	0.509	0.508	50.90%
	SVM	0.368	0.365	0.502	0.502	0.492	50.20%

Table 6.10 shows an in-depth overview of the evaluation's results in ORANGE obtained by using balanced Android datasets with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, the ORANGE examination of balanced Android datasets with a 70:30 ratio across three sample sizes demonstrates slight increase of the Random Forest classifier performance. Its accuracy gradually rises around the beginning at 61.50% for 1000 samples, slightly decrease to 59.80% for 5000 samples, and finally peaked at 61.40% accuracy for 10,000 samples.

The Decision Tree classifier follows in second best performance in ORANGE closely followed the Random Forest, and its accuracy shows a consistent rising trend across various sample sizes. It begins with an accuracy of 57.30% for 1000 samples, slightly increases to 57.30% for 5000 samples, and then peaked at accuracy of 58.30% for 10,000 samples. This pattern shows a continuous 3.8% increase from the smallest to the largest sample. This development emphasizes the Decision Tree's ability to make accurate predictions. In conclusion, Random Forest performance outperform all others in ORANGE and followed by the Decision Tree.

## 6.4 Result and Analysis Dataset III

In this section, the results from the project implementation phase are presented. All the results for Dataset III focuses on File System Behavior Ransomware dataset and discussion are shown below:

### 6.4.1 Evaluation Metrics Result of 1000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection

Table 6.11 Evaluation Metrics Result of 1000 File System Behavior Ransomware dataset (50:50)

Dataset	Algorithms	True Positive	False Positive	Precision	Recall	F-Measure	Overall Accuracy
		Rate	Rate				
<b>1000_Unbalanced_FileSystem</b>	Decision Tree	0.956	0.034	0.959	0.956	0.957	95.60%
	Random Forest	0.904	0.218	0.916	0.904	0.898	90.40%
	Naïve Bayes	0.812	0.353	0.810	0.812	0.799	81.20%
	LibSVM	0.738	0.594	0.810	0.738	0.661	73.80%
	SMO	0.894	0.215	0.899	0.894	0.889	89.40%
<b>1000_Balanced_FileSystem</b>	Decision Tree	0.954	0.045	0.955	0.954	0.954	95.40%
	Random Forest	0.960	0.040	0.960	0.960	0.960	<b>96.00% (Best)</b>
	Naïve Bayes	0.736	0.253	0.800	0.736	0.723	73.60%
	LibSVM	0.680	0.336	0.803	0.680	0.640	68.00%
	SMO	0.828	0.166	0.853	0.828	0.826	82.80%

Table 6.11 shows the evaluation metrics result of 1000 File System Behavior Ransomware dataset samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced File System Behavior Ransomware samples. 1000\_Unbalanced\_FileSystem Dataset consists of 700 ransomware samples and 300 benign samples (referred to as "Benign"). Whereas 1000\_Balanced\_FileSystem Dataset consists of 500 ransomware samples and 500 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 1000 File System Behavior Ransomware samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.11, for the unbalanced scenario, the Decision Tree achieved the highest True Positive Ratio among all classifiers which is 0.956. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.959. The False Positive is relatively lowest among all at 0.034 suggesting that it is classifying non-ransomware (Benign) instances with a high level of accuracy which is 95.60%.

For the case of balanced scenario, Random Forest maintains its good performance as the best classifier with highest accuracy 96.00%. It achieved the highest True Positive Ratio among all classifiers which is 0.960 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.040 suggesting that it is classifying non-ransomware (Benign) instances with a high level of accuracy which is 96.00 %.





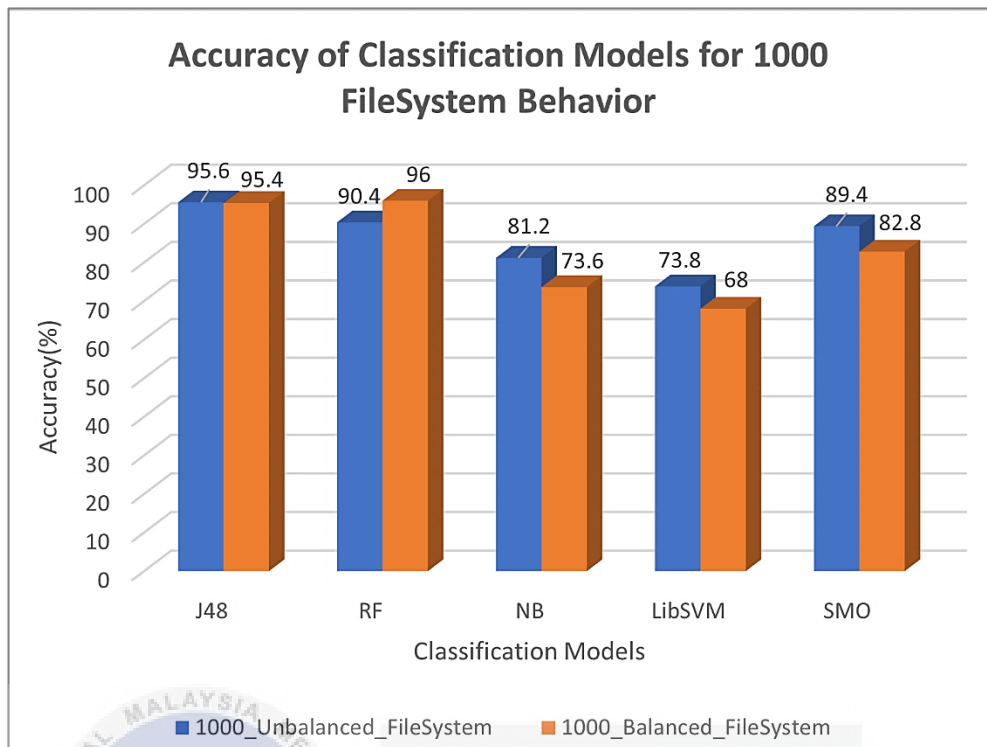


Figure 6.19 Graph Accuracy for 1000 File System Behavior Ransomware dataset (50:50)

Figure 6.19 shows the Graph Accuracy for 1000 File System Behavior Ransomware dataset samples for both cases. The tabular result has been transformed into a graph which is important to provide a clear visual representation of the model performance between the unbalanced and balanced scenarios. The graph illustrates a noticeable increase in accuracy across all five models, specifically Decision Tree J48, Random Forest, Naïve Bayes, LibSVM and Support Vector Machine with Polynomial Kernel (SMO), for transition from the unbalanced scenario to the balanced scenario.

To be specific, for the unbalanced dataset, Decision Tree J48 demonstrated its good performance by achieving highest accuracies of 95.6% but slightly decreased to 95.4% when transitioning to balanced dataset. Naïve Bayes, LibSVM and Support Vector Machine with Polynomial Kernel (SMO) also shows inconsistencies as the dataset transition to balanced state.

On the other hand, a significant improvement in accuracy can be seen for Random Forest which from 90.40% and peaked at 96.00% as the highest accuracy of for 1000 File System Behavior Ransomware dataset samples.

### 6.4.2 Evaluation Metrics Result of 5000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection

Table 6.12 Evaluation Metrics Result of 5000 File System Behavior Ransomware dataset (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure	Overall Accuracy
<b>5000_Unbalanced_FileSystem</b>	Decision Tree	0.991	0.017	0.991	0.991	0.991	99.08%
	Random Forest	0.855	0.331	0.880	0.855	0.839	85.48%
	Naïve Bayes	0.737	0.226	0.786	0.737	0.747	73.68%
	LibSVM	0.763	0.539	0.823	0.763	0.705	76.32%
	SMO	0.880	0.241	0.885	0.880	0.874	88.04%
<b>5000_Balanced_FileSystem</b>	Decision Tree	0.989	0.008	0.991	0.991	0.991	<b>99.10% (Best)</b>
	Random Forest	0.952	0.049	0.952	0.952	0.952	95.16%
	Naïve Bayes	0.845	0.152	0.854	0.845	0.845	84.52%
	LibSVM	0.771	0.237	0.841	0.771	0.758	77.12%
	SMO	0.834	0.162	0.853	0.834	0.832	83.36%

Table 6.12 shows the evaluation metrics result of 5000 File System Behavior Ransomware dataset samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced File System Behavior Ransomware samples. 5000\_Unbalanced\_FileSystem Dataset consists of 3500 ransomware samples and 1500 benign samples (referred to as "Benign"). Whereas 5000\_Balanced\_FileSystem Dataset consists of 2500 ransomware samples and 2500 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 1000 File System Behavior Ransomware samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.12, for the unbalanced scenario, Decision Tree J48 achieved the highest True Positive Ratio among all classifiers which is 0.991. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.991 as well. The False Positive is relatively lowest among all at 0.017 suggesting that it is classifying non-ransomware (Benign) instances with a high level of accuracy at 99.08%.

For the case of the balanced scenario, it is noticeable that the Decision Tree J48 maintains its position as the most effective classifier, reaching the accuracy of 99.10%. It achieved the highest True Positive Ratio among all classifiers which is 0.981 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.019 suggesting that it is classifying non-ransomware (Benign) instances with a high level of precision of 0.981.

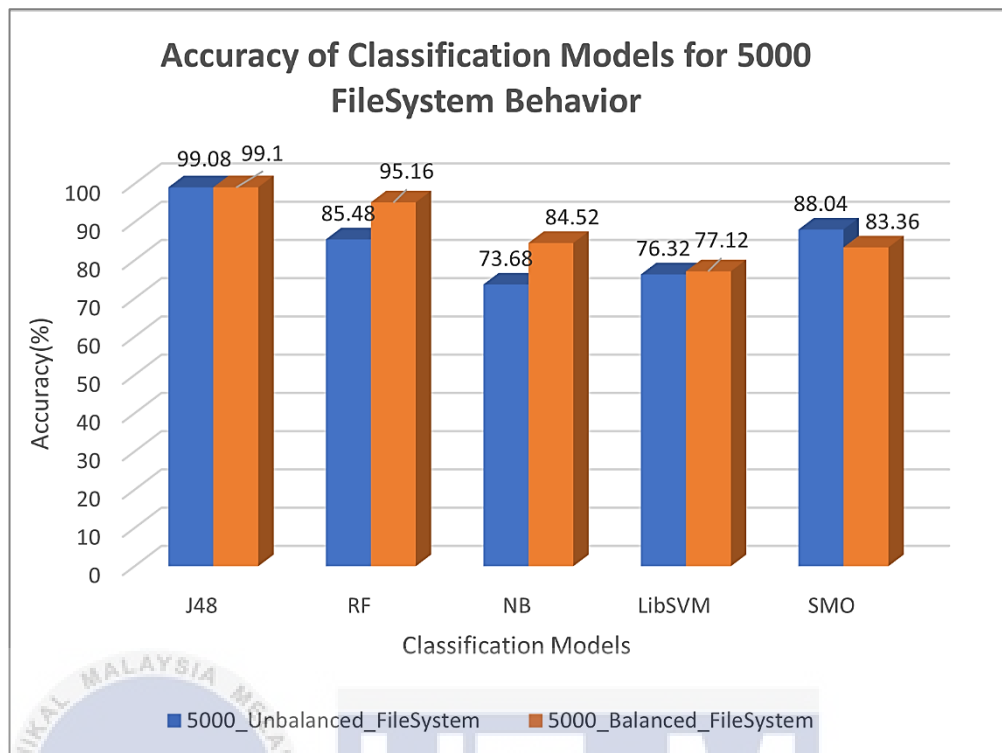


Figure 6.20 Graph Accuracy for 5000 File System Behavior Ransomware dataset (50:50)

Figure 6.20 shows the Graph Accuracy for 5000 File System Behavior Ransomware Dataset Samples for both cases. The graph illustrates a noticeable increase in accuracy across three models, specifically Random Forest, Naïve Bayes, and LibSVM for transition from the unbalanced scenario to the balanced scenario. It can be highlight that the performance of Naïve Bayes has significantly increase for 5000 samples (73.68% and 84.52%) compared to the previous 1000 samples (81.2% and 73.6%).

This shows that the Naïve Bayes model performance increases as the sample size increase to 5000 samples. Whereas Decision Tree J48 experience a slight increment in accuracy precisely 99.08% for unbalanced. For the case of balanced scenario, Decision Tree J48 still maintains its good performance increase up to accuracy of 99.10%.

### 6.4.3 Evaluation Metrics Result of 10 000 File System Behavior Ransomware dataset for Unbalanced and Balanced Ransomware Detection

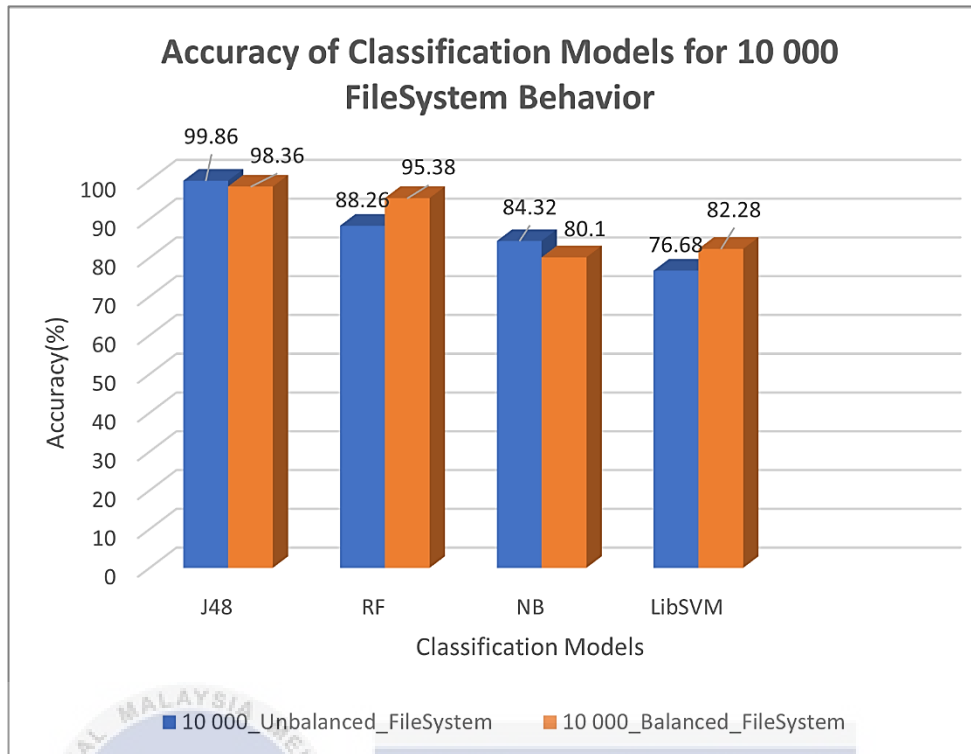
Table 6.13 Evaluation Metrics Result of 10 000 File System Behavior Ransomware dataset (50:50)

Dataset	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure	Overall Accuracy
<b>10 000_ Unbalanced_ FileSystem</b>	Decision Tree	0.989	0.018	0.989	0.989	0.989	<b>98.86%(Best)</b>
	Random Forest	0.883	0.271	0.899	0.883	0.873	88.26%
	Naïve Bayes	0.843	0.205	0.845	0.843	0.844	84.32%
	LibSVM	0.767	0.538	0.823	0.767	0.711	76.68%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A
<b>10 000_ Balanced_ FileSystem</b>	Decision Tree	0.984	0.016	0.984	0.984	0.984	98.36%
	Random Forest	0.954	0.046	0.954	0.954	0.954	95.38%
	Naïve Bayes	0.801	0.202	0.831	0.801	0.796	80.10%
	LibSVM	0.823	0.174	0.869	0.823	0.817	82.28%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.13 shows the evaluation metrics result of 10 000 File System Behavior Ransomware dataset samples. In this experiment the machine learning classifiers are tested for two scenarios: unbalanced and balanced File System Behavior Ransomware samples. 10000\_Unbalanced\_FileSystem Dataset consists of 7000 ransomware samples and 3000 benign samples (referred to as "Benign"). Whereas 10000\_Balanced\_FileSystem Dataset consists of 5000 ransomware samples and 5000 benign samples (referred to as "Benign"). The purpose of these two datasets is to identify the performance between balanced and unbalanced ransomware datasets of 10000 File System Behavior Ransomware samples. For each case, the following classification algorithms were applied, and their related evaluation metrics were calculated.

Based on the result as shown in Table 6.13, for the unbalanced scenario, Decision Tree J48 achieved the highest True Positive Ratio among all classifiers which is 0.989. The result indicated its effectiveness to correctly identified ransomware samples thus giving the highest precision of 0.989 as well. The False Positive is relatively lowest among all at 0.018 suggesting that it is classifying non-ransomware (Benign) instances with a high level of accuracy at 98.86%

For the case of the balanced scenario, it is noticeable that the Decision Tree J48 maintains its position as the most effective classifier, reaching the accuracy of 98.36%. It achieved the highest True Positive Ratio among all classifiers which is 0.984 which indicated its effectiveness to correctly identified ransomware sample. The False Positive is relatively lowest among all at 0.016 suggesting that it is classifying non-ransomware (Benign)) instances with a high level of precision of 0.984.



*Figure 6.21 Graph Accuracy for 10 000 File System Behavior Ransomware dataset (50:50)*

Figure 6.2 shows the Graph Accuracy for 10 000 File System Behavior Ransomware Dataset Samples for both cases. The graph illustrates a noticeable increase in accuracy across two models, specifically Random Forest and LibSVM for transition from the unbalanced scenario to the balanced scenario. In this case Support Vector Machine with Polynomial Kernel (SMO) labelled as N/A since it didn't show result even after running more than 3 hours in WEKA. In addition to that. Random Forest experiences a slight increment in accuracy as it increases from 88.26% for Unbalanced sampled and up to 95.38% when transitioning to balanced sampled. Based on the result above, it can be highlighted that Decision Tree J48 maintains its position as the most effective classifier for both unbalanced and balanced samples.



