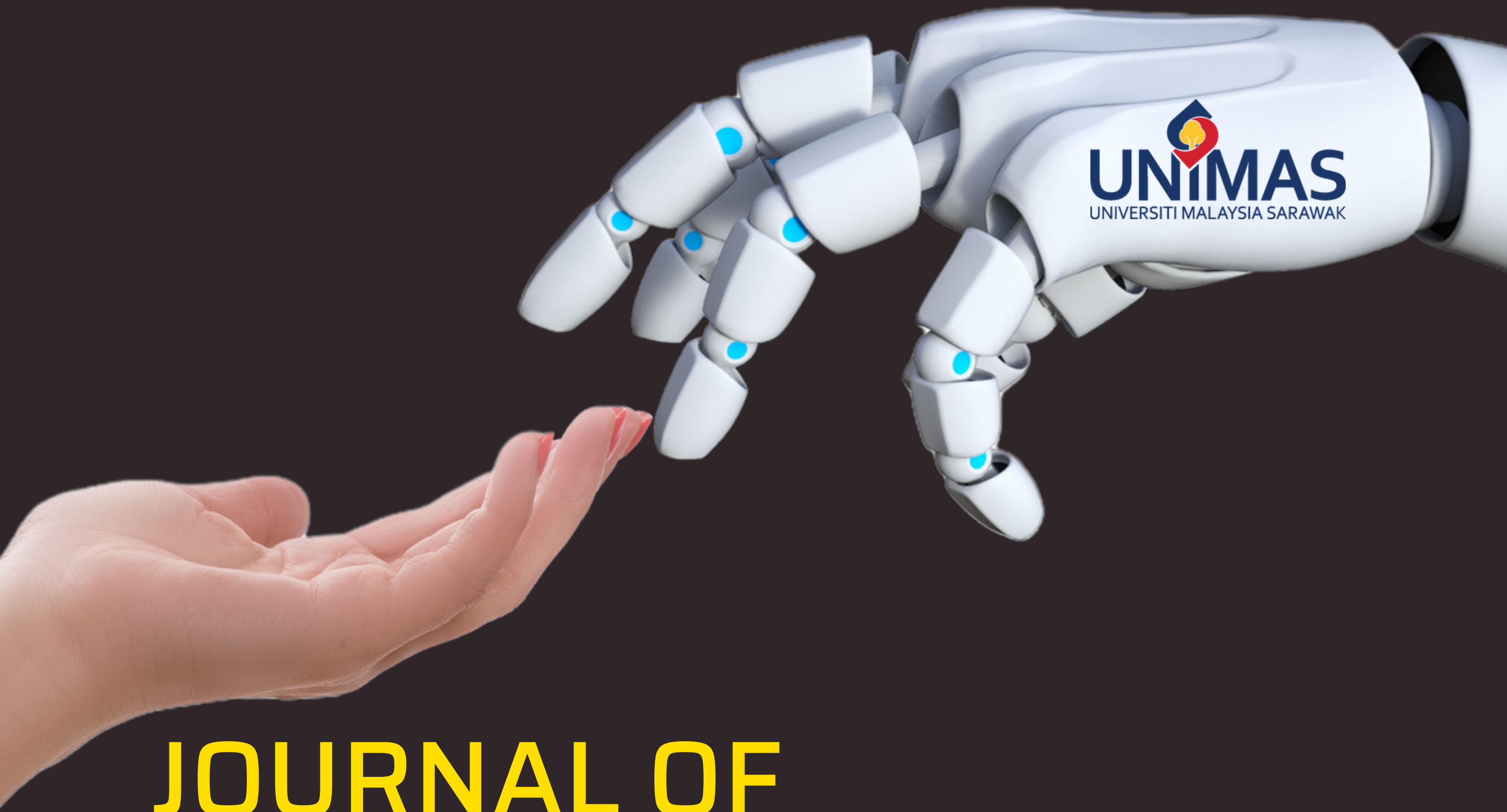

UNIMAS
UNIVERSITI MALAYSIA SARAWAK

JOURNAL OF COMPUTING AND SOCIAL INFORMATICS

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNIVERSITI MALAYSIA SARAWAK

eISSN 2821-3777

9 772821 377005

Journal of Computing and Social Informatics

The Journal of Computing and Social Informatics (JCSI) is an international peer-reviewed publication that focuses on the emerging areas of Computer Science and the overarching impact of technologies on all aspects of our life at societal level. This journal serves as a platform to promote the exchange of ideas with researchers around the world.

Articles can be submitted via *www.jcsi.unimas.my*

Assoc Prof Dr Chiew Kang Leng

Chief Editor
Journal of Computing and Social Informatics
Faculty of Computer Science and Information Technology
Universiti Malaysia Sarawak
94300 Kota Samarahan
Sarawak, Malaysia

Contents

Credit Risk Prediction for Peer-To-Peer Lending Platforms: An Explainable Machine Learning Approach

1*Chong Pei Swee, 2Farid Meziane, and 3Jane Labadin

1, 3Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
2School of Computing and Engineering, University of Derby, UK

email: 1*chongpeiswee@gmail.com, 2F.Meziane@derby.ac.uk, 3ljane@unimas.my

**Corresponding author*

Received: 23 Jun 2022 | Accepted: 30 August 2022 | Early access: 19 September 2022

Abstract - *Small and medium enterprises face the challenge of obtaining start-up fund due to the strict rules and conditions set by banks and financial institutions. The plight yields to the growth in popularity of online peer-to-peer lending platforms which are an easier way to obtain loan as they have fewer rigid rules. However, high flexibility of loan funding in peer-to-peer lending comes with high default probability of loan funded to high-risk start-ups. An efficient model for evaluating credit risk of borrowers in peer-to-peer lending platforms is important to encourage investors to fund loans and justify the rejection of unsuccessful applications to satisfy financial regulators and increase transparency. This paper presents a supervised machine learning model with logistic regression to address this issue and predicts the probability of default of a loan funded to borrowers through peer-to-peer lending platforms. In addition, factors that affect the credit levels of borrowers are identified and discussed. The research shows that the most important features that affect probability of default are debt-to-income ratio, number of mortgage account, and Fair, Isaac and Company Score.*

Keywords: Credit Risk Evaluation, Peer-to-Peer Lending, Logistic Regression; Explainable Machine Learning; Explainable AI.