#### 6.4.4 Accuracy of Classification Model Across Different Sample Sizes

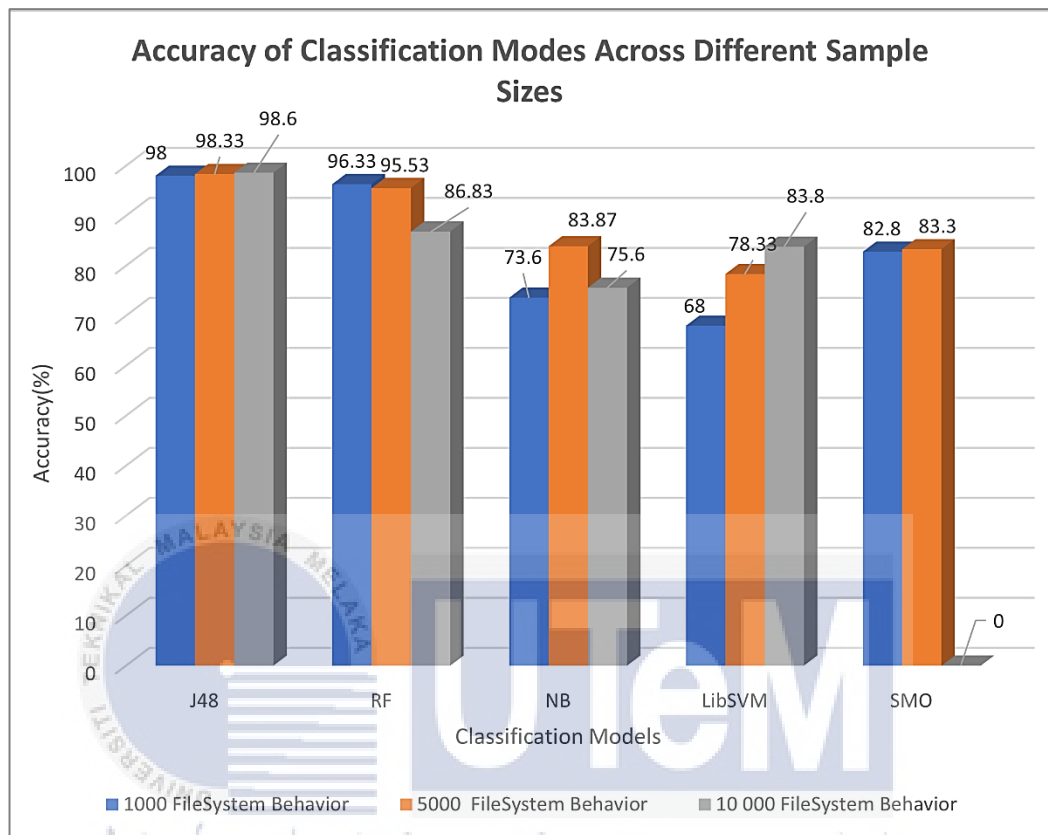


Figure 6.22 Graph Accuracy for Classification Model Across Different Sample Sizes of FileSystem Behavior Ransomware (50:50)

Figure 6.22 shows the Graph Accuracy for Classification Model Across Different Sample Sizes of FileSystem Behavior Ransomware. The results allow us to visualize the behavior of classification models for the sample of 1000, 5000 and 10 000 by doing the sub-sampling method to prevent heap size issues in WEKA. Based on the results, as the samples increases from 1000 to 5000 and 10 000, four out of five model experience increase in accuracy. It includes Decision Tree J48, Naïve Bayes, LibSVM and Support Vector Machine with Polynomial Kernel (SMO). However, Naïve Bayes shows inconsistencies as it decreases for transition from 5000 to 10 000. In this case it can be highlighted that, Decision Tree J48 shows the best performance out of all starting at 98.00% and increasing to 98.33% and peaked at 98.60% as the sample size increases.

### 6.4.5 Accuracy of Classification Model for Different Ratio

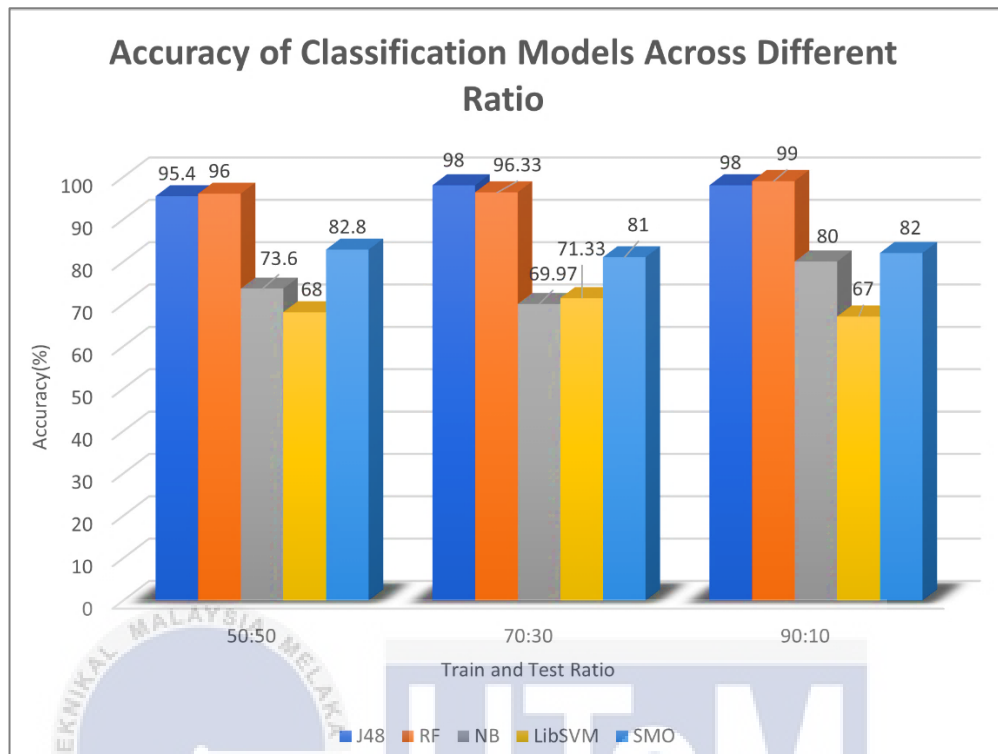


Figure 6.23 Graph Accuracy for 1000 File System Behavior Ransomware Dataset across Different Ratio

Figure 6.23 shows the Graph Accuracy for 1000 File System Behavior Ransomware Dataset samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, Random Forest and Decision Tree J48 shows the best performance as it consistently displays an incremental rise in accuracy. To be specific Random Forest shifted from 96.00% for 50:50, experiences a slight increase to 96.33% for 70:30, and reaches an impressive 99.00% for 90:10. Whereas Decision Tree J48 also shows a consistent increase from 95.4% up to 98% for both 70:30 and 90:10 ratios. On the other hand, Naïve Bayes, LibSVM and SMO displays sensitivity to class distributions with limited improvements in accuracy.

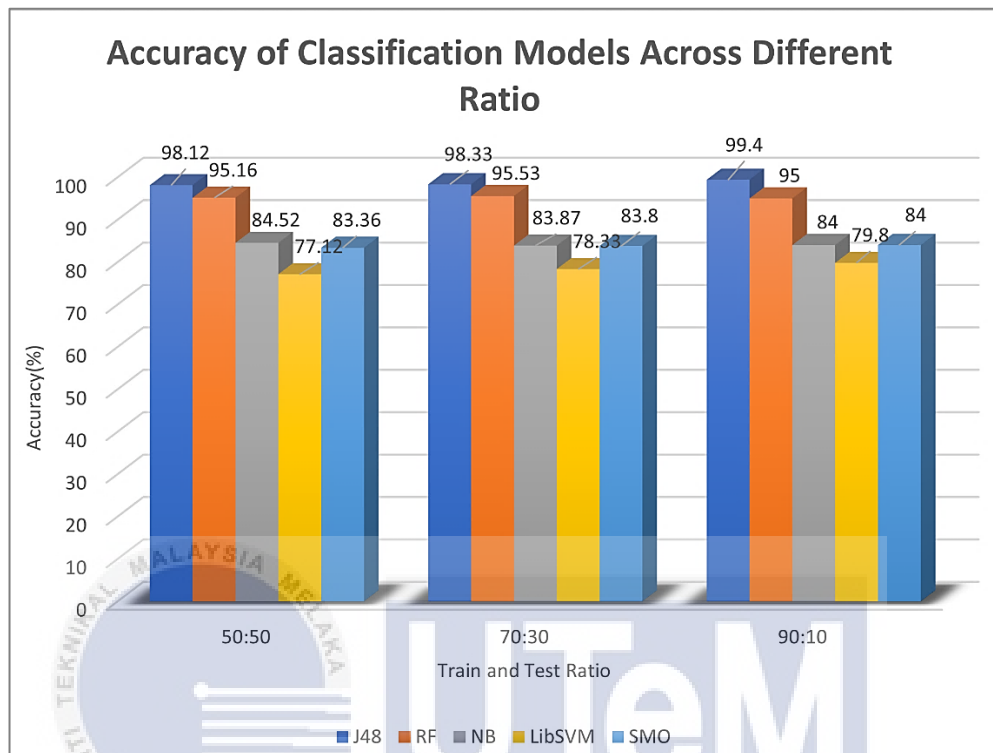


Figure 6.24 Graph Accuracy for 5000 File System Behavior Ransomware Dataset across Different Ratio

Figure 6.24 shows the Graph Accuracy for 5000 File System Behavior Ransomware Dataset samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, It can be highlighted that Decision Tree J48 consistently leads with the highest accuracy levels across all ratios. To be specific, Decision Tree shifted from 98.12% for 50:50, experiences a slight increase to 98.33% for 70:30, and peaked at 99.40% for 90:10. Random Forest followed closely as the second-best classifier with its best performance at 95.53% for 70:30 ratio.

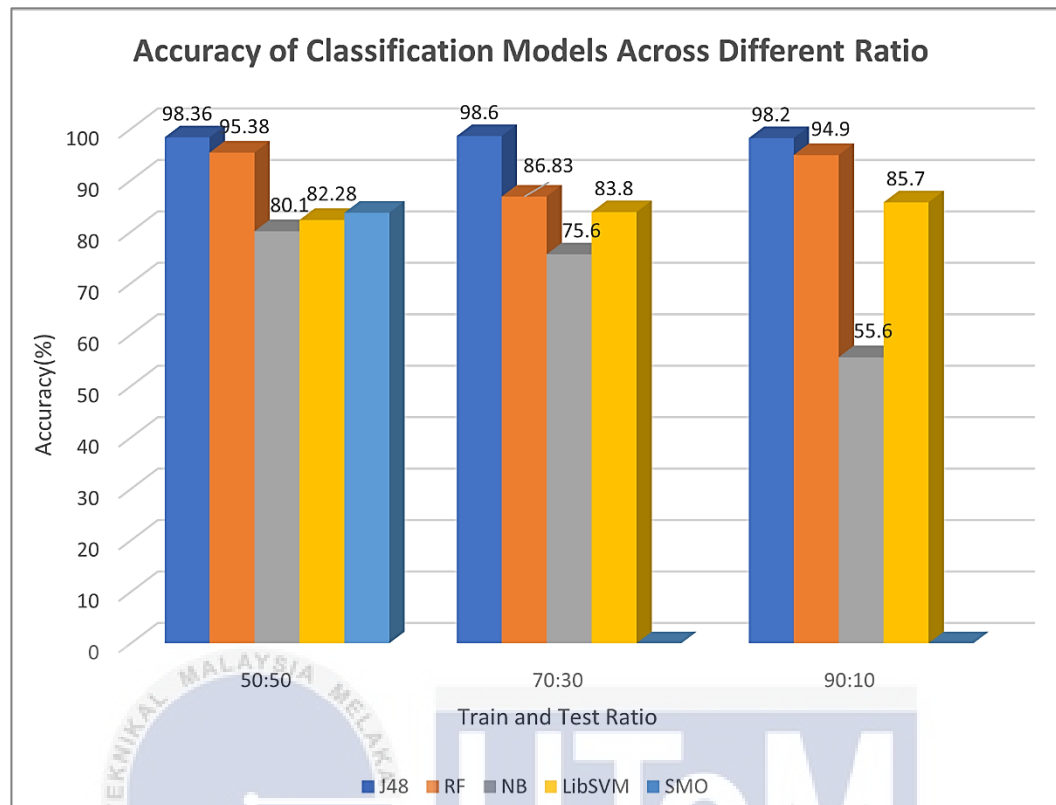


Figure 6.25 Graph Accuracy for 10 000 File System Behavior Ransomware Dataset across Different Ratio

Figure 6.25 shows the Graph Accuracy for 10 000 File System Behavior Ransomware Dataset samples across different ratios. The results allow us to visualize the behavior of classification models for the ratio of 50:50, 70:30 and 90:10 train and test ratio. Based on the results, as the class ratios move from 50:50 to 70:30 and then to 90:10, It can be highlighted that Decision Tree J48 consistently leads with the highest accuracy levels across all ratios. To be specific, Decision Tree shifted from 98.36% for 50:50, experiences a slight increase to 98.60% for 70:30, and slightly decrease to 98.20% for 90:10. Random Forest followed closely as the second-best classifier with its best performance after Decision Tree. Random Forest shifted from 95.38% for 50:50, decrease 86.83% for 70:30 and increase again to 94.90% for 90:10. On the other hand, Support Vector Machine with Polynomial Kernel (SMO) didn't show result even after running more than 3 hours using WEKA. In addition to that, LibSVM shows consistent increase as the ratio transition, which start at 82.28% for 50:50, slightly increase up to 83.80% for 70:30 and peaked at 85.70% for 90:10.

#### 6.4.6 Comparison between WEKA and Orange

Table 6.14 Summary of results Dataset III TPR, FPR, Precision, Recall, F-measure and Accuracy in WEKA (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.980	0.017	0.981	0.980	0.980	<b>98.00% (Best)</b>
	Random Forest	0.963	0.038	0.963	0.963	0.963	96.33%
	Naïve Bayes	0.697	0.264	0.803	0.697	0.677	69.97%
	LibSVM	0.713	0.332	0.813	0.713	0.679	71.33%
	SMO	0.810	0.169	0.847	0.810	0.808	81.00%
<b>5000</b>	Decision Tree	0.983	0.017	0.983	0.983	0.983	<b>98.33% (Best)</b>
	Random Forest	0.955	0.045	0.956	0.955	0.955	95.53%
	Naïve Bayes	0.839	0.161	0.845	0.839	0.838	83.87%
	LibSVM	0.783	0.217	0.848	0.783	0.773	78.33%
	SMO	0.838	0.162	0.856	0.838	0.836	83.80%
<b>10 000</b>	Decision Tree	0.986	0.014	0.986	0.986	0.986	<b>98.60% (Best)</b>
	Random Forest	0.868	0.131	0.875	0.868	0.868	86.83%
	Naïve Bayes	0.756	0.247	0.812	0.756	0.744	75.60%
	LibSVM	0.838	0.165	0.877	0.838	0.833	83.80%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.14 shows an in-depth overview of the evaluation's results in WEKA obtained by using balanced File System Behavior Ransomware Dataset with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, Decision Tree J48 obtains the highest True Positive Rate (TPR) of 0.980 and which contributes to the highest precision of 0.981 with 1000 samples. These numbers represent the algorithm's ability to detect positive cases correctly while minimizing false positives.

As the dataset size increases to 5000 samples, it can be seen Decision Tree J48 for WEKA still maintains as the best performance from 98.00% to 98.33% and peaked at 98.60% for 10 000 File System Behavior Ransomware Dataset samples as the best classification algorithms. Following closely behind, the Random Forest performs, starting with accuracy of 96.33% for 1000 samples, slightly decreases 95.53% for 5000 samples and accuracy of 86.83% for 10 000 samples. These findings highlight the reliability of Decision Tree J48 and Random Forest for effective ransomware detection over an extensive selection of dataset sizes.

Table 6.15 Summary of results Dataset III for TPR, FPR, Precision, Recall, F-measure and Accuracy in ORANGE (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.961	0.026	0.968	0.968	0.968	96.80%
	Random Forest	0.986	0.009	0.988	0.988	0.988	<b>98.80% (Best)</b>
	Naïve Bayes	0.810	0.067	0.877	0.872	0.871	87.20%
	SVM	0.888	0.040	0.926	0.924	0.924	92.40%
<b>5000</b>	Decision Tree	0.959	0.020	0.969	0.969	0.969	96.90%
	Random Forest	0.989	0.008	0.991	0.991	0.991	<b>99.10% (Best)</b>
	Naïve Bayes	0.683	0.057	0.882	0.876	0.875	87.60%
	SVM	0.810	0.059	0.836	0.813	0.810	81.30%
<b>10 000</b>	Decision Tree	0.964	0.018	0.973	0.973	0.973	97.30%
	Random Forest	0.991	0.005	0.933	0.933	0.933	<b>99.30% (Best)</b>
	Naïve Bayes	0.826	0.044	0.898	0.891	0.891	89.10%
	SVM	0.764	0.115	0.829	0.824	0.824	82.40%

Table 6.15 shows an in-depth overview of the evaluation's results in ORANGE obtained by using balanced File System Behavior Ransomware Dataset with sample sizes of 1000, 5000, and 10,000, all with a 70:30 ratio. The aim of this table is to present a comprehensive summary of several classification algorithms' performance measures, with a focus on True Positive Rate (TPR), False Positive Rate (FPR), precision, recall, F-measure, and overall accuracy. It can be highlighted that, the ORANGE examination of balanced File System Behavior Ransomware Dataset with a 70:30 ratio across three sample sizes demonstrates a good increase of the Random Forest as the best classifier performance. Its accuracy gradually rises around the beginning at 98.80% for 1000 samples, significantly increase to 99.10% for 5000 samples, and finally peaked at 99.30% accuracy for 10,000 samples.