1 Introduction

Peer-To-Peer (P2P) lending platforms are online services provided by financial institutions as an intermediary to initiate loans for private individuals (Bachmann et al., 2011). Loans for borrowers are funded by multiple investors, bound with agreed-upon terms and conditions, with profits generated from the interest made on the loans as the borrowers are given a certain duration to pay back the loan and interest. The higher the investment risk, the higher is the interest rate. Due to a reduction in loans to small businesses from banks, P2P lending has gained popularity for personal, small business start-ups and SMEs loans as these tend to have high failing rate to pay back their loans and with low credit scores. Indeed, P2P lending allows individuals and businesses to loan money directly from investors or lenders without going through the strict requirements and criteria of traditional banks and financial institutions. Although these platforms provide several instruments to assess and limit credit risks, they do not guarantee the repayment of loans (Meyer, 2007).

The most common credit score for risks assessment is the "Fair, Isaac and Company" (FICO) score. The FICO score is not suitable for P2P lending since these platforms are meant for relatively high-risk start-ups, and for those that failed to secure loans from banks due to their low credit scores. Small and medium-sized enterprises (SMEs) which are categorized as high-risk client by financial institution play an important role in many economies, and to encourage their growth, a reliable and accurate clients' credit risk evaluation is critical to build confidence among investors so that more funds are available on P2P lending platforms. This paper presents a supervised machine learning model that predicts the probability of default by considering more information related to the clients rather than just evaluating their credit score using FICO. The focus will be on solving the credit evaluation problem for P2P lending marketplace and determine important features that contribute to the probability of default.

2 Literature Review

P2P lending has become an alternative to obtain loans from traditional financial institutions. Most of the middle-income population lost their creditworthiness as borrowers to obtain loans from traditional financial institutions after the financial crisis in 2008, causing P2P lending became the choice for getting a loan for many individuals (Namvar, 2013). According to Emekter et al. (2014), the lack of a physical contact between lenders and borrowers in an online P2P lending process has posed the problem of information asymmetry between lenders and borrowers. Hence, having an efficient and accurate credit risk evaluation method to decrease the investment risk without human intervention is critical to sustain the steady development of the P2P lending industry.

Setiawan et al. (2019) developed a P2P lending default loan classification model using data acquired from the Lending Club through the application of Extremely Randomised Tree (ERT) and RF methods and optimised their performance with Binary Particle Swarm Optimisation (BPSO) and SVM during the feature selection. BPSO is the binary version for particle swarm optimisation (PSO), a branch of swarm intelligence, that iteratively optimises the candidate solution by guiding it towards best known position and thus finally reaching to the best solution. The evaluation of the models revealed that the average performance of ERT can outperform RF.

Emekter et al. (2014) carried out a binary Logistic Regression model for classifying default and non-default loans. The forward stepwise iterative maximum likelihood method was implemented to determine variables that have strong influence on the model and was analysed by backward stepwise of iterative maximum likelihood method. Research stated that higher credit grade is associated with lower default risk. The researchers further evaluated the selection bias by taking two different population samples, one contains data of the United States national consumers, and another contains data of Lending Club consumers. Insignificant difference of default probability for two sample indicates the consideration of data beyond the Lending Club platform is unnecessary.

High-dimensionality and imbalance class of dataset from P2P lending platform is always the challenge for making accurate prediction of default probability. In research conducted by Zhou et al. (2019), gradient boosting decision trees (GBDT), extreme gradient boosting (XGBoost), and light gradient boosting machine (LightGBM) were integrated with heterogeneous ensemble learning technology to address the issue. The ensemble model of GBDT, XGBoost and LightGBM outperformed individual classifiers of their own, proving the ability of ensemble learning model to optimize prediction from a high dimension and imbalance dataset.

Dong et al. (2010) applied the logistic regression model with random coefficients (LRR) to develop credit scorecard. A dataset with 1000 samples was divided into 10 subsets with 9 of the subsets used as training sets while the remaining subset as the testing set. The random coefficients for 900 samples are generated using Gibbs sampling within the Bayesian inference starting with estimated coefficient of logistic regression with fixed coefficients (LRF). They performed empirical experiment to evaluate the prediction accuracy of LRF and LRR with Percent Correctly Classified (PCC) method. The LRR has the overall accuracy of 74% which outperform LRF with only 71% of overall accuracy. Dong et al. argue that the logistic regression is an optimal solution for credit scoring model for financial industry in term of result interpretability.

Wang et al. (2015) had implemented lasso-logistic regression ensemble (LLRE) learning algorithm to predict default probability based on a large imbalanced dataset. Researchers clustered the majority data into sub-groups based on variables similarity and applied bagging method to minority data. Weighted average was computed for aggregation of the ensemble model. Wang et. al. created the generated variables from the original variables by partitioning them into specific intervals. The generated variables successfully reduced noise and non-linearity, thus improving the performance of the Lasso-logistic regression model. LLRE outperforms all the compared models (LLR, RF and the Classification and Regression Tree (CART)) in modelling imbalanced large dataset with significantly higher average AUC value.

Coenen et al. (2021) evaluates performance of machine learning methods from different families, namely the generalized linear models, support vector machines and gradient-boosted trees, under the context of spot factoring. They estimated the risk for spot factoring in terms of payment overdue using the machine learning methods mentioned earlier to achieve three tasks namely: binary classification of probability of default, prediction of days of overdue, and risk ranking with pre-defined labels. They found that the regression method shows higher consistency in getting high scores among all the method families in all the evaluation tasks.

The interpretability of the model is the major concern for financial institution since they are asked to provide evidence and reason for rejecting loan applications. Due to the regulation and transparency with regards to loan applications, "black-box" machine learning models (e.g., deep learning, tree-based model, and SVM) may not be a suitable approach for predicting the credit risk of borrowers. However, the logistic regression model provides good transparency on the relationship between predictors and the process of decision making. It is easier for the financial institution to interpret contributing factors to the default probability. An extension from default probability prediction, the dynamic behavioural scoring model, which predicts when the borrowers are likely to default (Wang et al., 2018), an advantage over classifications into default and non-default loans only. The logistic regression model is capable to provide probability outcomes to indicate the degree of influence from the variables on the loan default probability. Table 1 shows the summary of papers reviewed in this paper.

Table 1: Literature review summary.

Authors	**Machine Learning Model**	**Summary**
Setiawan et al. (2019)	Extremely Randomised Tree (ERT) and Random Forest (RF)	The evaluation of the models using Lending Club data revealed that the average performance of ERT can outperform RF.
Emekter et al. (2014)	Binary Logistic Regression	Selection bias evaluation with data of the United States national loan consumers, and data of Lending Club consumers shows insignificant difference of default probability. Consideration of data beyond the Lending Club platform is unnecessary.
Zhou et al. (2019)	Gradient Boosting Decision Trees (GBDT) with heterogeneous ensemble learning technology	Ensemble learning model optimized prediction from a high dimension and imbalance P2P lending platform dataset.
Dong et al. (2010)	Logistic Regression with Random Coefficients (LRR) and Fixed Coefficients (LRF)	The LRR outperformed LRF with higher overall accuracy. Logistic regression is an optimal solution for credit scoring model for financial industry in term of result interpretability.
Wang et al. (2015)	Lasso-logistic Regression Ensemble Learning algorithm (LLRE)	LLRE outperforms all the compared models of other families to predict default probability from imbalanced large dataset.
Coenen et al. (2021)	Generalized Linear Models, Support Vector Machines and Gradient-boosted Trees	Regression method shows higher consistency in getting high scores among all the method families in all the evaluation tasks.

3 Methodology

3.1 Model Formulation

Evident from our literature review, the logistic regression method is used of which the model equates the logit transform, the log-odds of the probability of a success, to the linear component as formulated in equation 1.

$$ln(\frac{p}{p-1}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_k x_k, \quad (1)$$

where p is the probability of loan to default and thus, 1-p is the probability of non-default loan occurred. The hypothesis function, is defined as: $h_\beta(x) = P(Y = 1/x\,; \beta)$, representing the predicted probability of loan, Y, to default corresponding to the loan information, x, as the independent variable and parametrised by β. In supervised learning, Y represents the label column with the value 1, representing a default loan and 0 indicating a non-default loan. Here, β represents the coefficients corresponding to each feature for fitting the model.

By rearranging equation (1), an expression for p is thus obtained as in equation (2):

$$p = \frac{1}{1+e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_k x_k)}} \quad (2)$$

Equating $h_\beta(x)$ and p, then our hypothesis function is simplified to:

$$h_\beta(x) = \frac{1}{1 + e^{\beta^T} x} \quad (3)$$

where β^T is the vector of coefficients corresponding to the independent variables x. The parameters β, of a logistics regression function, were estimated using the maximum likelihood method. Employing the likelihood function:

$$L(\beta; y|x) = \prod_{i=1}^{n} (\frac{\pi_i}{1-\pi_i})^{y_i} \cdot (1-\pi_i)^{n_i} 1 \tag{4}$$

where,

$$\pi_i = \frac{e^{\sum_{k=0}^{K} x_k^{(i)} \beta_k}}{1+e^{\sum_{k=0}^{K} x_k^{(i)} \beta_i}},$$

leading to L (β; y | x) to be

$$\prod_{i=1}^{n} (e^{y_i \sum_{k=0}^{K} x_k^{(i)} \beta_k}) \cdot (1 + e^{\sum_{k=0}^{K} x_k^{(i)} \beta_k})^{-n_i} \quad .$$

Taking the logarithm of the likelihood function, resulting in

$$l(\beta) = \sum_{i=1}^{N} y_i (\sum_{k=0}^{K} x_k^{(i)} \beta_k) - n_i \cdot log(1 + e^{\sum_{k=0}^{K} x_k^{(i)} \beta_k}). \tag{5}$$

To determine the critical point of the likelihood function, the partial derivative of the likelihood function with respect to each β_k, where $k=1, 2, 3, \ldots, K$, are found and set them equal to zero. To simplify the process of finding derivative, the logarithm of likelihood function, $l(\beta)$ given in equation (5), is used.

3.2 Data

The dataset used is provided by the Lending Club, a peer-to-peer lending company from the United States of America (USA). Dataset was made available on Kaggle, an online community of data scientists and machine learning practitioners, by George (2018). The dataset contains approved loan records starting from year 2007 until the year 2018. There is total of 2,260,701 records with 151 columns, each record labelled with corresponding loan status which are 'Fully Paid', 'Current', 'Charged Off', 'In Grace Period', 'Late (31-120 days)', 'Late (16-30 days)', 'Default', 'Does not meet the credit policy. Status: Fully Paid' and 'Does not meet the credit policy. Status: Charged Off'.

3.3 Model Design

Our machine learning process follows the flow depicted in Figure 1.

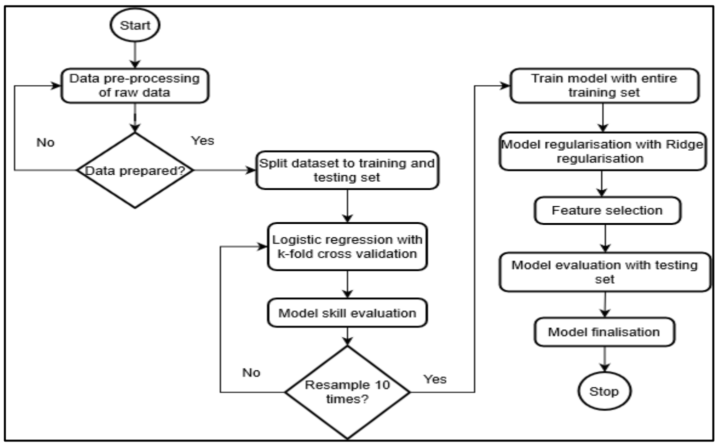
Start
Data pre-processing of raw data
No
Yes
Data prepared?
Split dataset to training and testing set
Logistic regression with k-fold cross validation
Model skill evaluation
No
Resample 10 times?
Yes
Train model with entire training set
Model regularisation with Ridge regularisation
Feature selection
Model evaluation with testing set
Model finalisation
Stop

Figure 1: Machine learning process flow.

As part of data pre-processing, columns with more than 49% of missing data were removed from the dataset. Data rows with missing value are labelled as default record (minority label) and the missing data is imputed with mean, median or mode. In addition, columns which cause data leakage and column of biographical data are also removed. Outliers are detected using box plot and are removed from the dataset. Data rows with loan status labelled as 'current' which do not indicate final status of loan are also removed. The labels are grouped into default or non-default, with the values 1 and 0 respectively. 1, 345, 350 records with 25 columns were the size of the dataset used in the experiments. Refer to Table A in appendix section for the description of columns selected. Once the dataset was pre-processed then it was split in the ratio of 80% for training and 20% for testing. Stratified sampling was implemented to ensure the default and non-default records were distributed evenly. There were 268,599 (19.96%) default records and 1,076,751 (80.04%) non-default records. Under-sampling is applied to majority class, the non-default class with NearMiss-3 algorithm. M number of closest majority samples for each minority samples are kept. Then, majority samples with the largest average distance to k nearest-neighbours (the minority samples) are selected. NearMiss-3 ensures each default sample is surrounded with non-default samples and those samples which are more distinct are kept for model fitting. In model training, the 10-fold cross validation is implemented for finding the best tuned parameters. The 80% of training dataset is partitioned into 10 blocks, and validation is iterated for 10 times. For each iteration, 9 blocks are used as training sets and the remaining block is held out from training and used as a test set. Data resampling in each iteration is done by randomly choosing 9 data blocks from the original dataset and are combined into a training set for cross validation, while the remaining one block is used as testing set.