The Decision Tree classifier follows in second best performance in ORANGE closely followed the Random Forest, and its accuracy shows a consistent rising trend across various sample sizes. It begins with an accuracy of 96.80% for 1000 samples, slightly increases to 96.90% for 5000 samples, and then peaked at accuracy of 97.30% for 10,000 samples. This pattern shows a continuous increase from the smallest to the largest sample. This development emphasizes the Decision Tree's ability to make accurate predictions. In conclusion, Random Forest performance outperform all others as the sample size increases in ORANGE and followed by the Decision Tree.



## 6.5 Tools Comparison

Table 6.16 Classifier performance comparison between WEKA and Orange  
(10 000 samples ,10:90)

Dataset	Tools	Decision Tree	Random Forest	Naïve Bayes	Support Vector Machines
<b>I</b>	Weka	<b>95.04%</b>	94.06%	66.79%	63.34%
	Orange	93.80%	<b>94.10%</b>	86.70%	90.40%
<b>II</b>	Weka	<b>69.86%</b>	68.27%	58.12%	69.51%
	Orange	54.20%	<b>55.40%</b>	50.60%	50.60%
<b>III</b>	Weka	<b>96.67%</b>	95.08%	82.49%	73.65%
	Orange	95.60%	<b>98.60%</b>	88.30%	95.40%
<b>Occurrences</b>		<b>3</b>	<b>3</b>	<b>0</b>	<b>0</b>
<b>Average Accuracy</b>	Weka	<b>87.19%(Best)</b>	85.80%	69.13%	68.83%
	Orange	81.20%	<b>82.70%(Best)</b>	75.20%	78.80

Table 6.16 shows comparison of classification techniques performance between WEKA and Orange for 10 000 samples using 10:90 train and test ratio. The classification techniques were applied to the Dataset I (BitcoinHeist Ransomware), Dataset II (Android Ransomware) and Dataset III (File System Behavior Ransomware). Based on the result, it can be highlighted, Decision Tree performs the best among all classifiers in WEKA, achieving an average accuracy of 87.19%. On the other hand, when using Orange, Random Forest with an average accuracy of 82.70% outperforms other classifiers. These findings highlight the effectiveness of Decision Tree and Random Forest with same number of occurrences in ransomware detection when using a 10:90 training and test ration for 10,000 samples.

Table 6.17 Classifier performance comparison between WEKA and Orange  
(10 000 samples ,30:70)

Dataset	Tools	Decision Tree	Random Forest	Naïve Bayes	Support Vector Machines
<b>I</b>	Weka	94.96%	<b>96.46%</b>	53.54%	64.40%
	Orange	94.40%	<b>95.10%</b>	86.70%	80.40%
<b>II</b>	Weka	69.94%	<b>74.61%</b>	51.99%	69.50%
	Orange	57.60%	<b>60.10%</b>	50.60%	50.10%
<b>III</b>	Weka	<b>98.53%</b>	95.40%	79.01%	80.01%
	Orange	97.30%	<b>98.90%</b>	88.00%	83.80%
<b>Occurrences</b>		<b>1</b>	<b>5</b>	<b>0</b>	<b>0</b>
<b>Average Accuracy</b>	Weka	87.81%	<b>88.82% (Best)</b>	61.51%	71.30%
	Orange	83.10%	<b>84.70% (Best)</b>	75.10%	71.43%

Table 6.17 shows comparison of classification techniques performance between WEKA and Orange for 10 000 samples using 30:70 train and test ratio. The classification techniques were applied to the Dataset I (BitcoinHeist Ransomware), Dataset II (Android Ransomware) and Dataset III (File System Behavior Ransomware). Based on the result, it can be concluded that Random Forest is the best classification technique, in both WEKA and Orange with most occurrences as the average best classifier for this case. Specifically, Random Forest achieved an average accuracy of 88.82% in WEKA and 84.70% in Orange. These findings highlight the effectiveness of Random Forest with better accuracy than previous 10:90 ratio for ransomware detection when using a 30:70 dataset split with 10,000 samples.

Table 6.18 Classifier performance comparison between WEKA and Orange  
(10 000 samples ,50:50)

Dataset	Tools	Decision Tree	Random Forest	Naïve Bayes	Support Vector Machines
<b>I</b>	Weka	95.2%	<b>95.46%</b>	54.84%	64.98%
	Orange	94.30%	<b>95.30%</b>	86.70%	76.80%
<b>II</b>	Weka	69.80%	<b>76.24%</b>	57.14%	69.66%
	Orange	57.60%	<b>60.10%</b>	50.60%	50.10%
<b>III</b>	Weka	<b>98.36%</b>	95.38%	80.10%	82.28%
	Orange	97.20%	<b>99.20%</b>	89.10%	82.00%
<b>Occurrences</b>		<b>1</b>	<b>5</b>	<b>0</b>	<b>0</b>
<b>Average Accuracy</b>	Weka	87.78%	<b>89.03%</b> (Best)	64.02%	72.31%
	Orange	83.03%	<b>84.86%</b> (Best)	75.46%	69.63%

Table 6.18 shows comparison of classification techniques performance between WEKA and Orange for 10 000 samples using 50:50 train and test ratio. The classification techniques were applied to the Dataset I (BitcoinHeist Ransomware), Dataset II (Android Ransomware) and Dataset III (File System Behavior Ransomware). Based on the result, it can be concluded that Random Forest is the best classification technique, in both WEKA and Orange with most occurrences as the average best classifier. Specifically, Random Forest achieved an average accuracy of 89.03% in WEKA and 84.86% in Orange. These findings highlight the effectiveness of Random Forest in ransomware detection when using a 50:50 dataset split with 10,000 samples.

Table 6.19 Classifier performance comparison between WEKA and Orange  
(10 000 samples ,70:30)

Dataset	Tools	Decision Tree	Random Forest	Naïve Bayes	Support Vector Machines
<b>I</b>	Weka	95.27%	<b>98.67%</b>	55.33%	65.97%
	Orange	94.50%	<b>95.50%</b>	86.40%	77.30%
<b>II</b>	Weka	70.20%	<b>76.07%</b>	69.00%	70.10%
	Orange	58.30	<b>61.40%</b>	50.90%	50.20%
<b>III</b>	Weka	<b>98.60%</b>	86.83%	75.60%	83.80%
	Orange	97.30%	<b>99.30%</b>	89.10%	82.40%
<b>Occurrences</b>		<b>1</b>	<b>5</b>	<b>0</b>	<b>0</b>
<b>Average</b>	Weka	<b>88.02%(Best)</b>	87.19%	66.64%	78.29%
	Orange	83.37%	<b>85.40% (Best)</b>	75.47%	69.97%

Table 6.19 shows comparison of classification techniques performance between WEKA and Orange for 10 000 samples using 70:30 train and test ratio. The classification techniques were applied to the Dataset I (BitcoinHeist Ransomware), Dataset II (Android Ransomware) and Dataset III (File System Behavior Ransomware). In this scenario, when using WEKA, Decision Tree emerges as the best classifier with an average accuracy of 88.02%. Orange on the other hand, has an average accuracy of 85.40% with Random Forest being the highest. It can be highlighted, while Decision Tree emerges as the best-performing classifier for WEKA in this scenario, it's important to note that

Random Forest maintains its status as the overall best classifier due to the highest number of occurrences across different scenarios when using a 50:50 dataset split with 10,000 samples.

Table 6.20 Classifier performance comparison between WEKA and Orange  
(10 000 samples, 90:10)

Dataset	Tools	Decision Tree	Random Forest	Naïve Bayes	Support Vector Machines
I	Weka	95.30%	<b>98.40%</b>	56.40%	66.60 %
	Orange	94.20%	<b>95.40%</b>	86.50%	77.00%
II	Weka	69.60%	<b>78.20%</b>	71.90%	69.90%
	Orange	59.10%	<b>61.30%</b>	50.60%	50.10%
III	Weka	<b>98.20%</b>	94.90%	55.60%	85.70%
	Orange	97.60%	<b>99.30%</b>	89.40%	78.00%
<b>Occurrences</b>		<b>1</b>	<b>5</b>	<b>0</b>	<b>0</b>
<b>Average</b>	Weka	87.70%	<b>90.50%</b> <b>(Best)</b>	61.30%	74.07
	Orange	83.63%	<b>85.33%</b> <b>(Best)</b>	75.5%	68.37

Table 6.20 shows comparison of classification techniques performance between WEKA and Orange for 10 000 samples using 90:10 train and test ratio. Based on the result, it can be concluded that Random Forest is the best classification technique, in both WEKA and Orange with most occurrences as the average best classifier. Specifically, Random Forest achieved an average

accuracy of 90.50% in WEKA and 85.33% in Orange. These findings highlight the effectiveness of Random Forest in ransomware detection when using a 50:50 dataset split with 10,000 samples.

In conclusion, Random Forest shows best accuracy in most scenario of all train and test ratio (10:90, 30:70, 50:50, 70:30, 90:10). This result aligns with previous research by (Almomani et al., 2021), (Alsoghyer et al., 2020) and (Ahmed et al., 2020) which achieve Random Forest as the best classification technique in their ransomware domain research. According to,

## 6.6 Significant Results

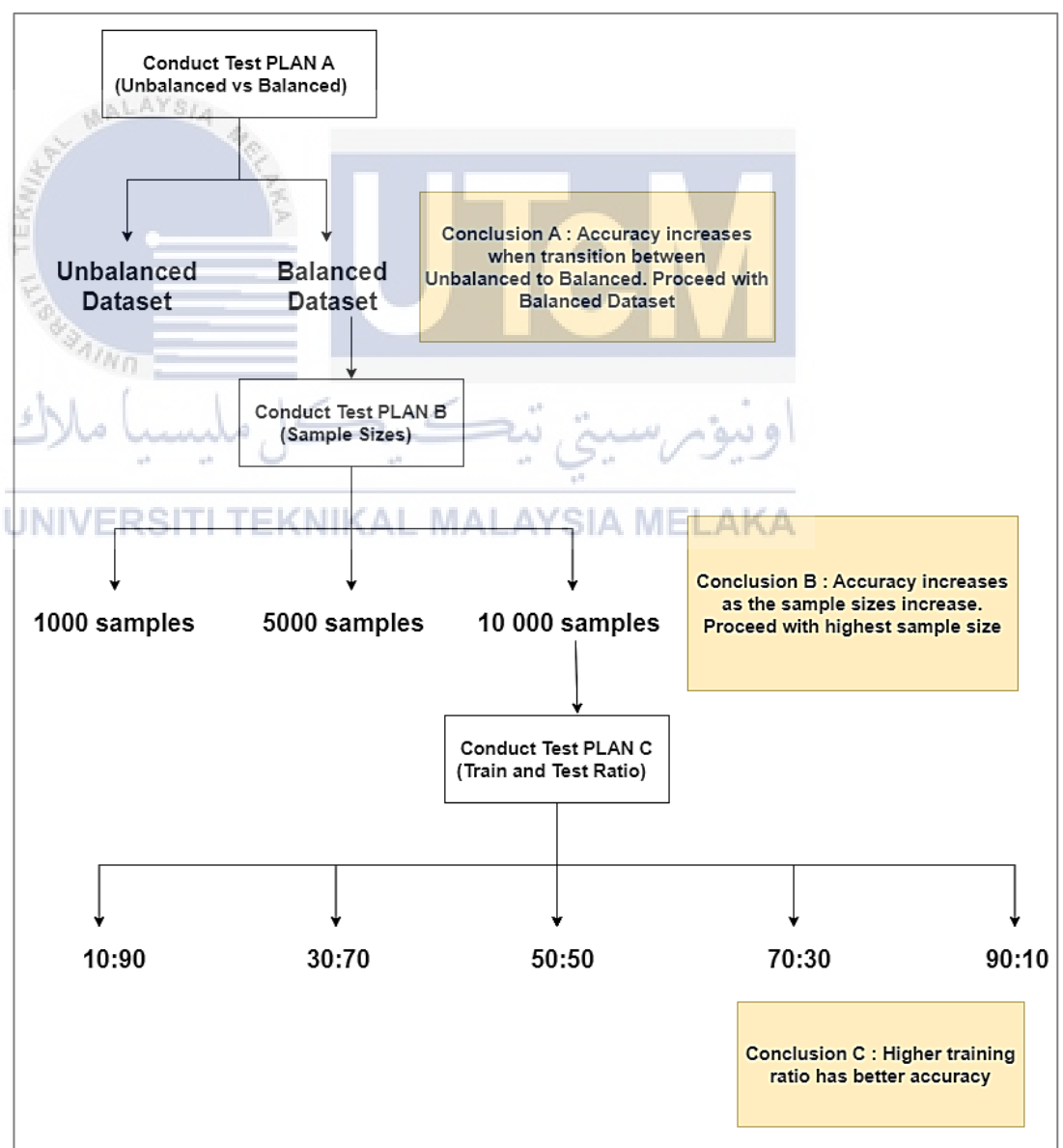


Figure 6.26 Overall Test Plan Findings

Figure 6.26 shows the conclusion based on the outcomes of each test plan. By conducting test plan A, we can conclude the classification algorithms increases when transition from unbalanced to balanced dataset. This is supported by (Mooijman et al.,2023) in his research “The effects of data balancing approaches” which states that, imbalance dataset tends to influence classifiers, causing them to prioritize the majority class when performing classification tasks thereby affecting the performance. In this context, imbalance dataset refers to a dataset with a significant difference in the number of samples among its classifications, as stated by (Dehkordy,2021).

In addition to that, there has been research using the same method for balanced cased (Alsoghyer et al., 2020), (Almomani et al., 2021), and (Mercaldo, 2021). However, previous authors only consider 1000 balanced samples precisely 500 Ransomware samples and 500 Benign samples. Therefore, we were able to produce a better accuracy 99.30% by expanding the balanced sample size up to 10 000 samples which indirectly justified our Test Plan B. Not only that, according to (Ajiboye et al., 2015) in the paper titled "Evaluating The Effect Of Dataset Size On Predictive Model Using Supervised Learning Technique," it is evident that enhancing the results can be achieved through dataset size which supports our findings.

For test Plan C, as suggested by (Dobbin and Simon, 2011) the recommended amount of training and testing ratio is 70:30. By expanding the ratio, 10:90, 30:70, 50:50, 70:30 and 90:10 we were able to identify Random Forest as the best classification techniques for training ratio 70% and above with 99.30% accuracy. This result aligns with previous research by (Almomani et al., 2021), (Alsoghyer et al., 2020) and (Ahmed et al., 2020) which achieve Random Forest as the best classification technique in their ransomware domain research.

*Table 6.21 Significant Results for comparison of previous research  
Ransomware (R)=Malicious (M), Benign (B)=White (W)*

Research	Train Test Ratio	Studied Dataset	Balance d	Accuracy
(Almomani et al., 2021)	N/A	500 R 500 B	Yes	98.30%
(Alsoghyer, S, 2020)	N/A	500 R 500 B	Yes	96.90%
(Khammas, B. M. 2020)	50:50	840 R 840B	Yes	97.74%
(Alzahrani et al., 2015)	N/A	100R 200 B	No	91.00%
(Popryho, 2023)	N/A	1056 R 399 B	No	97.00%
(Coronado-De-Alba et al., 2017)	N/A	1531 M 765 B	No	97.56%
(Victoriano, 2019)	N/A	668 R 1255 B	No	98.05%
<b>Proposed Work Dataset I</b>	70:30	5000 R 5000 W	Yes	98.67%
	90:30	5000 R 5000 W	Yes	99.10%
<b>Proposed Work Dataset II</b>	70:30	5000 R 5000 B	Yes	78.20%
	90:10	5000 R 5000 B	Yes	78.60%



<b>Proposed Work Dataset III</b>	70:30	5000 R 5000 B	Yes	<b>99.30%</b> <b>(Best)</b>
	90:10	5000 R 5000 B	Yes	<b>99.30%</b> <b>(Best)</b>

Based on table 6.17, in comparison to previous studies, the study presented in this research marks significant improvements in the field of ransomware detection. It can be highlighted, utilization of balanced datasets with higher number of samples as justified previously able to show outstanding accuracy rates of up to 99.30%. From the result, most machine learning classification models achieve better performance for testing ratio starting with 70% and above. In conclusion, expanding the test set to encompass three distinct aspects is a commendable approach, enabling an in-depth investigation into the performance behavior across various scenarios.

## 6.7 Summary

In this chapter, an in-depth examination of ransomware detection using various machine learning algorithms is performed on the BitcoinHeist Ransomware dataset (Dataset I), Android Ransomware dataset (Dataset II) and File System Behavior Ransomware Detection dataset (Dataset III). In addition to that, various aspects of testing has been expanded and covered in this research such as in aspect of the a) various dataset sizes, b) unbalanced versus balanced datasets, and c) various training and test ratios. The aim of expanding the scope is to study the behavior of the ransomware detection model under a variety of scenarios. To sum up, our test plan has shown that balanced datasets, a sample size of 10,000, and a training ratio of 70% and above consistently yield the best Random Forest (RF) results, achieving an impressive accuracy rate of 99.30%. Random Forest has also shown as the best classification model with most number of occurrences in both WEKA and Orange for all datasets.

## CHAPTER 7: CONCLUSION

### 7.0 Introduction

The previous chapter focuses on testing and evaluation of the dataset. The machine learning model performance has been evaluated and discussed according to the evaluation metrics in the previous chapter. As a result, this chapter will be the final chapter for the project conclusion. This chapter will go through the project summary, project contributions, project limitations, and future work to improve the current research. This concluding chapter holds significant importance as it provides a comprehensive overview of our project, "Analysis of Ransomware Detection Based on Machine Learning Approach." Its primary function is to serve as a practical guide for future researchers aiming to refine and build upon our model by summarizing its limitations and offering suggestions for enhancement.

## 7.1 Project Summarization

This project has been designed to analyze the performance of various machine learning classification models for ransomware detection. Overall, our research has successfully accomplished all three stated objectives, providing significant insights into the area of ransomware detection. Our first objective, which was to study classification techniques for ransomware detection, has been accomplished by an extensive review of the existing literature in the area of interest. We thoroughly researched and analyzed a wide variety of ransomware detection methods, getting insights into their strengths, limitations, and applicability for different types of situations. In response to our first objective, we discovered that a variety of machine learning classification models have been used for ransomware detection, including Decision Tree, Random Forest, Support Vector Machines (SVM) and Naïve Bayes.

During the execution of the second objective, valuable experience in the practical application of classification techniques to ransomware dataset is gained. The hands-on process involved extensive data preprocessing conducted through Jupiter Notebook, providing an opportunity to learn and apply the Python programming language for data preparation. Furthermore, we developed and executed a comprehensive test plan that included three critical components, all of which were carried out by employing the WEKA and Orange platforms.

To address our third objective a comprehensive evaluation process was conducted using various evaluation metrics tools. The accuracy of each classification technique was extensively assessed, taking into account metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F-measure, and Overall Accuracy. As conclusion, it's shown that balanced datasets, a sample size of 10,000, and a training ratio of 70% and above consistently yield the best Random Forest (RF) results, achieving an impressive accuracy rate of 99.30%.

## 7.2 Project Contributions

The results of the test in Chapter 6 provided in this research paper demonstrate that, the proposed approaches outperform the existing research in the same domain. A comprehensive review of this research is carried out to further highlight their accuracy and effectiveness. We've not only implemented and tested various ransomware detection datasets category (Dataset I, II, III), as showcased in the results of Chapter 6. We've also conducted a meticulous comparative analysis which is inspired from (Ajiboye et al., 2015) research on dataset size's impact on predictive models using supervised learning techniques, exploring various testing scenarios: a) unbalance vs balanced dataset b) various sample sizes c) various ratio of training and test.

In our research, we discovered an interesting trend in which performance constantly improves as dataset size increases using the balanced scenario. This occurrence occurs across all three datasets, which are Dataset I, Dataset II, and Dataset III. In addition to that, another interesting finding is that certain machine learning algorithms, such as Random Forest, have a remarkable performance in its accuracy while migrating from imbalanced to balanced datasets. It also can be highlighted that, most machine learning classification models achieve better performance for training ratio starting with 70% and above.

This significant improvement highlights the ability of these algorithms to efficiently adapt to varied data distributions, which is critical in the domain of ransomware detection. Furthermore, we managed idenn. Furthermore, we were successful in finding a solution for heap size constraints encountered during the execution of machine learning algorithms. We constructed an effective sub-sampling approach by using the capability of Jupyter Notebook in the Python programming language.

## 7.3 Project Limitations

Several challenges occurred throughout the span of our research, including heap size constraints in WEKA during the execution of several machine learning algorithms. Although we have tried to maximize heap size (up to 8GB) WEKA by overriding the initial value using command

%JAVA\_OPTS% -Xms8192m. Heap size successfully changed but error persist on WEKA. Subsequent problems have led us to explore sub-sampling as a potential solution to WEKA-related problems. Another aspect to consider is data processing and model training times increased with larger datasets and more complex attributes. This can be shown as we were only able to evaluate a maximum of 10 000 samples which WEKA took more than 2 hours to generate the results. Therefore consideration needs to be given, to address these computational constraints that would allow for the exploration of even larger datasets and more intricate machine learning models.

#### 7.4 Future Work

Future work should broaden the scope of our research in order to further expand on the project. Most importantly, expanding the dataset size is critical for improving model resilience. A broader range of classifiers and tools that are more suitable should also be studied to increase the diversity of techniques. To be specific as we only focus on the supervised machine learning technique, future researchers may also explore deep learning methods, browser plugin development, zero-day ransomware detection and most importantly the utilization of larger datasets to advance ransomware detection.

## 7.5 Summary

In conclusion, our study has shown significant findings in the field of ransomware detection, indicating a substantial advancement above typical antivirus systems that rely on signature-based detection approaches. This research has been constructed with the objective of investigating various classification techniques for the analysis of ransomware. The second objective has been achieved by applying the classification techniques to Ransomware dataset after preprocessing conducted through Jupiter Notebook. The third objective has been addressed by evaluating the accuracy result of classification techniques using different evaluation metrics tools.

The test is executed repeatedly to study the behavior of the classifications model performance under different conditions such as, balanced vs unbalanced, various dataset size and ratios. Based on the results it can be concluded that Random Forest performs as the best classification techniques for both WEKA and Orange with most number of occurrences. This research can serve as a guide for future studies and improvement of applications in ransomware detection.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## REFERENCES

- A. Khalil, & M. Khammas, B. (2022). An effective and efficient features vectors for ransomware detection via machine learning technique. *Iraqi Journal of Information and Communication Technology*, 5(3), 23–33. Accessed 20 March 2023, <https://doi.org/10.31987/ijict.5.3.205>
- Antal, G. (2023, February 15). Lockbit ransomware: Here's what you need to know. *Heimdalsecurity Blog*. Accessed 20 March, <https://heimdalsecurity.com/blog/what-is-lockbit-ransomware/>
- Abbasi, M. S. (2023). Automating Behavior-Based Ransomware Analysis, Detection, and Classification Using Machine Learning. Accessed 10 May 2023, <https://doi.org/10.26686/wgtn.22180858>
- Abdullah, Z., Muhadi, F. W., Saudi, M. M., Hamid, I. R., & Foozy, C. F. (2019). Android Ransomware Detection Based on Dynamic Obtained Features. *Advances in Intelligent Systems and Computing*, 121–129. Accessed 18 September 2023, [https://doi.org/10.1007/978-3-030-36056-6\\_12](https://doi.org/10.1007/978-3-030-36056-6_12)
- Ahmed, Y. A., Koçer, B., Huda, S., Saleh Al-rimy, B. A., & Hassan, M. M. (2020). A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection. *Journal of Network and Computer Applications*, 167, 102753. Accessed 14 May 2023, <https://doi.org/10.1016/j.jnca.2020.102753>
- Al-Haija, Q. A., & Alsulami, A. A. (2021). High performance classification model to identify ransomware payments for heterogeneous bitcoin networks. *Electronics*, 10(17), 2113. Accessed 10 May 2023, <https://doi.org/10.3390/electronics10172113>
- Alalousi, A., Razif, R., AbuAlhaj, M., Anbar, M., & Nizam, S. (2016). A preliminary performance evaluation of K-means, KNN and EM unsupervised machine learning methods for network flow classification. *International Journal of Electrical and Computer Engineering (IJECE)*, 6(2), 778. Accessed 20 May 2023, <https://doi.org/10.11591/ijece.v6i2.pp778-784>
- Ajiboye, A. R., Abdullah-Arshah, R., Qin, H., & Isah-Kebbe, H. (2015). Evaluating the effect of dataset size on predictive model using supervised learning technique. *International Journal of Computer Systems & Software Engineering*, 1(1), 75–84. Accessed 10 August 2023, <https://doi.org/10.15282/ijsecs.1.2015.6.0006>
- Alzahrani, A., Alshehri, A., Alshahrani, H., Alharthi, R., Fu, H., Liu, A., & Zhu, Y. (2018). Randroid: Structural similarity approach for detecting ransomware applications in Android platform. 2018 IEEE International Conference on

Electro/Information Technology (EIT). Accessed 9 September 2023, <https://doi.org/10.1109/eit.2018.8500161>

Alsoghyer, S., & Almomani, I. (2020). On the Effectiveness of Application Permissions for Android Ransomware Detection. 2020 6th Conference on Data Science and Machine Learning Applications (CDMA). Accessed 18 September 2023, <https://doi.org/10.1109/cdma47397.2020.00022>

Almomani, I., AlKhayer, A., & Ahmed, M. (2021). An Efficient Machine Learning-based Approach for Android v.11 Ransomware Detection. 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA). Accessed 18 September 2023, <https://doi.org/10.1109/caida51941.2021.9425059>

Almomani, I., Alkhayer, A., & El-Shafai, W. (2023). E2E-RDS: Efficient End-to-End Ransomware Detection System Based on Static-Based ML and Vision-Based DL Approaches. *Sensors*, 23(9), 4467. Accessed 18 September 2023, <https://doi.org/10.3390/s23094467>

Bensalah, A. (2022, July 31). Ransomware detection data set. Kaggle. Accessed 4 May 2023, <https://www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set?resource=download>

Belcic, I. (2022, May 19). What is CryptoLocker ransomware and how to remove it. Accessed 10 May 2023, <https://www.avast.com/c-cryptolocker>

Chakraborty, S. (2023). Android Ransomware Detection [Data set]. Kaggle. Accessed 4 May 2023, <https://doi.org/10.34740/KAGGLE/DSV/4987535>

Cusack, G., Michel, O., & Keller, E. (2018). Machine learning-based detection of ransomware using SDN. *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. Accessed 26 May 2023, <https://doi.org/10.1145/3180465.3180467>

Chumachenko, K. (2017). Machine Learning Methods for Malware Detection and Classification. *Proceedings of the 21st Pan-Hellenic Conference on Informatics - PCI 2017*, 93

Chaudhuri, K. D. (2022, March 25). Building naive Bayes classifier from scratch to perform sentiment analysis. *Analytics Vidhya*. Accessed 26 May 2023, <https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/>

Coronado-De-Alba, L. D., Rodriguez-Mota, A., & Escamilla-Ambrosio, P. J.



- (2016). Feature selection and ensemble of classifiers for android malware detection. 2016 8th IEEE Latin-American Conference on Communications (LATINCOM). Accessed 9 September 2023, <https://doi.org/10.1109/latincom.2016.7811605>
- Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. *BMC Medical Genomics*, 4(1). Accessed 6 June 2023, <https://doi.org/10.1186/1755-8794-4-31>
- Dehkordy, D. T., & Rasoolzadegan, A. (2021). A new machine learning-based method for android malware detection on imbalanced dataset. *Multimedia Tools and Applications*. Accessed 19 September 2023, <https://doi.org/10.1007/s11042-021-10647-z>
- El Naqa, I., & Murphy, M. J. (2015). What is machine learning? *Machine Learning in Radiation Oncology*, 3–11. Accessed 20 May 2023, [https://doi.org/10.1007/978-3-319-18305-3\\_1](https://doi.org/10.1007/978-3-319-18305-3_1)
- Fedor, O. (2022, November 3). 93 must-know ransomware statistics. *AntivirusGuide*. Accessed 20 March 2023, <https://www.antivirusguide.com/cybersecurity/ransomware-statistics/>
- Fernando, D. W., Komninos, N., & Chen, T. (2020). A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques. *IoT*, 1(2), 551–604. Accessed 10 May 2023, <https://doi.org/10.3390/iot1020030>
- Gagulic, D., Lynn Zumtaugwald, & Siddhant Sahu. (February 2023). Ransomware Detection with Machine Learning in Storage Systems. Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland. Accessed 20 March 2023, <https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/map-gagulic-zumtaugwald-sahu.pdf>
- Gupta, A. (2023, January 30). An introduction to scikit-learn: Machine learning in Python. *Simplilearn.com*. Retrieved from Accessed 20 May 2023, <https://www.simplilearn.com/tutorials/python-tutorial/scikit-learn>
- Herrera Silva, J. A., Barona López, L. I., Valdivieso Caraguay, Á. L., & Hernández-Álvarez, M. (2019). A survey on situational awareness of ransomware attacks—detection and prevention parameters. *Remote Sensing*, 11(10), 1168. Accessed 10 May 2023, <https://doi.org/10.3390/rs11101168>
- Horduna, M., Lăzărescu, S.-M., & Simion, E. (2023). A note on machine learning applied in ransomware detection. Accessed 20 March 2023, <https://eprint.iacr.org/2023/045>

- Humayun, M., Jhanjhi, N. Z., Alsayat, A., & Ponnusamy, V. (2021). Internet of things and Ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1), 105–117. Accessed 10 May 2023, <https://doi.org/10.1016/j.eij.2020.05.003>
- Javatpoint. (2023). How to get datasets for Machine Learning. Accessed 13 May 2023 <https://www.javatpoint.com/how-to-get-datasets-for-machine-learning>
- Ilascu, Ionut. (2022, May 19). Ransomware gangs rely more on weaponizing vulnerabilities. *BleepingComputer*. Accessed May 5 2023, <https://www.bleepingcomputer.com/news/security/ransomware-gangs-rely-more-on-weaponizing-vulnerabilities/>
- Ibrahim, S., Herami, N. A., Naqbi, E. A., & Aldwairi, M. (2020). Detection and analysis of drive-by downloads and malicious websites. *Communications in Computer and Information Science*, 72–86. Accessed 20 May 2023, [https://doi.org/10.1007/978-981-15-4825-3\\_6](https://doi.org/10.1007/978-981-15-4825-3_6)
- Jainani, P. (2021, April 30). Azure machine learning service - part 1: An introduction. *Medium*. Accessed 15 May 2023, <https://towardsdatascience.com/azure-machine-learning-service-part-1-an-introduction-739620d1127b>
- Kharraz, A., Robertson, W., & Kirda, E. (2018). Protecting against ransomware: A new line of research or restating classic ideas? *IEEE Security & Privacy*, 16(3), 103–107. Accessed 10 May 2023, <https://doi.org/10.1109/msp.2018.2701165>
- Kok, S. H., Azween, A., & Jhanjhi, N. (2020). Evaluation metric for crypto-ransomware detection using machine learning. *Journal of Information Security and Applications*, 55, 102646. Accessed 10 May 2023, <https://doi.org/10.1016/j.jisa.2020.102646>
- Kaspersky. (2023, April 19). What is ransomware. Accessed 10 May 2023, <https://www.kaspersky.com/resource-center/threats/ransomware>
- Khalil, N. A., & M. Khammas, B. (2022). An effective and efficient feature vector for ransomware detection via machine learning techniques. *Iraqi Journal of Information and Communication Technology*, 5(3), 23–33. Accessed 20 May 2023, <https://doi.org/10.31987/ijict.5.3.205>
- Khammas, B. M. (2020). Ransomware detection using random forest technique. *ICT Express*, 6(4), 325–331. Accessed 20 May 2023, <https://doi.org/10.1016/j.ict.2020.11.001>
- Kaspersky IT Encyclopedia. (2023). What is RaaS (Ransomware-as-a-Service)? Accessed 20 March 2023, <https://encyclopedia.kaspersky.com/glossary/ransomware-as-a-service-raas/>

- Kizito, N. (2022, March 28). Parameters and hyperparameters in machine learning and deep learning. Medium. Accessed 20 May 2023, <https://towardsdatascience.com/parameters-and-hyperparametersaa609601a9ac#:~:text=Simply%20put%2C%20parameters%20in%20machine,choice%20of%20hyperparameters%20you%20provide.>
- Lee, J., Lee, J., & Hong, J. (2017). How to make efficient decoy files for ransomware detection? Proceedings of the International Conference on Research in Adaptive and Convergent Systems. Accessed 20 May 2023, <https://doi.org/10.1145/3129676.3129713>
- Madani, H., Ouerdi, N., & Azizi, A. (2023). Ransomware: Analysis of Encrypted Files. *International Journal of Advanced Computer Science and Applications*, 14(1). Accessed 20 March 2023, <https://doi.org/10.14569/ijacsa.2023.0140124>
- Maigida, A. M., Abdulhamid, S. M., Olalere, M., Alhassan, J. K., Chiroma, H., & Dada, E. G. (2019). Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms. *Journal of Reliable Intelligent Environments*, 5(2), 67–89. Accessed 10 May 2023, <https://doi.org/10.1007/s40860-019-00080-3>
- Mercaldo, F. (2021). A framework for supporting ransomware detection and prevention based on hybrid analysis. *Journal of Computer Virology and Hacking Techniques*, 17(3), 221–227. Accessed 18 September 2023, <https://doi.org/10.1007/s11416-021-00388-w>
- Microsoft Azure. (2023). Artificial Intelligence vs. Machine Learning. Accessed 6 May 2023, <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/artificial-intelligence-vs-machine-learning/#introduction>
- Mohammad, A. H. (2020). Analysis of Ransomware on Windows Platform. Accessed 20 March 2023, <https://doi.org/10.13140/RG.2.2.11150.59202>
- Moussaileb, R., Navas, R. E., & Cuppens, N. (2020). Watch out! Doxware on the way. *Journal of Information Security and Applications*, 55, 102668. Accessed 10 May 2023, <https://doi.org/10.1016/j.jisa.2020.102668>
- Mooijman, P., Catal, C., Tekinerdogan, B., Lommen, A., & Blokland, M. (2023). The effects of data balancing approaches: A case study. *Applied Soft Computing*, 132, 109853. Accessed 18 September 2023, <https://doi.org/10.1016/j.asoc.2022.109853>