For feature selection, a null hypothesis, h_{null}, is used that stated that there is no relationship between the independent variables and the dependent variable with 0.05 significance level. A significant level of 0.05 is chosen based on the recommendation from Fisher (2022) where it is approximately twice the standard deviations from the mean of normal distribution. Coefficients with p-values of more than 0.05 falling within the confidence level are eliminate with backward elimination where significant test of the independent variables started with the full model. Least significant variable is removed from the model until only the remaining independent variables that have significant contribution to the dependent variable are left. This method can show the joint behaviour of all variables in a full model, thus avoiding removal of variable which is less significant when it is include independently into the model (Chowdhury & Turin, 2020). To avoid overfitting of the model, L2 regularisation (also known as Ridge regularisation) was adopted. Ridge is a method which shrinks the weight of less important coefficient towards zero without reaching the value zero.

Models trained using the best features selected is evaluated by plotting the Receiver Operating Characteristics (ROC) curve. Recall, precision, and F1-score for default loan (minority class) are used to evaluate the model performance and for fine-tuning decision threshold which gives the best model performance. Recall measures the fraction of correctly classified positive sample (true positive). Precision measures the fraction of correct predictions made among all the positive predictions. However, recall and precision are trade-off whereby the increase of recall causes decreases in precision and vice versa. Therefore, F-measure or F1-score is used to measure the harmonic mean of precision and recall. Logistic regression model with the highest F1-score will be chosen and used in the model finalisation stage. The evaluation of the model is done using unseen data.

4 Implementation and Testing

In this section, Pearson correlation is applied to analyse the correlation between numerical features and remove features which causes multicollinearity. Outliers for each numerical feature are removed based on the upper and lower inner fences of the data distribution. Categorical features are encoded and transformed into dummy variables. Missing values in the dataset are imputed with the corresponding median in the testing dataset. Medians for imputing the missing values in both training and testing were computed from the training set alone to avoid data leakage from the isolated testing set which causes the predictive model to know information of unseen dataset. Standardisation is applied to all numerical columns in the dataset using the formula given in equation (6),

$$X' = \frac{X-\mu}{\sigma} \quad . \tag{6}$$

Rescaling the features using standardisation allows fair comparison of impacts of independent variables on the dependent variable based on weight of coefficients. Table 2 shows the mean and variance for standardising each feature.

Table 2: Mean and variance for feature scaling.

Features	Mean (5 decimal place)	Variance (5 decimal place)
loan_amt	14333.05653	69883770.0
annual_inc_log	4.81408	0.04229
dti	18.29155	67.23525
pub_rec	0.20219	0.25033
revol_bal_log	4.04207	0.12949
revol_util	54.71481	518.36330
mo_sin_old_il_acct	123.36687	1750.75500
mo_sin_old_rev_tl_op	167.76806	5898.07600
mort_acc	1.52570	3.037950
num_rev_accts	13.84327	44.18611
FICO_mean	694.16450	676.26580

The full dataset of 1,060,604 records and 47 columns (with dummy variables created from categorical variables) was split in stratified fashion with respect to the label column (dependent variable) where 80% was taken as the training dataset and the 20% was the testing dataset. Stratify splitting ensures both training and testing set have the same ratio of default and non-default records. The distribution of records grouped with loan status is shown in Table 3. Table 4 shows the distribution of under-sampled records using NearMiss-3.

Table 3: Number of records in training and testing dataset grouped with loan status.

Datasets	**Training**	**Testing**
Non-default	679,397	169,850
Default	169,086	42,271

Table 4: Number of records in training and testing dataset grouped with loan status.

Datasets	**Training**	**Testing**
Non-default	169,086	169,850
Default	169,086	42,271

Receiver-Operator-Characteristic (ROC) area-under-the-curve (AUC) was used as test score in the cross-validation. Table 5 and Table 6 show the performance of the models fitted to imbalanced and under-sampled balance dataset in 10-fold cross validation.

Table 5: Logistic regression model 10-fold cross-validation performance with Ridge regularisation of $10^{-5} \leq \lambda \leq 10^{-2}$

Magnitude of Penalty Term	**Mean Model Fitting Time (s)**	**Mean ROC AUC**	**Performance Ranking**
0.00001	13.4482	0.6974	13
0.0000177828	7.9149	0.7002	10
0.0000316228	6.5548	0.7009	9
0.0000562341	7.0857	0.7043	6
0.0001	6.7144	0.7051	4
0.000177828	6.7424	0.7057	2
0.000316228	6.2994	0.7058	1
0.000562341	6.3834	0.7054	3
0.001	6.2227	0.7045	5
0.00177828	6.2059	0.7031	7
0.00316228	5.9686	0.7014	8
0.00562341	5.7796	0.6997	11
0.01	4.7390	0.6980	12

In Table 5, logistic regression model with λ=0.000316228 (actual value is 0.00031622776601683794) has the best ROC AUC score of 0.7058.

Table 6: Logistic regression model 10-fold cross-validation performance with Ridge regularisation of $10^{-5} \leq \lambda \leq 10^{-2}$ using NearMiss-3 under sampled training dataset.

Magnitude of Penalty Term	**Mean Model Fitting Time (s)**	**Mean ROC AUC**	**Performance Ranking**
0.00001	6.3202	0.6719	11
0.0000177828	5.0328	0.6704	13
0.0000316228	4.0227	0.6715	12
0.0000562341	3.5477	0.6747	9
0.0001	3.2301	0.6733	10
0.000177828	2.9689	0.6747	8
0.000316228	2.7428	0.6771	6
0.000562341	2.4805	0.6761	7
0.001	2.3890	0.6779	3
0.00177828	2.4373	0.6781	1
0.00316228	2.5117	0.6779	2
0.00562341	2.3430	0.6777	4
0.01	2.1376	0.6774	5

In Table 6, logistic regression model with λ=0.00177828 (actual value is 0.0017782794100389228) has the best ROC AUC score of 0.6781. L2 logistic regression model with the best performance penalty term, λ, is fitted to complete imbalanced and balanced training sets. Logistic regression model is fitted to imbalanced training dataset with λ=0.00031622776601683794 and is fitted to under-sampled training data with λ=0.0017782794100389228.

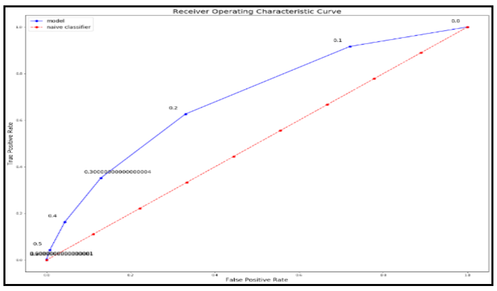

Figure 3: ROC AUC plot for logistic regression classifier of imbalanced dataset with threshold labels.