- Muslim, A. K., Mohd Dzulkifli, D. Z., Nadhim, M. H., & Abdellah, R. H. (2019). A study of ransomware attacks: Evolution and prevention. *Journal of Social Transformation and Regional Development*, 1(1). Accessed 20 May 2023, <https://doi.org/10.30880/jstard.2019.01.01.003>
- Mahajan, G., Saini, B., & Anand, S. (2019). Malware classification using machine learning algorithms and Tools. 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP). Accessed 10 May 2023, <https://doi.org/10.1109/icaccp.2019.8882965>
- Norouzi, M., Souri, A., & Samad Zamini, M. (2016). A data mining classification approach for behavioral malware detection. *Journal of Computer Networks and Communications*, 2016, 1–9. Accessed 10 May 2023, <https://doi.org/10.1155/2016/8069672>
- Prakash, K. B., Kannan, R., Alexander, S. A., & Kanagachidambaresan, G. R. (2021). Advanced deep learning for engineers and scientists. *EAI/Springer Innovations in Communication and Computing*. Accessed 10 May 2023, <https://doi.org/10.1007/978-3-030-66519-7>
- P. Fabian, V. Gael, and G. Alexandre (2012), “Scikit-learn: machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. Accessed 26 May 2023, <https://doi.org/10.48550/arXiv.1201.0490>
- Padmavaty, V., Geetha, C., & Priya, N. (2020). Analysis of data mining tool Orange. *International Journal of Modern Agriculture*, 9(4), 1146-1150. Retrieved from Accessed 20 May 2023, <http://www.modern-journals.com/index.php/ijma/article/view/485>.
- Popryho, Y., (2023). Behaviour-based detection of ransomware attacks in the Cloud using machine learning. Accessed 19 September 2023, <https://www.diva-portal.org/smash/get/diva2:1773681/FULLTEXT02.pdf>
- Qolomany, B., Al-Fuqaha, A., Gupta, A., Benhaddou, D., Alwajidi, S., Qadir, J., & Fong, A. C. (2019). Leveraging machine learning and big data for Smart Buildings: A comprehensive survey. *IEEE Access*, 7, 90316–90356. Accessed 20 May 2023, <https://doi.org/10.1109/access.2019.2926642>
- Salvi, H. U. (2019). Raas ransomware-as-a-service. *International Journal of Computer Sciences and Engineering*, 7(6), 586–590. Accessed 10 May 2023, <https://doi.org/10.26438/ijcse/v7i6.586590>
- Smith, D., Khorsandroo, S., & Roy, K. (2022). Machine learning algorithms and frameworks in ransomware detection. *IEEE Access*, 10, 117597–117610. Accessed 4 May 2023, <https://doi.org/10.1109/access.2022.3218779>
- Security Intelligence. (2023, March 15). Costa Rica State of emergency declared after

ransomware attacks. Accessed 20 March 2023, <https://securityintelligence.com/news/costa-rica-state-emergency-ransomware/>

Sgandurra, D., Muñoz-González, L., Mohsen, R., & Lupu, E. C. (2016). Automated Dynamic Analysis of Ransomware: Benefits, Limitations and Use for Detection. arXiv preprint arXiv:1609.03020. Accessed 4 May 2023, <https://arxiv.org/abs/1609.03020>

Srivastava, S. (2014). Weka: A tool for data preprocessing, classification, Ensemble, clustering and association rule mining. *International Journal of Computer Applications*, 88(10), 26–29. Accessed 6 June 2023, <https://doi.org/10.5120/15389-3809>

Sipra, V. (2021, May 14). Machine learning for newbies. Medium. Retrieved from Accessed 20 May 2023, <https://towardsdatascience.com/machine-learning-for-newbies-7dd33dd6b764>

Sharma, S., Rama Krishna, C., & Sahay, S. K. (2018). Detection of advanced malware by Machine Learning Techniques. *Advances in Intelligent Systems and Computing*, 333–342. Accessed 10 May 2023, [https://doi.org/10.1007/978-981-13-0589-4\\_31](https://doi.org/10.1007/978-981-13-0589-4_31)

Thampi, S., Perez, G., Ko, R., & Rawat, D. B. (2020). Security in computing and communications. *Communications in Computer and Information Science*. Accessed 10 May 2023, <https://doi.org/10.1007/978-981-15-4825-3>

Victoriano, O. B. (2019). Exposing Android ransomware using machine learning. *Proceedings of the 2019 International Conference on Information System and System Management*. Accessed 9 September 2023, <https://doi.org/10.1145/3394788.3394923>

Malwarebytes.(2023). What is a brute force attack. Accessed 10 May 2023, <https://www.malwarebytes.com/cybersecurity/business/brute-force-attack>

Yusof, R., Adnan, N. S., Abd. Jalil, N., & Abdullah, R. S. (2019). Analysis of data mining tools for Android malware detection. *Journal of Advanced Computing Technology and Application (JACTA)*, 1(2), 21-24. Accessed 30 May 2023, <https://jacta.utm.edu.my/jacta/article/view/5196>

## APPENDIX A

Research	Type of Tools	Type of Algorithms	Type of Techniques	Problems + Objective	Advantages	Limitation
(Kok et al., 2020)	<ul style="list-style-type: none"> <li>• PEDA (Pre-encryption detection algorithm (PEDA))</li> <li>• Cuckoo for generating datasets</li> </ul>	<ul style="list-style-type: none"> <li>• Random Forest</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> 1) How to detect presence of crypto-ransomware before any encryption occurs?</p> <p><b>O:</b> 1) To propose development of pre-encryption detection algorithm (PEDA) for early detection of crypto-ransomware.</p> <p>2) To propose new metrics for the evaluation of a predictive model used in ransomware detection.</p>	Combination of two detection level which is signature repository(SR) and learning algorithm (LA) enables PEDA to detect known crypto-ransomware faster and also detect similar behavioural crypto-ransomware with unknown signature.	The proposed metrics include PLR, NNM, DOR, J Index, and NND is difficult to represent into one graph, as proposed metrics (DOR,PLR) may have an infinite value. This may result in a misunderstanding of the true metric value.
(Khalil et al., 2022)	<ul style="list-style-type: none"> <li>• WEKA</li> <li>• MATLAB</li> </ul>	<ul style="list-style-type: none"> <li>• Support Vector Machines (SVM)</li> <li>• K-Nearest Neighbors (KNN)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> 1) How can static analysis be used to overcome the constraints of dynamic analysis in order to construct a detection model?</p>	According to the experimental result, the Random Forest achieved the highest detection accuracy compared to others.	This method is not extensively explored in the current existing literature, so it's recommended that the future study to concentrate on the development of a revolutionary static analysis-based approach for



		<ul style="list-style-type: none"> <li>• Random Forest (RF)</li> <li>• Logistic Regression (LR)</li> <li>• Naive Bayes (NB)</li> </ul>		<p><b>O:</b> 1) To propose a new technique based on static analysis for detecting and classifying ransomware utilising five machine learning algorithms.</p>		identifying and distinguishing ransomware.
(Abbasi, 2023)	<ul style="list-style-type: none"> <li>• Cuckoo Sandbox</li> <li>• Tensorflow</li> </ul>	<ul style="list-style-type: none"> <li>• Regularized Logistic Regression (RLR)</li> <li>• Random Forest (RF)</li> <li>• Decision Tree (DT)</li> <li>• Support Vector Machines (SVM)</li> <li>• k-Nearest Neighbors (KNN)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How can the challenges of high-dimensional data and time-intensive manual inspection in behavior-based ransomware detection be overcome?</p> <p><b>O:</b> To propose a new representation of API call sequences, for early ransomware detection.</p>	<p>The research proof that identifying critical call arguments alongside API call names in sequences can help improve the classification performance.</p>	<p>The scope of the analysis environment is limited. The research excludes ransomware that targets multiple operating systems such as Linux and Mac, as well as devices such as mobile phones. As a result, the findings are limited to ransomware compatible with Windows 7 PCs only.</p>
(Khammas, 2020)	<ul style="list-style-type: none"> <li>• WEKA</li> </ul>	<ul style="list-style-type: none"> <li>• Random Forest (RF)</li> </ul>	<ul style="list-style-type: none"> <li>• Classification</li> </ul>	<p><b>P:</b> How can we overcome the issue of complicated disassemble process when detecting ransomware attacks?</p> <p><b>O:</b> To propose a new method of ransomware detection using Random Forest technique based on static analysis.</p>	<p>According to the research result it shows good performance of random forest classifier with the byte level static analysis for ransomware attack detection</p>	<p>The classification time is directly proportional with increasing in tree numbers. Therefore, it can be complex to determine the best number of tree that provides high accuracy with acceptable time for classification.</p>

(Al-Haija et al., 2021)	MATLAB	<ul style="list-style-type: none"> <li>• shallow neural networks (SNNs)</li> <li>• optimizable decision trees (ODT)</li> </ul>	Classification	<p>P: How to identify and detect ransomware attacks in early detection of bitcoin transaction.</p> <p>O: To develop a predictive system that can classify ransomware payments for heterogeneous bitcoin networks.</p>	Produced high performance classification model	Lack of in-depth analysis of quality measure for future ML development especially dealing with imbalanced dataset.
(Ibrahim, et al., 2020)	WEKA Orange Scikit	<ul style="list-style-type: none"> <li>• NaiveBayse</li> <li>• JRip</li> <li>• J48</li> </ul>	Classification	<p>P: How to address the challenges and problems associated with malicious website detection.</p> <p>O: To produce solutions for feature selection in machine learning for drive-by download problem.</p>	The research is able to distinguish benign and malicious websites through URL-based analysis. The suggested approach focuses on protecting users against browser vulnerability-related attacks.	The research has limitations including a small scope of datasets. The research also uses limited classifier which in WEKA.



## APPENDIX B

### a) Steps to classify data for Random Forest, Support Vector Machines (SVM) and Naïve Bayes in WEKA.

- 1) Navigate to the “Classify” tab as shown in Figure 1 below. Under the “Classifier” section, click choose to select the machine learning classification algorithms.

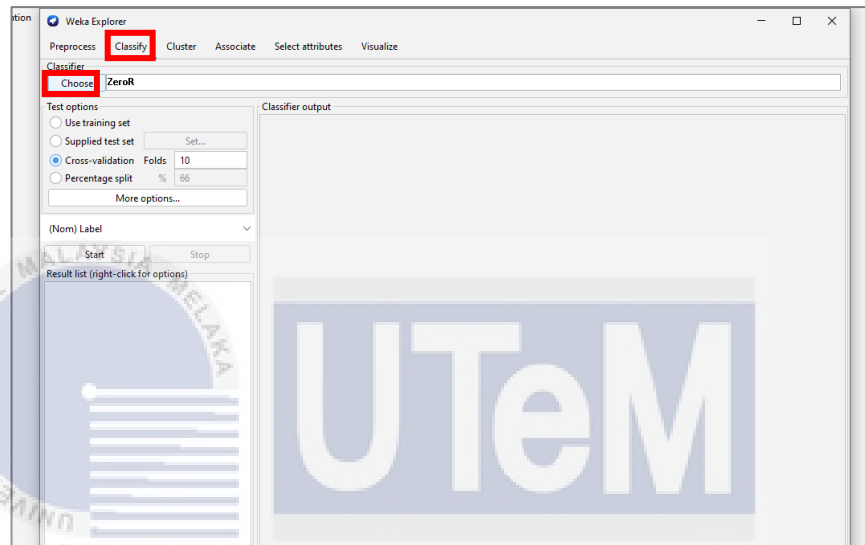


Figure 1 WEKA explorer

- 2) To implement the **Random Forest classification algorithms**, choose the “RandomForest” under the trees section of the classifier as shown in Figure 2 below.

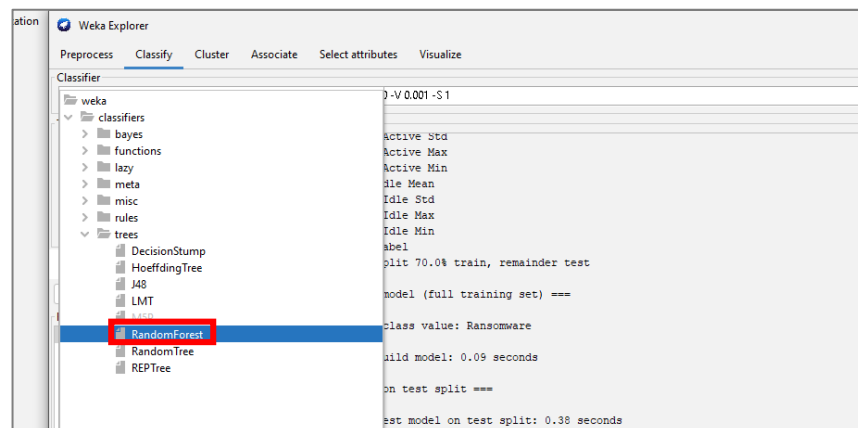
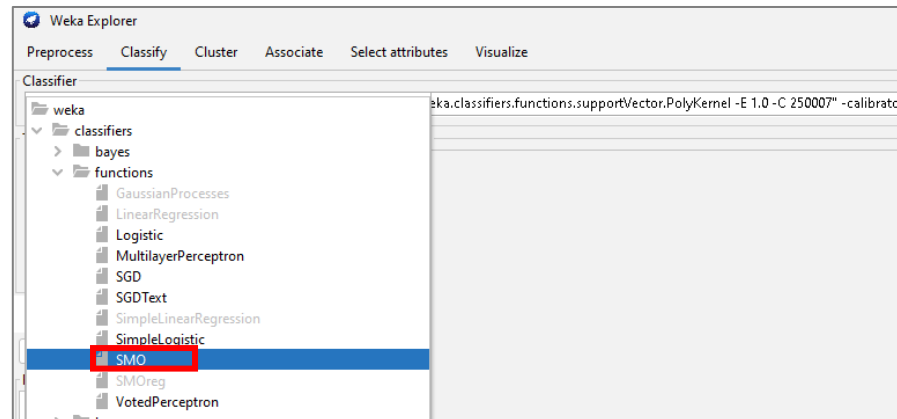


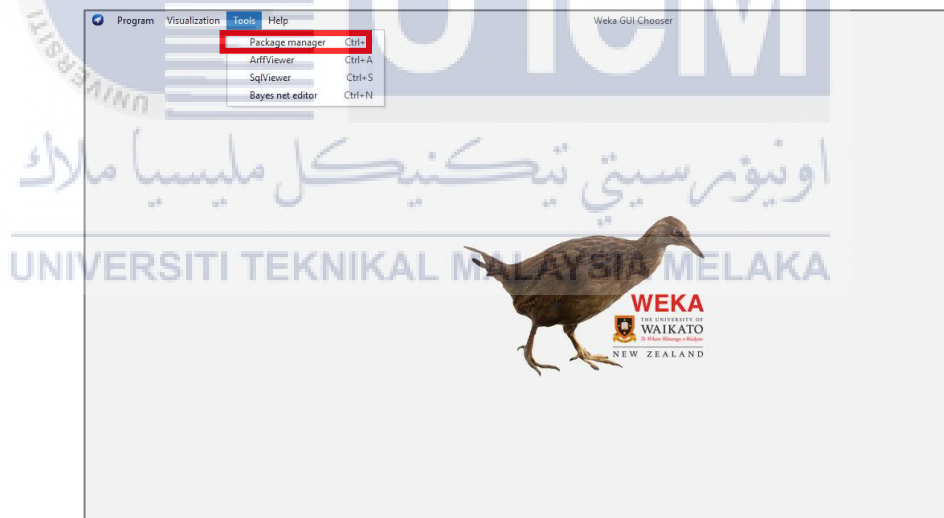
Figure 2 Selection of Random Forest in WEKA

- 3) To implement the **Support Vector Machines (SVM) classification algorithms**, there are two options available which are using the default package SMO or the external package LibSVM. To implement the SMO click “SMO” under the functions section of the classifier as shown in Figure 3 below.



*Figure 3 Selection of Support Vector Machines (SVM) in WEKA using SMO*

- 4) For the LibSVM, we have to install the external package first. Navigate to tools as shown in Figure 4 below. Select the Package manager.



*Figure 4 Package Manager in WEKA*

- 5) Search LibSVM and click install to start downloading the package as shown in Figure 5 below. Close any open WEKA application windows before proceeding with the installation.

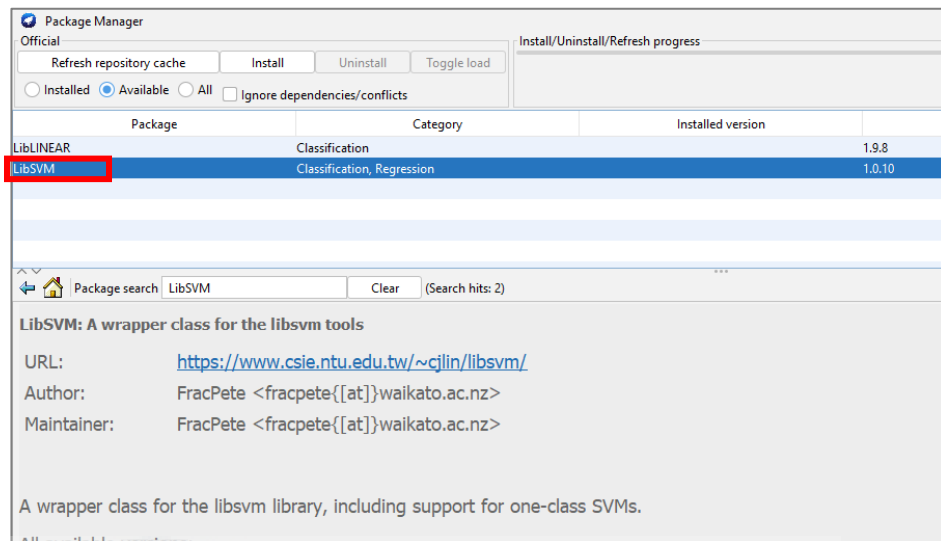


Figure 5 Installation of LibSVM in WEKA

- 6) To implement the Support Vector Machines (SVM) classification algorithms using the LibSVM, select the LibSVM under the functions section as shown in Figure 6 below.

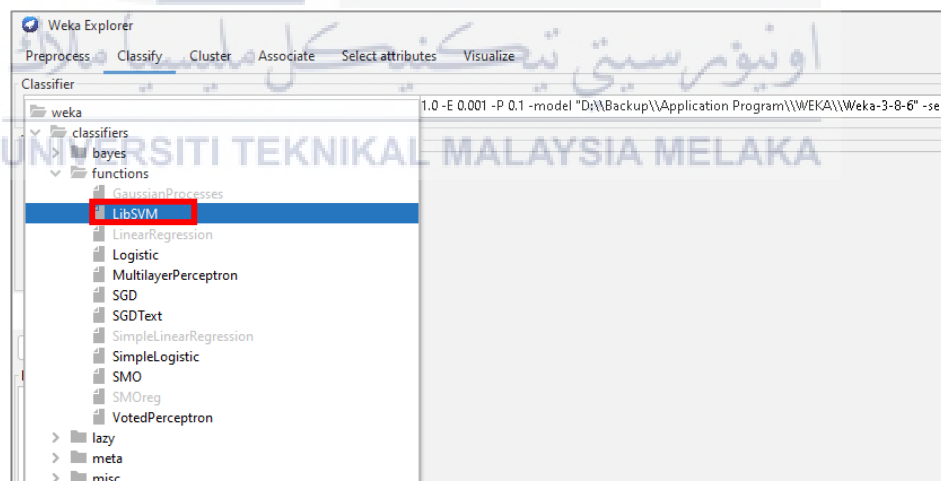


Figure 6 Selection of Support Vector Machines (SVM) in WEKA using LibSVM

- 7) To implement the **Naïve Bayes classification algorithms**, choose the “NaïveBayes” under the bayes section of the classifier as shown in Figure 7 below.

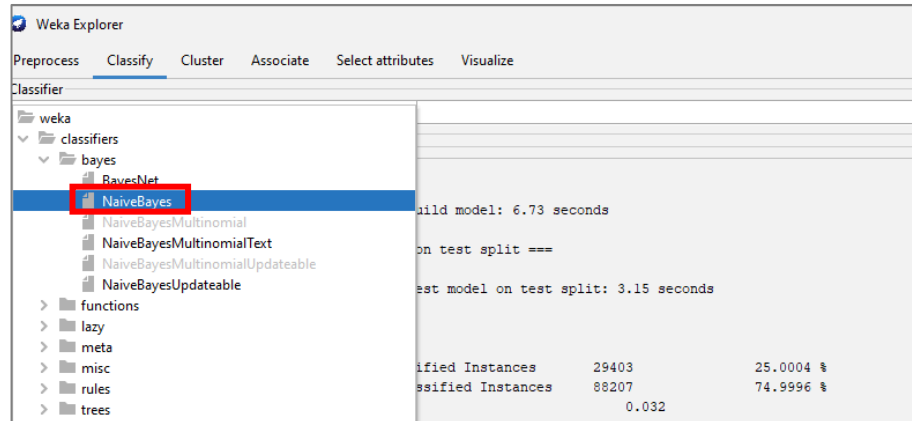


Figure 7 Selection of Naïve Bayes in WEKA

**b) Steps to classify data for Random Forest, Support Vector Machines (SVM) and Naïve Bayes in Orange.**

- 1) To implement the Random Forest, select the “Random Forest” widget which represents the Random Forest classification algorithm as shown in Figure 8 below.

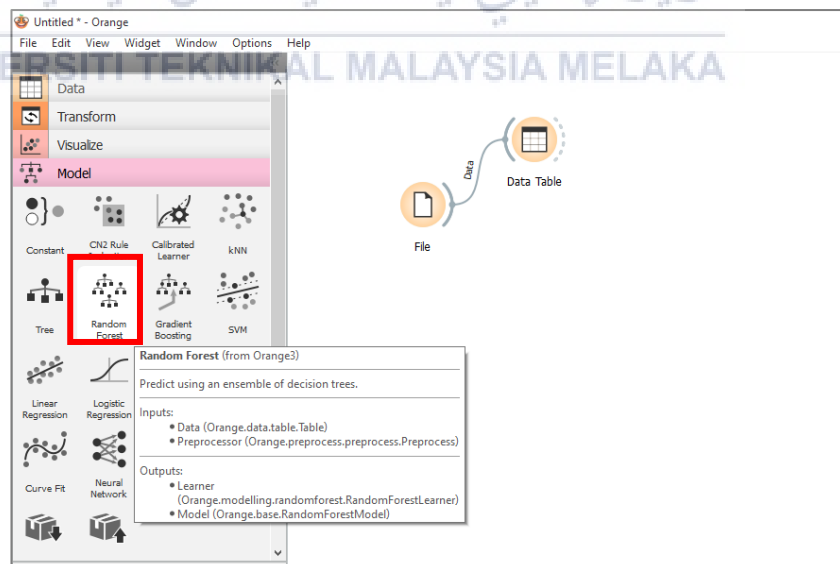


Figure 8 Selection of Random Forest classification algorithms in Orange

- 2) Connect the nodes to start training the model using the Random Forest classification algorithms as shown in Figure 9 below.

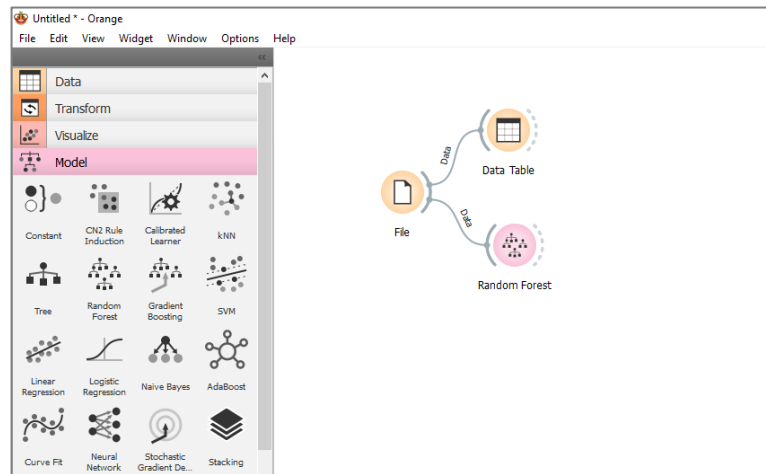


Figure 9 Connecting each node for Random Forest in Orange

S

- 3) To implement the Support Vector Machines (SVM), select the “SVM” widget which represents the Support Vector Machines (SVM) classification algorithm as shown in Figure 10 below.

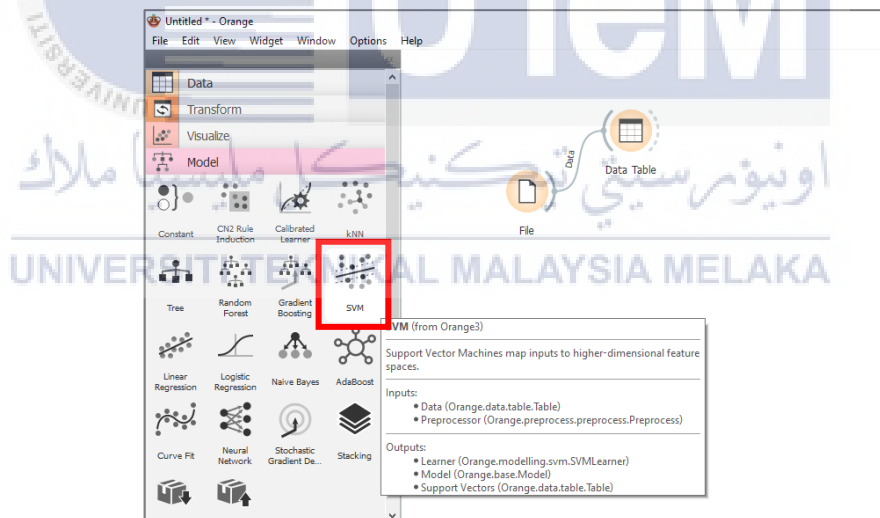


Figure 10 Selection of Support Vector Machines (SVM) classification algorithms in Orange

- 4) Connect the nodes to start training the model using the Support Vector Machines (SVM) classification algorithms as shown in Figure 11 below.

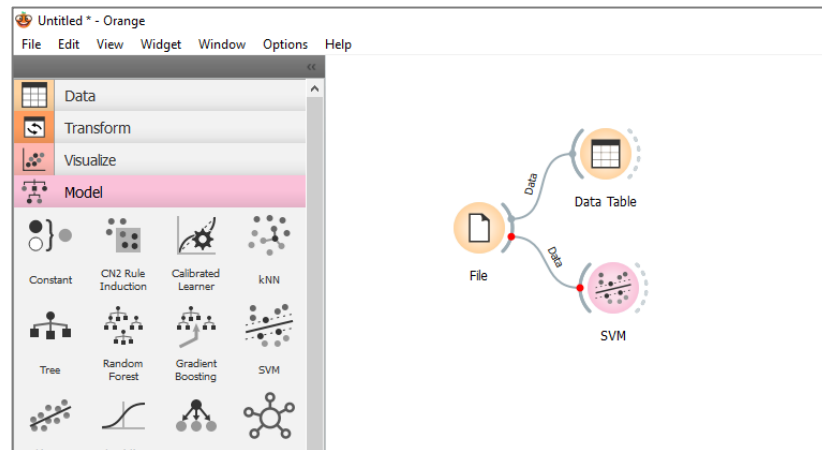


Figure 11 Connecting each node for Support Vector Machines (SVM) classification algorithms in Orange

- 5) To implement the Naïve Bayes select the “Naïve Bayes” widget which represents the Naïve Bayes classification algorithm as shown in Figure 12 below.

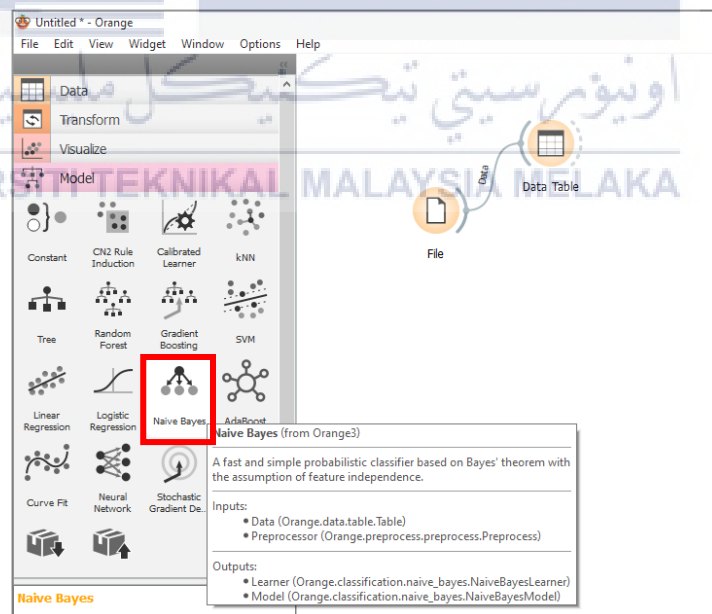


Figure 12 Selection of Naïve Bayes classification algorithms in Orange

- 6) Connect the nodes to start training the model using the Naïve Bayes classification algorithms as shown in Figure 13 below.

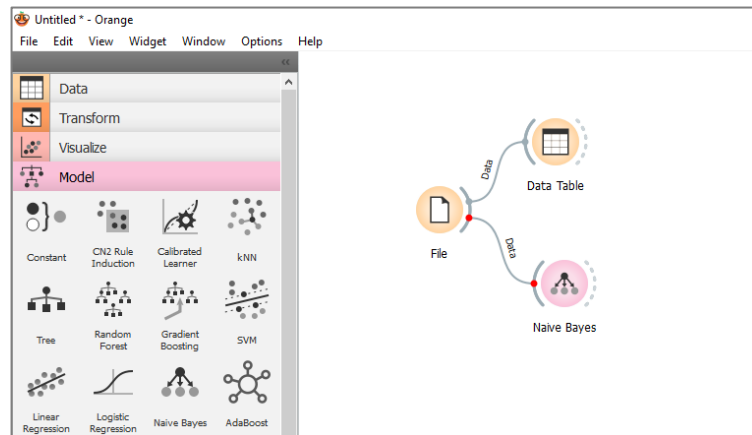
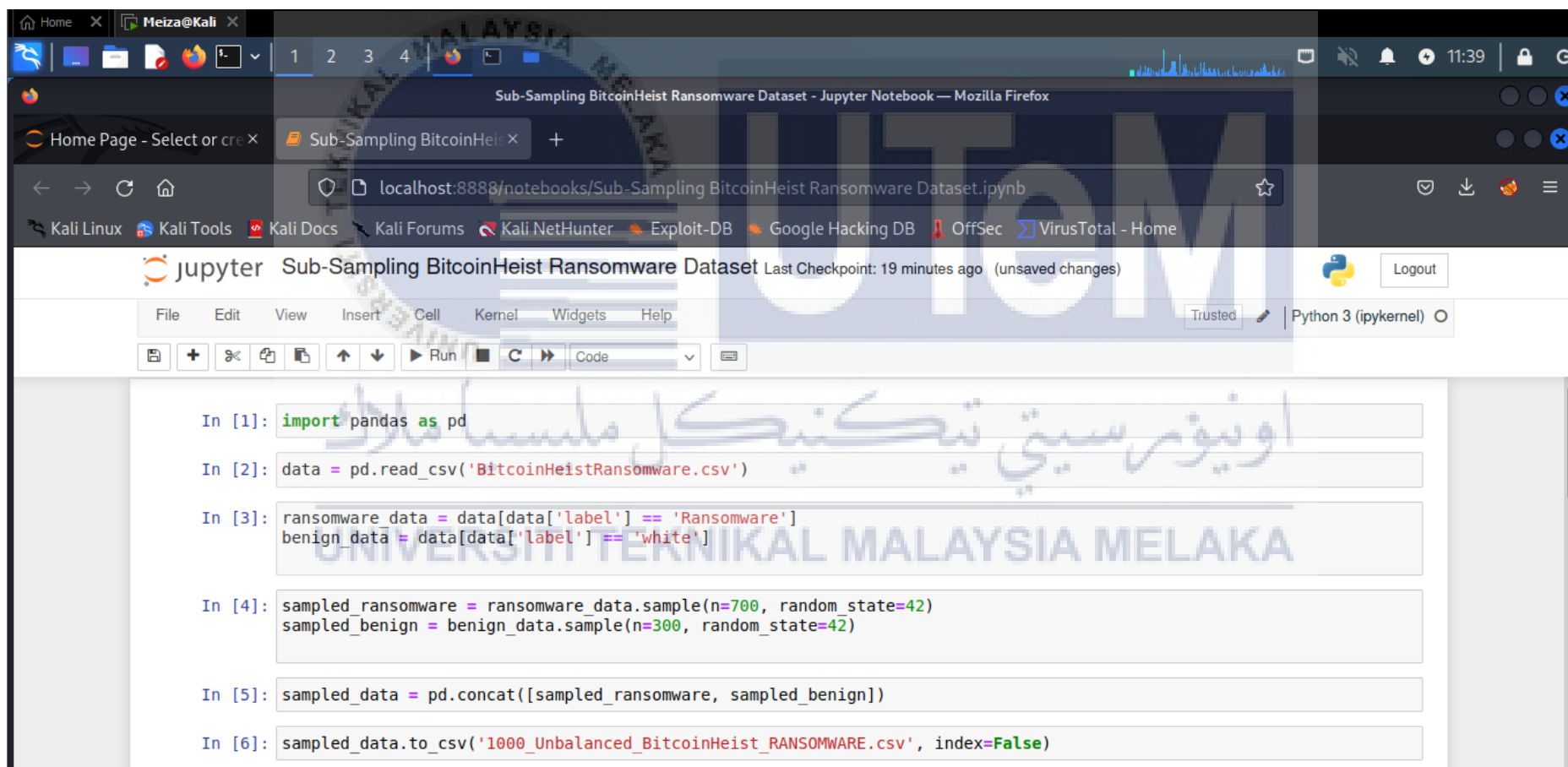


Figure 13 Connecting each node for Naïve Bayes classification algorithms in Orange



## APPENDIX C

### Sample of code for Sub-sampling using Python language in Jupyter Notebook



```
In [1]: import pandas as pd

In [2]: data = pd.read_csv('BitcoinHeistRansomware.csv')

In [3]: ransomware_data = data[data['label'] == 'Ransomware']
benign_data = data[data['label'] == 'white']

In [4]: sampled_ransomware = ransomware_data.sample(n=700, random_state=42)
sampled_benign = benign_data.sample(n=300, random_state=42)

In [5]: sampled_data = pd.concat([sampled_ransomware, sampled_benign])

In [6]: sampled_data.to_csv('1000_Unbalanced_BitcoinHeist_RANSOMWARE.csv', index=False)
```



Sub-Sampling BitcoinHeist Ransomware Dataset - Jupyter Notebook — Mozilla Firefox

localhost:8888/notebooks/Sub-Sampling BitcoinHeist Ransomware Dataset.ipynb

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec VirusTotal - Home

jupyter Sub-Sampling BitcoinHeist Ransomware Dataset Last Checkpoint: 22 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [7]: data.shape
Out[7]: (1048575, 10)

In [8]: data = pd.read_csv('1000_Unbalanced_BitcoinHeist_RANSOMWARE.csv')

In [9]: data.shape
Out[9]: (1000, 10)

In [10]: sampled_data_class_counts = sampled_data['label'].value_counts()
print("Sampled dataset class counts:")
print(sampled_data_class_counts)