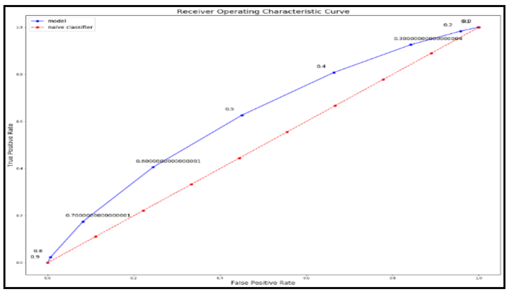

Figure 4: ROC AUC plot for logistic regression classifier of under-sampled dataset with threshold labels.

ROC curve shown in Figure 3 indicates that the threshold of 0.2 give reasonable classification result with high TPR but relatively low FPR for the model trained with imbalanced dataset. Figure 4 shows that threshold of 0.5 is the best threshold for model fitted to balanced dataset. To further evaluate the choice of suitable decision threshold, precision, recall and F1-score for default loan classification at each threshold was computed. F1-score is used to find the harmonic mean of recall and precision. F1-score ranges from 0.0 to 1.0 where 1.0 for perfect recall and precision. Figure 5 and Figure 6 show the changes of precision and recall against decision thresholds for model fitted to imbalanced and balanced dataset.

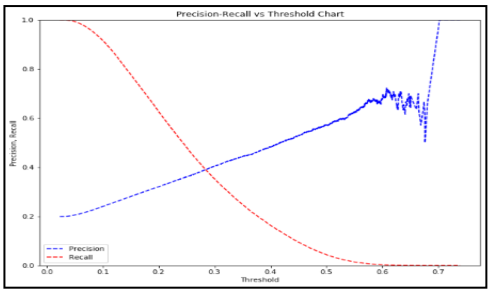

Figure 5: Default class's precision-recall curves of model trained with imbalanced data.

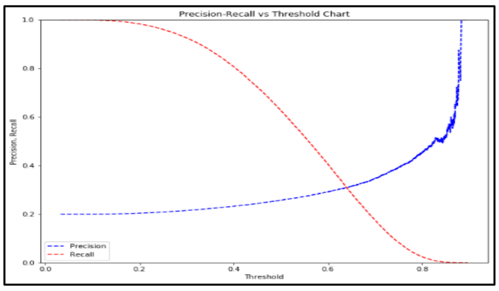

Figure 6: Precision-recall curves of model trained with under-sampled balance data.

High recall can trade off precision, therefore F1-score is used to seek balance between recall and precision. Thus, threshold which gives the highest F1-score is preferred.

Table 7: Default class's precision, recall, and F1-score of model trained with imbalanced data.

Threshold	Precision	Recall	F1-score
0.0	0.19928	1.00000	0.33233
0.1	0.24036	0.91649	0.38084
0.2	0.32084	0.62653	0.42437
0.3	0.40435	0.35242	0.37660
0.4	0.48275	0.16349	0.24426
0.5	0.57200	0.04369	0.08119
0.6	0.68000	0.00282	0.00561
0.7	1.00000	0.000047	0.000095

Result in Table 7 shows that logistic regression model fitted to imbalanced dataset gives the highest F1-score of 0.42437 with precision of 0.32084 and recall of 0.62653 at threshold of 0.2.

Table 8: Default class's precision, recall, and F1-score of model trained with under-sampled balance data.

Threshold	Precision	Recall	F1-score
0.0	0.19928	1.00000	0.33233
0.1	0.19960	0.99924	0.33274
0.2	0.20360	0.98306	0.33734
0.3	0.21480	0.92560	0.34868
0.4	0.23251	0.80736	0.36104
0.5	0.25702	0.62698	0.36458
0.6	0.29215	0.40586	0.33974
0.7	0.34672	0.17506	0.23265
0.8	0.45281	0.02418	0.04590

From Table 8, logistic regression model fitted to balanced dataset gives the highest default class's F1-score of 0.36458. The respective default class's precision is 0.25702 and 0.62698 for recall at threshold of 0.5.

5 Results and Discussion

Area under the Receiver-Operator-Characteristic (ROC) curve measures the ability of the classification model to distinguish between two classes. The larger the area under the curve (AUC), the better the proposed model to distinguish between the classes. The baseline of ROC curve is a straight diagonal line with AUC = 0.5, indicating a random classifier which makes a random guess on the distinction between the two classes.

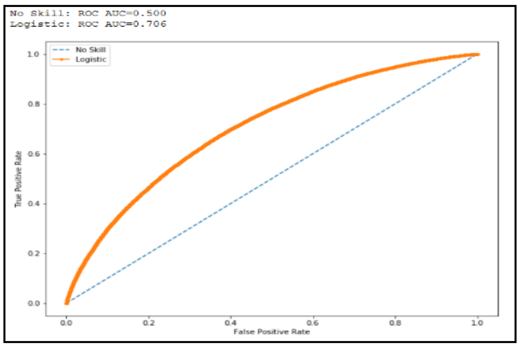

Figure 7: ROC AUC plot of model trained with imbalanced dataset.

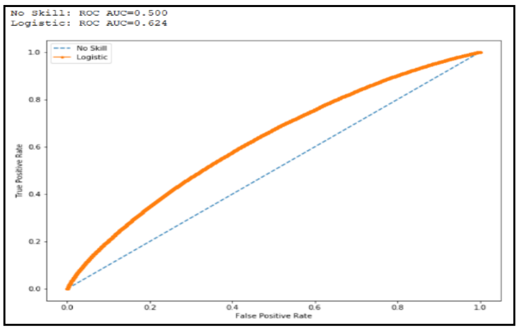

Figure 8: ROC AUC plot of model trained with under-sampled balance dataset.

Figure 7 depicts the model fitted to imbalanced dataset giving ROC AUC value of 0.706 whereas Figure 8 shows the model fitted to balanced dataset giving the value of ROC AUC to be 0.624. The difference in the ROC AUC values indicates that the model fitted to imbalanced dataset has better capability to differentiate between the default and non-default classes than the model fitted to balanced dataset. However, ROC AUC measures the overall classification performance of the model without considering the effect of majority class which cause the algorithm to be bias towards the non-default class. Due to the large skewed class distribution, ROC may give over-promising evaluation on an algorithm performance (Davis & Goadrich, 2006).