Sampled dataset class counts:
Ransomware    700
white          300
```

## APPENDIX D

### 1) BitcoinHeist Ransomware Detection Result (Dataset I)

*Table 1 Summary of evaluation metrics results in WEKA Dataset I (10:90)*

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.860	0.139	0.866	0.860	0.859	86.00%
	Random Forest	0.892	0.107	0.911	0.892	0.891	<b>89.22%</b>
	Naïve Bayes	0.829	0.172	0.838	0.829	0.828	82.89%
	LibSVM	0.561	0.435	0.689	0.561	0.474	56.11%
	SMO	0.901	0.098	0.910	0.910	0.910	90.11%
<b>5000</b>	Decision Tree	0.941	0.059	0.941	0.941	0.941	94.11%
	Random Forest	0.928	0.073	0.937	0.928	0.927	92.78%
	Naïve Bayes	0.885	0.115	0.887	0.885	0.885	88.51%
	LibSVM	0.612	0.391	0.700	0.612	0.562	61.16%
	SMO	0.964	0.036	0.964	0.964	0.964	<b>96.38%</b>
<b>10 000</b>	Decision Tree	0.950	0.050	0.954	0.950	0.950	<b>95.04%</b>
	Random Forest	0.941	0.060	0.947	0.941	0.941	94.06
	Naïve Bayes	0.668	0.331	0.775	0.668	0.633	66.79%
	LibSVM	0.633	0.368	0.687	0.633	0.605	63.34%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 2 Summary of evaluation metrics results in WEKA Dataset I (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.939	0.062	0.942	0.939	0.938	93.86%
	Random Forest	0.947	0.054	0.951	0.947	0.947	<b>94.75%</b>
	Naïve Bayes	0.869	0.131	0.869	0.869	0.869	86.86%
	LibSVM	0.611	0.397	0.707	0.611	0.557	61.14%
	SMO	0.937	0.062	0.938	0.937	0.937	93.71%
<b>5000</b>	Decision Tree	0.949	0.051	0.951	0.949	0.949	94.94%
	Random Forest	0.973	0.027	0.974	0.973	0.973	<b>97.26%</b>
	Naïve Bayes	0.880	0.120	0.882	0.880	0.880	88.03%
	LibSVM	0.635	0.365	0.703	0.635	0.601	68.49%
	SMO	0.962	0.038	0.963	0.962	0.962	96.22%
<b>10 000</b>	Decision Tree	0.950	0.050	0.953	0.950	0.949	94.96%
	Random Forest	0.965	0.035	0.967	0.965	0.965	<b>96.46%</b>
	Naïve Bayes	0.535	0.467	0.702	0.535	0.413	53.54%
	LibSVM	0.644	0.354	0.704	0.644	0.616	64.40%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 3 Summary of evaluation metrics results in WEKA Dataset I (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.938	0.065	0.940	0.938	0.938	93.80%
	Random Forest	0.926	0.078	0.935	0.926	0.925	92.60%
	Naïve Bayes	0.828	0.165	0.862	0.828	0.825	82.80%
	LibSVM	0.642	0.373	0.730	0.642	0.599	64.20%
	SMO	0.940	0.059	0.941	0.940	0.940	<b>94.00%</b>
<b>5000</b>	Decision Tree	0.950	0.051	0.952	0.950	0.950	95.04%
	Random Forest	0.959	0.043	0.962	0.959	0.959	95.88%
	Naïve Bayes	0.908	0.092	0.908	0.908	0.908	90.76%
	LibSVM	0.637	0.373	0.691	0.637	0.606	63.72%
	SMO	0.966	0.034	0.966	0.966	0.966	<b>96.56%</b>
<b>10 000</b>	Decision Tree	0.952	0.047	0.954	0.952	0.952	95.2%
	Random Forest	0.965	0.035	0.967	0.965	0.965	96.46%
	Naïve Bayes	0.548	0.460	0.713	0.548	0.436	54.84%
	LibSVM	0.650	0.346	0.704	0.650	0.626	64.98%
	SMO	0.982	0.018	0.982	0.982	0.982	<b>98.16%</b>

Table 4 Summary of evaluation metrics results in WEKA Dataset I (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.940	0.059	0.940	0.940	0.940	94.00%
	Random Forest	0.957	0.046	0.957	0.957	0.957	<b>95.67%</b>
	Naïve Bayes	0.833	0.144	0.877	0.833	0.831	83.33%
	LibSVM	0.663	0.381	0.732	0.663	0.626	66.33%
	SMO	0.947	0.048	0.950	0.947	0.947	94.67%
<b>5000</b>	Decision Tree	0.953	0.047	0.955	0.953	0.953	95.33%
	Random Forest	0.963	0.037	0.965	0.963	0.963	96.27%
	Naïve Bayes	0.846	0.154	0.876	0.846	0.843	84.60%
	LibSVM	0.632	0.368	0.680	0.632	0.606	63.20%
	SMO	0.967	0.033	0.968	0.967	0.967	<b>96.73%</b>
<b>10 000</b>	Decision Tree	0.953	0.048	0.954	0.953	0.953	95.27%
	Random Forest	0.987	0.014	0.987	0.987	0.987	<b>98.67%</b>
	Naïve Bayes	0.553	0.440	0.716	0.553	0.453	55.33%
	LibSVM	0.660	0.344	0.709	0.660	0.637	65.97%
	SMO	0.985	0.015	0.985	0.985	0.985	98.47%

Table 5 Summary of evaluation metrics results in WEKA Dataset I (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.950	0.062	0.951	0.950	0.950	95.00%
	Random Forest	0.940	0.083	0.946	0.940	0.939	94.00%
	Naïve Bayes	0.680	0.232	0.818	0.680	0.663	68.00%
	LibSVM	0.670	0.423	0.689	0.670	0.635	67.00%
	SMO	0.960	0.036	0.961	0.960	0.960	<b>96.00%</b>
<b>5000</b>	Decision Tree	0.952	0.047	0.952	0.952	0.952	95.20%
	Random Forest	0.968	0.029	0.970	0.968	0.968	96.80%
	Naïve Bayes	0.852	0.161	0.877	0.852	0.848	85.20%
	LibSVM	0.626	0.350	0.687	0.626	0.602	62.6%
	SMO	0.974	0.028	0.975	0.974	0.974	<b>97.4%</b>
<b>10 000</b>	Decision Tree	0.953	0.047	0.954	0.953	0.953	95.30%
	Random Forest	0.984	0.016	0.984	0.984	0.984	98.40%
	Naïve Bayes	0.564	0.431	0.720	0.564	0.241	56.40%
	LibSVM	0.666	0.337	0.723	0.666	0.642	66.60 %
	SMO	0.991	0.009	0.991	0.991	0.991	<b>99.10%</b>

Table 6 Summary of evaluation metrics results in ORANGE Dataset I (10:90)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.808	0.192	0.868	0.858	0.857	85.80%
	Random Forest	0.892	0.108	0.888	0.888	0.888	<b>88.80%</b>
	Naïve Bayes	0.843	0.157	0.860	0.859	0.859	85.90%
	SVM	0.821	0.179	0.840	0.893	0.893	83.90%
<b>5000</b>	Decision Tree	0.931	0.070	0.930	0.930	0.930	93.00%
	Random Forest	0.922	0.045	0.939	0.939	0.939	<b>93.90%</b>
	Naïve Bayes	0.894	0.166	0.865	0.864	0.864	86.40%
	SVM	0.928	0.056	0.936	0.936	0.936	93.60%
<b>10 000</b>	Decision Tree	0.930	0.053	0.938	0.938	0.938	93.80%
	Random Forest	0.922	0.040	0.942	0.941	0.941	<b>94.10%</b>
	Naïve Bayes	0.904	0.171	0.869	0.867	0.866	86.70%
	SVM	0.830	0.023	0.912	0.904	0.903	90.40%

Table 7 Summary of evaluation metrics results in ORANGE Dataset I (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.880	0.064	0.909	0.908	0.908	90.80%
	Random Forest	0.869	0.051	0.911	0.909	0.908	<b>90.90%</b>
	Naïve Bayes	0.855	0.122	0.876	0.866	0.866	86.60%
	SVM	0.872	0.067	0.904	0.902	0.902	90.20%
<b>5000</b>	Decision Tree	0.943	0.057	0.939	0.939	0.939	93.90%
	Random Forest	0.973	0.027	0.946	0.944	0.944	<b>94.40%</b>
	Naïve Bayes	0.849	0.151	0.870	0.869	0.869	86.90%
	SVM	0.981	0.019	0.891	0.871	0.871	87.10%
<b>10 000</b>	Decision Tree	0.951	0.049	0.944	0.944	0.944	94.40%
	Random Forest	0.971	0.029	0.952	0.951	0.951	<b>95.10%</b>
	Naïve Bayes	0.836	0.164	0.870	0.867	0.867	86.70%
	SVM	0.804	0.145	0.865	0.800	0.798	80.40%



Table 8 Summary of evaluation metrics results in ORANGE Dataset I (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.916	0.081	0.917	0.917	0.917	91.70%
	Random Forest	0.892	0.054	0.920	0.918	0.918	91.80%
	Naïve Bayes	0.896	0.162	0.868	0.867	0.867	86.70%
	SVM	0.876	0.046	0.918	0.915	0.915	91.50%
<b>5000</b>	Decision Tree	0.949	0.051	0.941	0.941	0.941	94.10%
	Random Forest	0.968	0.032	0.951	0.950	0.950	<b>95.00%</b>
	Naïve Bayes	0.838	0.162	0.866	0.864	0.864	86.40%
	SVM	0.915	0.085	0.839	0.822	0.820	82.20%
<b>10 000</b>	Decision Tree	0.950	0.050	0.943	0.943	0.943	94.30%
	Random Forest	0.971	0.029	0.953	0.953	0.953	<b>95.30%</b>
	Naïve Bayes	0.836	0.164	0.871	0.867	0.867	86.70%
	SVM	0.892	0.108	0.798	0.768	0.762	76.80%

Table 9 Summary of evaluation metrics results in ORANGE Dataset I (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.908	0.088	0.910	0.910	0.910	91.00%
	Random Forest	0.888	0.038	0.927	0.925	0.925	<b>92.50%</b>
	Naïve Bayes	0.892	0.142	0.875	0.875	0.875	87.50%
	SVM	0.802	0.026	0.900	0.887	0.887	88.70%
<b>5000</b>	Decision Tree	0.940	0.045	0.948	0.948	0.948	94.80%
	Random Forest	0.933	0.030	0.952	0.951	0.951	<b>95.10%</b>
	Naïve Bayes	0.910	0.185	0.866	0.862	0.862	86.20%
	SVM	0.693	0.089	0.817	0.802	0.800	80.20%
<b>10 000</b>	Decision Tree	0.936	0.046	0.945	0.945	0.945	94.50%
	Random Forest	0.936	0.028	0.955	0.955	0.955	<b>95.50%</b>
	Naïve Bayes	0.912	0.183	0.868	0.864	0.864	86.40%
	SVM	0.619	0.073	0.802	0.773	0.767	77.30%

Table 10 Summary of evaluation metrics results in ORANGE Dataset I (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.918	0.074	0.922	0.922	0.922	92.20%
	Random Forest	0.906	0.041	0.933	0.932	0.932	<b>93.20%</b>
	Naïve Bayes	0.908	0.170	0.872	0.870	0.870	87.00%
	SVM	0.795	0.043	0.885	0.874	0.873	88.40%
<b>5000</b>	Decision Tree	0.937	0.054	0.941	0.941	0.941	94.10%
	Random Forest	0.930	0.025	0.954	0.953	0.953	<b>95.30%</b>
	Naïve Bayes	0.915	0.189	0.867	0.862	0.862	86.20%
	SVM	0.702	0.105	0.810	0.800	0.798	80.0%
<b>10 000</b>	Decision Tree	0.931	0.046	0.943	0.942	0.942	94.20%
	Random Forest	0.933	0.025	0.955	0.954	0.954	<b>95.40%</b>
	Naïve Bayes	0.907	0.176	0.868	0.865	0.865	86.50%
	SVM	0.635	0.095	0.791	0.770	0.766	77.00%

## 2) Android Ransomware Detection Result (Dataset II)

Table 11 Summary of evaluation metrics results in WEKA Dataset II (10:90)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.697	0.697	0.697	0.697	0.697	<b>69.67%</b>
	Random Forest	0.654	0.628	0.594	0.654	0.607	65.44%
	Naïve Bayes	0.532	0.390	0.642	0.532	0.548	53.22%
	LibSVM	0.699	0.689	0.714	0.699	0.579	69.89%
	SMO	0.683	0.649	0.614	0.683	0.608	68.33%
<b>5000</b>	Decision Tree	0.699	0.699	0.699	0.699	0.699	69.93%
	Random Forest	0.716	0.544	0.686	0.716	0.679	<b>71.58%</b>
	Naïve Bayes	0.532	0.368	0.656	0.532	0.547	53.20%
	LibSVM	0.698	0.697	0.589	0.698	0.577	69.80%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A
<b>10 000</b>	Decision Tree	0.699	0.699	0.699	0.699	0.699	<b>69.86%</b>
	Random Forest	0.683	0.544	0.651	0.683	0.658	68.27%
	Naïve Bayes	0.581	0.431	0.640	0.581	0.598	58.12%
	LibSVM	0.695	0.698	0.607	0.695	0.584	69.51%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 12 Summary of evaluation metrics results in WEKA Dataset II (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.697	0.697	0.697	0.697	0.697	69.71%
	Random Forest	0.723	0.590	0.709	0.723	0.660	<b>72.29%</b>
	Naïve Bayes	0.657	0.581	0.618	0.657	0.628	65.71%
	LibSVM	0.694	0.685	0.614	0.694	0.584	69.42%
	SMO	0.697	0.614	0.651	0.697	0.634	69.71%
<b>5000</b>	Decision Tree	0.704	0.704	0.704	0.704	0.704	70.43%
	Random Forest	0.738	0.478	0.718	0.738	0.717	<b>73.83%</b>
	Naïve Bayes	0.518	0.292	0.710	0.518	0.512	51.77%
	LibSVM	0.701	0.700	0.592	0.701	0.586	70.11%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A
<b>10 000</b>	Decision Tree	0.699	0.699	0.699	0.699	0.699	69.94%
	Random Forest	0.746	0.419	0.733	0.746	0.735	<b>74.61%</b>
	Naïve Bayes	0.520	0.306	0.696	0.520	0.524	51.99%
	LibSVM	0.695	0.687	0.607	0.695	0.587	69.50%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 13 Summary of evaluation metrics results in WEKA Dataset II (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.694	0.694	0.694	0.694	0.694	69.40%
	Random Forest	0.722	0.550	0.700	0.722	0.676	<b>72.20%</b>
	Naïve Bayes	0.560	0.344	0.676	0.560	0.574	56.00%
	LibSVM	0.696	0.671	0.652	0.696	0.590	69.60%
	SMO	0.710	0.614	0.688	0.710	0.637	71.00%
<b>5000</b>	Decision Tree	0.695	0.695	0.695	0.695	0.695	69.48%
	Random Forest	0.747	0.439	0.732	0.747	0.730	<b>74.72%</b>
	Naïve Bayes	0.541	0.303	0.701	0.541	0.547	54.08%
	LibSVM	0.692	0.689	0.592	0.692	0.575	69.16%
	SMO	0.740	0.489	0.723	0.740	0.711	74.00%
<b>10 000</b>	Decision Tree	0.698	0.698	0.698	0.698	0.698	69.80%
	Random Forest	0.762	0.398	0.751	0.762	0.751	76.24%
	Naïve Bayes	0.571	0.294	0.713	0.571	0.583	57.14%
	LibSVM	0.697	0.688	0.623	0.697	0.583	69.66%
	SMO	0.765	0.469	0.759	0.765	0.736	<b>76.52%</b>

Table 14 Summary of evaluation metrics results in WEKA Dataset II (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.663	0.663	0.663	0.663	0.663	66.33%
	Random Forest	0.727	0.509	0.747	0.727	0.675	<b>72.67% (Best)</b>
	Naïve Bayes	0.560	0.350	0.658	0.560	0.567	56.00%
	LibSVM	0.660	0.646	0.592	0.660	0.550	66.00%
	SMO	0.707	0.549	0.721	0.707	0.642	70.67%
<b>5000</b>	Decision Tree	0.692	0.692	0.692	0.692	0.692	69.20%
	Random Forest	0.746	0.409	0.733	0.746	0.735	74.60%
	Naïve Bayes	0.525	0.301	0.699	0.525	0.527	52.53%
	LibSVM	0.689	0.680	0.606	0.689	0.577	68.87%
	SMO	0.751	0.474	0.740	0.751	0.722	<b>75.07% (Best)</b>
<b>10 000</b>	Decision Tree	0.702	0.702	0.702	0.702	0.702	70.20%
	Random Forest	0.761	0.386	0.750	0.761	0.753	76.07%
	Naïve Bayes	0.690	0.369	0.711	0.690	0.698	69.00%
	LibSVM	0.701	0.683	0.637	0.701	0.596	70.10%
	SMO	0.771	0.462	0.765	0.771	0.744	<b>77.13% (Best)</b>

Table 15 Summary of evaluation metrics results in WEKA Dataset II (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.700	0.700	0.700	0.700	0.700	70.00%
	Random Forest	0.770	0.480	0.770	0.770	0.737	<b>77.00% (Best)</b>
	Naïve Bayes	0.540	0.388	0.648	0.540	0.557	54.00%
	LibSVM	0.690	0.685	0.591	0.690	0.588	69.00%
	SMO	0.770	0.480	0.770	0.770	0.737	77.00%
<b>5000</b>	Decision Tree	0.690	0.690	0.690	0.690	0.690	69.00%
	Random Forest	0.794	0.355	0.787	0.794	0.783	<b>79.40% (Best)</b>
	Naïve Bayes	0.566	0.302	0.704	0.566	0.575	56.60%