Precision-recall curve (PRC) is a better alternative of ROC for evaluating the performance of binary classifier on an imbalanced dataset. Unlike fixed baseline of ROC, baseline of PRC changes with the ratio of positive (P) and negative (N) class in the dataset. PRC baseline is defined as y = P/ (P+N) and AUC of no-skill classifier is identical to y position of PRC baseline (Saito & Rehmsmeier, 2015).

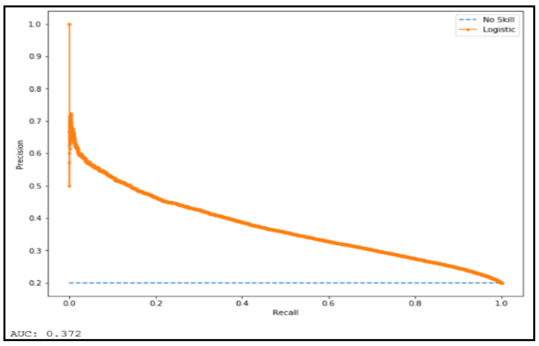

Figure 9: Precision-recall curve of model fitted to imbalanced dataset.

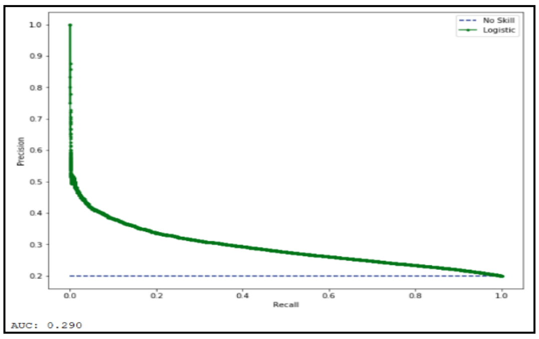

Figure 10: Precision-recall curve of model fitted to balanced dataset.

Based on the results shown in Figure 9 and Figure 10, AUC of no-skill classifier is found to be 0.2. Both models have AUC larger than the no-skill classifier which indicates they are not random classifier. Model fitted to the imbalanced dataset has larger AUC of 0.372 than model fitted to balanced dataset with AUC of 0.290. Therefore, the model fitted to an imbalanced dataset outperforms the model fitted to balanced dataset in distinguishing between two classes.

A total of 212,121 samples of the Lending Club loan records from isolated testing dataset were used to make predictions using two logistic regression models where one model is fitted to imbalanced dataset and the other one fitted to balanced dataset. The testing set contains 169,850 non-default samples and 42,271 default samples.

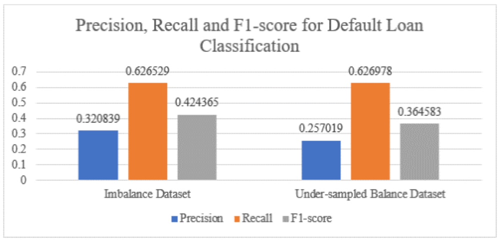

Figure 11: Comparison of default class's precision, recall, and F1-score of logistic regression models trained on imbalanced and under-sampled balance training dataset.

Based on the histogram shown in Figure 11, recalls of both models do not have significant difference where 0.626529 was found for imbalanced dataset and 0.626978 for balanced dataset. However, the model trained with imbalanced dataset has both higher F1-score and precision than model trained with balanced dataset. The result shows that NearMiss-3 under-sampling method does not improve the model performance on classifying default and non-default loan. NearMiss-3 ensures positive and negative samples with significant difference are selected while allowing positive samples to be surrounded by some majority samples. In exchange of keeping positive samples surrounded by negative samples, overlapping of both classes occurs causing a decrement for positive class precision. Precision evaluates fraction of exactly positive samples which are correctly classified as positive. Although high recall allows classification model to become more sensitive to positive class, high precision is important for avoiding misclassification of non-default loan and good clients. Hence, in comparison of the two models, it is found that the model trained with imbalanced dataset has better performance evident from the higher precision obtained and the evaluation of the F1-score. In feature selection, the logistic regression model fitted to the imbalanced dataset is employed. The model has 47 independent variables with a constant variable, which is the model intercept. Hypothesis testing with *p*-value computed from *t*-test is implemented to select statistically significant features. Defining a null hypothesis, H_0, of which the feature is insignificant to the default probability of client, tested with p-values of the feature at significance level, α of 0.05. Backward elimination was implemented for feature selection where the most insignificant feature is removed, and model is retrained with the remaining features before the next significance test is carried out. The steps are repeated until no insignificant features are left.

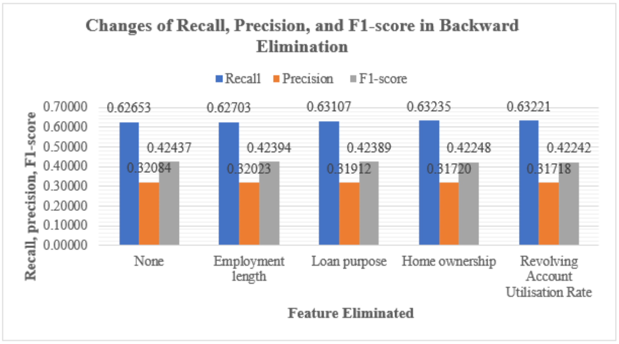

Figure 12: Changes of Recall, Precision, and F1-score in Backward Elimination.

Figure 12 shows positive class' recall increased from 0.62653 to 0.63235 after the elimination of insignificant categorical features. However, the precision decreases from 0.32084 to 0.31720 and the F1-score decreases from 0.42437 to 0.42248. Based on the three evaluation criteria, it was found that none have shown significant changes, and thus, further support the hypothesis test for which employment length, loan purpose and home ownership are insignificant to the probability of client to default in loan. The elimination of insignificant categorical features has revealed that revolving account utilisation rate, "revol_util", is insignificant with p-value of 0.866 which is larger than 0.05 significance level. Elimination of revolving account utilisation rate from the model causes the recall to drop from 0.63235 to 0.63221. There is also insignificant drop for precision and F1-score which changes from 0.31720 to 0.31718 and 0.42248 to 0.42242 respectively. Upon the implementation of the feature selection with backward elimination, 19 features were selected.

In model finalisation phase, all data samples available are used for model fitting including those from testing set which is isolated from model fitting previously. A list of new coefficients correspond to each feature is obtained. Table 9 shows the coefficients obtained by fitting the logistic regression model to the full dataset.

Table 9: Coefficients of logistic regression model fitted to complete dataset of Lending Club loan record from year 2007 until the fourth quarter of year 2018.