	LibSVM	0.686	0.678	0.601	0.686	0.575	68.60%
	SMO	0.782	0.418	0.781	0.782	0.760	78.20%
<b>10 000</b>	Decision Tree	0.690	0.690	0.690	0.690	0.690	69.60%
	Random Forest	0.782	0.353	0.774	0.782	0.775	78.20%
	Naïve Bayes	0.719	0.362	0.725	0.719	0.722	71.90%
	LibSVM	0.699	0.674	0.664	0.669	0.592	69.90%
	SMO	0.786	0.429	0.786	0.786	0.762	<b>78.60% (Best)</b>



Table 16 Summary of results evaluation metrics results in ORANGE Dataset II (10:90)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.551	0.449	0.549	0.548	0.548	54.80%
	Random Forest	0.553	0.447	0.549	0.549	0.548	54.90%
	Naïve Bayes	0.551	0.449	0.555	0.556	0.555	55.60%
	SVM	0.554	0.446	0.543	0.541	0.534	54.10%
<b>5000</b>	Decision Tree	0.525	0.475	0.530	0.529	0.528	52.90%
	Random Forest	0.531	0.469	0.535	0.534	0.534	<b>53.40%</b>
	Naïve Bayes	0.499	0.501	0.500	0.500	0.500	50.00%
	SVM	0.518	0.482	0.513	0.510	0.468	51.00%
<b>10 000</b>	Decision Tree	0.540	0.460	0.542	0.542	0.542	54.20%
	Random Forest	0.554	0.446	0.554	0.554	0.554	<b>55.40%</b>
	Naïve Bayes	0.505	0.495	0.505	0.506	0.505	50.60%
	SVM	0.513	0.487	0.508	0.506	0.463	50.60%

Table 17 Summary of evaluation metrics results in ORANGE Dataset II (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.572	0.428	0.575	0.575	0.575	57.50%
	Random Forest	0.590	0.410	0.591	0.591	0.591	<b>59.10%</b>
	Naïve Bayes	0.577	0.423	0.574	0.574	0.574	57.40%
	SVM	0.602	0.398	0.560	0.532	0.471	53.20%
<b>5000</b>	Decision Tree	0.551	0.449	0.555	0.554	0.554	55.40%
	Random Forest	0.569	0.431	0.570	0.570	0.570	<b>57.00%</b>
	Naïve Bayes	0.499	0.501	0.499	0.499	0.499	49.90%
	SVM	0.502	0.498	0.501	0.501	0.498	50.10%
<b>10 000</b>	Decision Tree	0.570	0.430	0.577	0.576	0.575	57.60%
	Random Forest	0.597	0.403	0.601	0.601	0.601	<b>60.10%</b>
	Naïve Bayes	0.505	0.495	0.506	0.506	0.505	50.60%
	SVM	0.500	0.500	0.501	0.501	0.497	50.10%

Table 18 Summary of evaluation metrics results in ORANGE Dataset II (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.566	0.434	0.583	0.579	0.575	57.90%
	Random Forest	0.592	0.408	0.600	0.599	0.598	59.90%
	Naïve Bayes	0.581	0.419	0.575	0.575	0.574	57.50%
	SVM	0.587	0.413	0.548	0.517	0.422	51.70%
<b>5000</b>	Decision Tree	0.559	0.441	0.563	0.563	0.562	56.30%
	Random Forest	0.578	0.422	0.579	0.579	0.579	<b>57.90%</b>
	Naïve Bayes	0.501	0.499	0.501	0.501	0.501	50.10%
	SVM	0.501	0.499	0.501	0.501	0.501	50.10%
<b>10 000</b>	Decision Tree	0.570	0.430	0.577	0.576	0.575	57.60%
	Random Forest	0.597	0.403	0.601	0.601	0.601	60.10%
	Naïve Bayes	0.505	0.495	0.506	0.506	0.505	50.60%
	SVM	0.500	0.500	0.501	0.501	0.497	50.10%

Table 19 Summary of evaluation metrics results in ORANGE Dataset II (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.615	0.462	0.574	0.573	0.573	57.30%
	Random Forest	0.630	0.426	0.615	0.615	0.615	<b>61.50%</b>
	Naïve Bayes	0.513	0.486	0.584	0.584	0.584	58.40%
	SVM	0.375	0.346	0.514	0.511	0.450	51.10%
<b>5000</b>	Decision Tree	0.606	0.457	0.575	0.574	0.574	57.40%
	Random Forest	0.611	0.416	0.598	0.598	0.598	<b>59.80%</b>
	Naïve Bayes	0.522	0.518	0.502	0.502	0.502	50.20%
	SVM	0.267	0.259	0.505	0.504	0.475	50.40%
<b>10 000</b>	Decision Tree	0.618	0.452	0.583	0.583	0.582	58.30%
	Random Forest	0.627	0.400	0.614	0.614	0.614	<b>61.40%</b>
	Naïve Bayes	0.555	0.537	0.509	0.509	0.508	50.90%
	SVM	0.368	0.365	0.502	0.502	0.492	50.20%

Table 20 Summary of evaluation metrics results in ORANGE Dataset II (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.572	0.480	0.546	0.545	0.545	54.50%
	Random Forest	0.641	0.408	0.617	0.616	0.616	<b>61.60%</b>
	Naïve Bayes	0.158	0.094	0.580	0.580	0.579	58.00%
	SVM	0.527	0.369	0.572	0.541	0.465	54.10%
<b>5000</b>	Decision Tree	0.596	0.435	0.581	0.581	0.581	58.10%
	Random Forest	0.615	0.414	0.601	0.601	0.601	<b>60.10%</b>
	Naïve Bayes	0.559	0.531	0.514	0.515	0.514	51.50%
	SVM	0.218	0.189	0.522	0.510	0.463	51.00%
<b>10 000</b>	Decision Tree	0.585	0.415	0.591	0.591	0.590	59.10%
	Random Forest	0.611	0.389	0.613	0.613	0.613	<b>61.30%</b>
	Naïve Bayes	0.507	0.493	0.506	0.506	0.506	50.60%
	SVM	0.503	0.497	0.502	0.501	0.500	50.10%

### 3) File System Behavior Ransomware Detection Result (Dataset III)

Table 21 Summary of evaluation metrics results in WEKA Dataset III (10:90)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.950	0.050	0.951	0.950	0.950	<b>95.00%</b>
	Random Forest	0.944	0.055	0.944	0.944	0.944	94.44%
	Naïve Bayes	0.793	0.207	0.795	0.793	0.793	79.33%
	LibSVM	0.617	0.380	0.783	0.617	0.552	61.67%
	SMO	0.873	0.127	0.873	0.873	0.873	87.33
<b>5000</b>	Decision Tree	0.970	0.030	0.970	0.970	0.970	<b>97.00%</b>
	Random Forest	0.955	0.045	0.956	0.955	0.955	95.53%
	Naïve Bayes	0.792	0.207	0.807	0.792	0.790	79.22%
	LibSVM	0.692	0.309	0.809	0.692	0.660	69.24%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A
<b>10 000</b>	Decision Tree	0.967	0.033	0.967	0.967	0.967	<b>96.67%</b>
	Random Forest	0.951	0.049	0.951	0.951	0.951	95.09%
	Naïve Bayes	0.825	0.174	0.847	0.825	0.822	82.48%
	LibSVM	0.737	0.265	0.827	0.737	0.717	73.65%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 22 Summary of evaluation metrics results in WEKA Dataset III (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.953	0.047	0.954	0.953	0.953	<b>95.29%</b>
	Random Forest	0.944	0.056	0.944	0.944	0.944	94.43%
	Naïve Bayes	0.716	0.279	0.791	0.716	0.697	71.57%
	LibSVM	0.657	0.351	0.796	0.657	0.610	65.71%
	SMO	0.830	0.167	0.853	0.830	0.827	83.00%
<b>5000</b>	Decision Tree	0.980	0.020	0.980	0.980	0.980	<b>98.02%</b>
	Random Forest	0.953	0.047	0.954	0.953	0.953	95.31%
	Naïve Bayes	0.843	0.157	0.847	0.843	0.843	84.31%
	LibSVM	0.746	0.254	0.831	0.746	0.729	74.60%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A
<b>10 000</b>	Decision Tree	0.985	0.015	0.985	0.985	0.985	<b>98.53%</b>
	Random Forest	0.954	0.046	0.955	0.954	0.954	95.40%
	Naïve Bayes	0.790	0.211	0.812	0.790	0.786	79.01%
	LibSVM	0.800	0.199	0.857	0.800	0.792	80.01%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 23 Summary of evaluation metrics results in WEKA Dataset III (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.954	0.045	0.955	0.954	0.954	95.40%
	Random Forest	0.960	0.040	0.960	0.960	0.960	<b>96.00% (Best)</b>
	Naïve Bayes	0.736	0.253	0.800	0.736	0.723	73.60%
	LibSVM	0.680	0.336	0.803	0.680	0.640	68.00%
	SMO	0.828	0.166	0.853	0.828	0.826	82.80%
<b>5000</b>	Decision Tree	0.989	0.008	0.991	0.991	0.991	<b>99.10% (Best)</b>
	Random Forest	0.952	0.049	0.952	0.952	0.952	95.16%
	Naïve Bayes	0.845	0.152	0.854	0.845	0.845	84.52%
	LibSVM	0.771	0.237	0.841	0.771	0.758	77.12%
	SMO	0.834	0.162	0.853	0.834	0.832	83.36%
<b>10 000</b>	Decision Tree	0.984	0.016	0.984	0.984	0.984	98.36%
	Random Forest	0.954	0.046	0.954	0.954	0.954	95.38%
	Naïve Bayes	0.801	0.202	0.831	0.801	0.796	80.10%
	LibSVM	0.823	0.174	0.869	0.823	0.817	82.28%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A



Table 24 Summary of evaluation metrics results in WEKA Dataset III (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.980	0.017	0.981	0.980	0.980	<b>98.00%</b>
	Random Forest	0.963	0.038	0.963	0.963	0.963	96.33%
	Naïve Bayes	0.697	0.264	0.803	0.697	0.677	69.97%
	LibSVM	0.713	0.332	0.813	0.713	0.679	71.33%
	SMO	0.810	0.169	0.847	0.810	0.808	81.00%
<b>5000</b>	Decision Tree	0.983	0.017	0.983	0.983	0.983	<b>98.33%</b>
	Random Forest	0.955	0.045	0.956	0.955	0.955	95.53%
	Naïve Bayes	0.839	0.161	0.845	0.839	0.838	83.87%
	LibSVM	0.783	0.217	0.848	0.783	0.773	78.33%
	SMO	0.838	0.162	0.856	0.838	0.836	83.80%
<b>10 000</b>	Decision Tree	0.986	0.014	0.986	0.986	0.986	<b>98.60%</b>
	Random Forest	0.868	0.131	0.875	0.868	0.868	86.83%
	Naïve Bayes	0.756	0.247	0.812	0.756	0.744	75.60%
	LibSVM	0.838	0.165	0.877	0.838	0.833	83.80%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 25 Summary of evaluation metrics results in WEKA Dataset III (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.980	0.014	0.981	0.980	0.980	98.00%
	Random Forest	0.990	0.007	0.990	0.990	0.990	<b>99.00%</b>
	Naïve Bayes	0.800	0.145	0.865	0.800	0.798	80.00%
	LibSVM	0.670	0.456	0.790	0.670	0.600	67.00%
	SMO	0.820	0.130	0.874	0.820	0.819	82.00%
<b>5000</b>	Decision Tree	0.994	0.006	0.994	0.994	0.994	<b>99.40%</b>
	Random Forest	0.950	0.048	0.951	0.950	0.950	95.00%
	Naïve Bayes	0.840	0.167	0.844	0.840	0.839	84.00%
	LibSVM	0.798	0.183	0.858	0.798	0.792	79.80%
	SMO	0.840	0.170	0.853	0.840	0.838	84.00%
<b>10 000</b>	Decision Tree	0.982	0.018	0.982	0.982	0.982	<b>98.20%</b>
	Random Forest	0.949	0.051	0.950	0.949	0.868	94.90%
	Naïve Bayes	0.556	0.449	0.701	0.556	0.456	55.60%
	LibSVM	0.857	0.145	0.889	0.857	0.854	85.70%
	SMO	N/A	N/A	N/A	N/A	N/A	N/A

Table 26 Summary of evaluation metrics results in ORANGE Dataset III (10:90)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.949	0.075	0.937	0.937	0.937	93.70%
	Random Forest	0.945	0.038	0.954	0.954	0.954	<b>95.40%</b>
	Naïve Bayes	0.777	0.048	0.876	0.865	0.864	86.50%
	SVM	0.792	0.078	0.863	0.857	0.856	85.70%
<b>5000</b>	Decision Tree	0.957	0.040	0.959	0.959	0.959	95.90%
	Random Forest	0.977	0.034	0.971	0.971	0.971	<b>97.10%</b>
	Naïve Bayes	0.814	0.057	0.885	0.879	0.878	87.90%
	SVM	0.951	0.080	0.936	0.936	0.936	93.60%
<b>10 000</b>	Decision Tree	0.939	0.026	0.957	0.956	0.956	95.60%
	Random Forest	0.983	0.011	0.986	0.986	0.986	<b>98.60%</b>
	Naïve Bayes	0.816	0.049	0.890	0.883	0.883	88.30%
	SVM	0.943	0.035	0.954	0.954	0.954	95.40%

Table 27 Summary of evaluation metrics results in ORANGE Dataset III (30:70)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.953	0.047	0.953	0.953	0.953	95.30%
	Random Forest	0.967	0.033	0.968	0.968	0.968	<b>96.80%</b>
	Naïve Bayes	0.933	0.067	0.878	0.870	0.870	87.00%
	SVM	0.900	0.100	0.914	0.913	0.913	91.30%
<b>5000</b>	Decision Tree	0.955	0.023	0.966	0.966	0.966	96.60%
	Random Forest	0.984	0.014	0.985	0.985	0.985	<b>98.50%</b>
	Naïve Bayes	0.815	0.052	0.888	0.881	0.881	88.10%
	SVM	0.676	0.048	0.840	0.814	0.810	81.40%
<b>10 000</b>	Decision Tree	0.956	0.011	0.973	0.973	0.973	97.30%
	Random Forest	0.989	0.010	0.989	0.989	0.989	<b>98.90%</b>
	Naïve Bayes	0.819	0.058	0.886	0.880	0.880	88.00%
	SVM	0.738	0.062	0.852	0.838	0.836	83.80%

Table 28 Summary evaluation metrics results in ORANGE Dataset III (50:50)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.948	0.030	0.959	0.959	0.959	95.90%
	Random Forest	0.979	0.016	0.981	0.981	0.981	98.10%
	Naïve Bayes	0.794	0.058	0.876	0.868	0.867	86.80%
	SVM	0.936	0.075	0.931	0.931	0.931	93.10%
<b>5000</b>	Decision Tree	0.960	0.019	0.971	0.971	0.971	97.10%
	Random Forest	0.990	0.008	0.991	0.991	0.991	<b>99.10%</b>
	Naïve Bayes	0.813	0.059	0.884	0.877	0.877	87.70%
	SVM	0.675	0.057	0.833	0.809	0.805	80.90%
<b>10 000</b>	Decision Tree	0.962	0.018	0.972	0.972	0.972	97.20%
	Random Forest	0.990	0.005	0.992	0.992	0.992	<b>99.20%</b>
	Naïve Bayes	0.826	0.045	0.898	0.891	0.891	89.10
	SVM	0.744	0.105	0.827	0.820	0.819	82.00

Table 29 Summary of evaluation metrics results in ORANGE Dataset III (70:30)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.961	0.026	0.968	0.968	0.968	96.80%
	Random Forest	0.986	0.009	0.988	0.988	0.988	<b>98.80%</b>
	Naïve Bayes	0.810	0.067	0.877	0.872	0.871	87.20%
	SVM	0.888	0.040	0.926	0.924	0.924	92.40%
<b>5000</b>	Decision Tree	0.959	0.020	0.969	0.969	0.969	96.90%
	Random Forest	0.989	0.008	0.991	0.991	0.991	<b>99.10%</b>
	Naïve Bayes	0.683	0.057	0.882	0.876	0.875	87.60%
	SVM	0.810	0.059	0.836	0.813	0.810	81.30%
<b>10 000</b>	Decision Tree	0.964	0.018	0.973	0.973	0.973	97.30%
	Random Forest	0.991	0.005	0.933	0.933	0.933	<b>99.30%</b>
	Naïve Bayes	0.826	0.044	0.898	0.891	0.891	89.10%
	SVM	0.764	0.115	0.829	0.824	0.824	82.40%

Table 30 Summary of evaluation metrics results in ORANGE Dataset III (90:10)

Sample Size	Algorithms	True Positive Rate	False Positive Rate	Precision	Recall	F-measure	Overall Accuracy
<b>1000</b>	Decision Tree	0.957	0.035	0.961	0.961	0.961	96.10%
	Random Forest	0.986	0.010	0.988	0.988	0.988	<b>98.80%</b>
	Naïve Bayes	0.838	0.074	0.885	0.883	0.883	88.30%
	SVM	0.814	0.021	0.908	0.898	0.898	89.80%
<b>5000</b>	Decision Tree	0.969	0.019	0.974	0.974	0.974	97.40
	Random Forest	0.991	0.006	0.993	0.993	0.993	<b>99.30%</b>
	Naïve Bayes	0.715	0.075	0.834	0.818	0.816	81.80%
	SVM	0.822	0.046	0.894	0.887	0.887	88.70%
<b>10 000</b>	Decision Tree	0.964	0.013	0.976	0.976	0.976	97.60%
	Random Forest	0.991	0.006	0.993	0.933	0.993	<b>99.30%</b>
	Naïve Bayes	0.831	0.042	0.901	0.894	0.894	89.40%
	SVM	0.737	0.177	0.782	0.780	0.779	78.00%