Features	**Coefficients, β**	**Exp(β)**
Intercept	-2.3733	0.0932
Loan amount	0.1592	1.1726
Log-transformed borrowers' annual income	-0.0730	0.9296
Debt to income ratio	0.1942	1.2143
Public derogatory records	0.0160	1.0161
Log-transformed total credit revolving balance	-0.1002	0.9047
Months since oldest bank instalment account opened	-0.0271	0.9733
Months since oldest revolving account opened	-0.0491	0.9521
Number of mortgage accounts	-0.1919	0.8254
Number of revolving accounts	0.0435	1.0445
Mean FICO score	-0.1782	0.8368
64 months payment term	0.5531	1.7386
Credit rating: Grade B	0.3205	1.3778
Credit rating: Grade C	0.7255	2.0658
Credit rating: Grade D	0.9880	2.6859
Credit rating: Grade E	1.1781	3.2482
Credit rating: Grade F	1.2641	3.5399
Credit rating: Grade G	1.2147	3.3693
Income source verified by borrower's employer	0.1353	1.1449
Income source verified by Lending Club	0.1041	1.1097
Joint application	-0.0811	0.9221

The discourse on features is separated into two parts which are discussion on continuous numerical features and the other on categorical features represented in dummy variables. All continuous numerical features are standardised to obtain standardised regression coefficients which allow comparison of absolute values to determine their relative importance in the logistic regression model. According to the absolute value of standardised coefficients, the top three most important numerical features are found to be debt to income ratio, total number of mortgage account and FICO score.

Debt-to-income ratio (DTI) with coefficient of 0.1942 has positive relation with the dependent variable which indicates that the higher the DTI, the higher the chance of borrower to default on loan. DTI is the ratio calculated by dividing monthly debt obligation with monthly gross income. Therefore, DTI reflects the ability of borrower to secure a loan whereby high DTI indicates the borrower is less likely to afford extra debt with the current income. Standardised coefficient of the total number of mortgage account is -0.1919 which defines borrower with more mortgages has lower loan default rate. Mortgage is a secured loan with real asset as collateral, and the evaluation on the ability of applicant to afford the real asset is used for mortgage application from financial institution, as it is suspected that borrower with several mortgage indicates that their credit records are good enough to fulfil the requirements of getting the mortgage loan. This explains the research outcome that the borrower with more mortgage account has lower probability of default (POD) than those with less mortgage record. The third important feature for predicting POD is the mean FICO score. Negative coefficient of -0.1782 indicates that borrowers with high FICO score tends to pay off the loan. Formula behind FICO credit score is kept secret from customers, but there are five key factors for FICO score credit report disclosed by FICO which are payment history, account owed, credit history, credit mix, and new credit. According to a study carried out by Avery, Brevoort and Canner (2012) on the effect of credit history length on credit score of foreign-born individuals in U.S. made, short credit history has caused lower credit score in this population. The result supports the consideration of length of credit history in FICO credit report where individual with longer credit history tends to have better credit score.

Next, loan amount has positive coefficient of 0.1592 which indicates that loan of higher amount has greater POD. The larger the amount loan offered by the Lending Club, the higher the interest charged which indicates higher risk. Credit revolving balance is the next important independent variables for predicting POD with negative coefficient of -0.1002. Revolving balance is the carried forward unpaid balance after each payment cycle of a credit account. In general, higher debt owed leads to higher POD, but the occurrence of negative coefficient of credit revolving balance shows that amount of debt owed does not directly reflect the POD of a borrower. It is shown that the amount of outstanding credit balance is positively correlated to income and amount of real asset owned by an individual (Kim & Devaney, 2001). High-income population have higher credit limit, hence rising their purchasing power and increase the revolving balance subsequently. With the same reason, it explains the finding that an increase in mortgage account implies a decrease in the probability of default, since better financial status allow a person to afford more mortgages. In fact, one can have high debt amount but with large available credit, whereas an individual who owes less debt may have less credit available or even max out credit card. Due to this reason, credit scoring model such as FICO score will consider credit utilisation rate which provide more informative debt to credit limit ratio. Number of months since oldest bank instalment account opened and number of months since the oldest revolving account was opened, have negative coefficient of -0.0271 and -0.0491 respectively. These two coefficients are complementing to the fact that individuals with longer credit history have better credit. Credit scoring models available in market always consider length of credit account since it is opened and used, as well as average age of all account owned by a borrower for credit evaluation since all this information reflect the attitude of the individual towards their credit. Among all numerical features, the number of public derogatory records is the least important predictor for POD with positive coefficient of 0.0160. Public derogatory records include tax liens, public bankruptcy record and any financial obligation which are not paid as agreed. It is reasonable that individual with more public derogatory record tends to have bad credit history resulting in loan repayment failure.

Discussion on categorical features is made by comparing how each dummy variables or category level contribute to the probability of default. To measure the contribution of reference category to POD, only one categorical variable remained in the model each time. POD of reference category is determined by intercept or constant of the model while all numeric predictors remained zero. Since the logistic regression model calculates the log-odds of loan default, equation (7) is used for the transformation to POD:

$$p = \frac{1}{1+e^{-\beta_x^T}} \tag{7}$$

Table 10: Probability of default for each category in loan payment term.

Category	Coefficients, β	Exp(β)	Probability of Default
Intercept (36 months)	-1.7533	0.1732	0.1476
64 months	0.8733	2.3948	0.2932

Payment term (term) feature has two level of categories namely 36 months and 64 months. 36 months payment term is the reference category and "term_code_1" indicates 64 months loan payment period. From Table 10, POD of 36-months payment period is determined by intercept value. Loan with longer payment period, which is 64 months, has higher POD of 29.32% than loan with 36 months payment term (POD = 14.76%). The major loan purpose in Lending Club is debt consolidation, which is a type of loan of combining 2 and more loans into single mortgage. Delinquency of mortgage loan is closely related to income volatility even for high-income profile (Diaz-Serrano, 2005). Since income volatility may increase over time, thus extending the loan payment period can subsequently increase the risk of loan.

Table 11: Probability of default for each category in credit grade.

Category	Coefficients, β	Exp(β)	Probability of Default
Intercept	-2.2680	0.1035	0.0938
Grade B	0.3743	1.4540	0.1308
Grade C	0.8880	2.4302	0.2010
Grade D	1.2103	3.3545	0.2577
Grade E	1.5213	4.5782	0.3215
Grade F	1.6699	5.3116	0.3548
Grade G	1.5878	4.8930	0.3362

According to Table 11, the reference category, Grade A has the lowest POD of 9.38% and the POD are 13.08%, 20.1%, 25.77%, 32.15%, 35.48%, and 33.62% for grade B, C, D, E, F, and G respectively. Although Grade F is having the higher POD than the worst credit rating of Grade G, but the risk of default increase as the risk goes higher. It is reasonable to conjecture that Lending Club rating system is reliable reference for other financial institution while evaluating borrower's credit.

Table 12: Probability of default for each category in income source verification status.

Category	Coefficients, β	Exp(β)	Probability of Default
Intercept	-1.6672	0.1888	0.1588
Source Verified, income source verified by borrower’s employer	0.2205	1.2467	0.1905
Verified, income source verified by Lending Club	0.2542	1.2894	0.1958

According to the Table 12, loan without income source verification has the lowest POD of 15.88% while POD of loan with “source verified” and “verified” are 19.05% and 19.58% respectively. The label “income source verified” defines that Lending Club had contacted the borrower’s employer to verify his or her claim on the amount of earning; “income verified” defines the situation when Lending Club verified that the earning amount claimed by the borrower is within an acceptable range. According to Lending Club’s company data obtained by Bloomberg, only 35.6% of income sources for application of popular loan types are verified in 2016 (Scully, 2017). As explained by Lending Club, verification of income is not applied to initial application which already passed their screening model, and the applicant is considered by Lending Club as lower risk borrower. However, Blackburn and Vermilyea (2012) found out that misstated income from borrower is one of the major causes for default on mortgage loan. Thus, the low POD of unverified income is inappropriate to explain credit level of borrower who does not passed Lending Club screening model.

Table 13: Probability of default for each category in loan application type.

Category	Coefficients, β	Exp(β)	Probability of Default
Intercept	-1.4981	0.2236	0.1827
Joint application	-0.1387	0.8705	0.1629

Lending Club allows joint application for single loan. Lending Club considers information from one of the applicants or both as factors to decide whether to approve or reject the loan. Both co-borrowers have the obligation to pay loan payment once the loan is approved. From Table 13, POD of reference category, individual application is 18.27% which is riskier than joint application with POD of 16.29%. Joint application for loan usually offered to population with short and incomplete credit history especially to those in undeveloped region, and it is proven to outperform individual application in term of repayment performance (Zhou & Wei, 2020).

The objectives of this research are to create a machine learning model which can predict probability of default and classifies the client’s based on their ability to pay the loan. In this research, loan applicant with POD higher or equal to 20% is classified as default while POD lower than 20% is classified as non-default class.

6 Conclusions and Recommendations

The objectives of this research were to create a less bias solution that not only define client’s credit through FICO score, but also a comprehensive evaluation that considers other factors related to the client for predicting probability of default (POD) using machine learning model. Logistic regression model fitted to imbalanced dataset outperforms model fitted to balanced dataset. Evaluation using area under precision-recall curve validates the model built for default loan classification is not a random classifier. Decision threshold value which achieves maximum balance between model recall and precision is selected with the highest F1-score.

Top 3 important features that affect POD are debt-to-income ratio, number of mortgage account, and FICO score. High debt-to-income ratio significantly contributes to the rise of POD. Revolving balance feature provides evidence to support the fact that the amount of outstanding payment does not reflect credit of an individual. However, high buying power due to high credit limit and good financial status can cause more revolving balance owed by an individual. This suggests that the utilisation rate of credit account and debt-to-income ratio are better evaluation factors for credit risk. The research result shows that FICO score is still an important factor for credit evaluation in P2P lending platform. The number of months since oldest bank instalment account opened and the number of months since oldest revolving account opened both have negative coefficient, which support the consideration of credit history length as effective factor for credit evaluation. Both mortgage number and credit history length explained the low credit score of inexperience borrower with short credit history and the reason why financial institutions prefer to allocate more resource to experienced borrowers.

The result of our model suggests that lenders should take extra precaution while dealing with borrowers who are having more public derogatory records and offering higher amount of loan is riskier. Besides, lenders should also beware of longer loan term which can increase the risk due to uncertainty causes by borrower’s income volatility.

Nevertheless, credit rating model from the Lending Club is proven to have significant contribution on determining the risk of borrower in P2P lending platform, where the lower the grade of borrower the riskier he or she is. The model suggests that income source claimed by borrowers should be further verified with their employers to avoid misstate of income source which can increase the risk. Lack of credit score such as FICO score among SME entrepreneurs also posed difficulties while applying for loan. Thus, joint application of loan with better repayment performance is suggested as an alternative to offer loan to high-risk borrowers in online P2P lending platform.

Limitation of machine learning model proposed is that the model is fitted to imbalanced dataset. This causes the decision threshold for classifying default loan is set low to 0.2 for achieving highest F1-score. Low threshold value leads to higher false positive rate and causes loss of potential excellent borrowers. More appropriate resampling method can be applied for creating balanced dataset. As suggested by Yen and Lee (n.d.), different clusters in a dataset have their own characteristic where clusters with more majority samples than minority will behave like majority class, and a cluster will pose characteristic of minority class if it has more minority class samples. Under-sampling method based on clustering can be carried out to select majority class sample which may help the machine learning algorithm to better classifying default and non-default loan. Moreover, dataset from Lending Club contains loan records range from year 2007 until the fourth quarter of year 2018 which also includes records during the 2008 US financial crisis. Thus, it is suggested to select subset of data for model training based on economic situation such as economic downturn and economic upswing.

In conclusion, logistic regression model proposed provides human-interpretable information of how borrower's information and loan type affect the probability of default of loan on online P2P lending platform. Logistic regression model ensures the transparency of decision-making for loan approval and rejection which satisfy the requirement of Central Bank of Malaysia. it is hoped that the result obtained in this research can help local P2P lending platform in Malaysia to improve their credit screening process, hence provide a reliable online financial platform for both lenders and SME entrepreneurs.

Acknowledgement

The authors wish to thank Universiti Malaysia Sarawak for the facilities provided during the running of this research project.

References

Avery, R. B., Brevoort, K. P., & Canner, G. (2012). Does Credit Scoring Produce a Disparate Impact? *Real Estate Economics, 40*. doi:10.1111/j.1540-6229.2012.00348.x

Bachmann, A., Becker, A., Buerckner, D., Hilker, M., Kock, F., Lehmann, M., & Tiburtius, P. (2011). Online Peer-to-Peer Lending – A Literature Review. *Journal of Internet Banking and Commerce, 16*(23).

Blackburn, M. L., & Vermilyea, T. (2012). The prevalence and impact of misstated incomes on mortgage loan applications. *Journal of Housing Economics*, *21*(2), 151–168. https://doi.org/10.1016/j.jhe.2012.04.003

Chowdhury, M. Z. I., & Turin, T. C. (2020). Variable selection strategies and its importance in clinical prediction modelling. *Family Medicine and Community Health*, *8*(1), e000262. https://doi.org/10.1136/fmch-2019-000262

Coenen, L., Verbeke, W., & Guns, T. (2021). Machine learning methods for short-term probability of default: A comparison of classification, regression and ranking methods. *Journal of the Operational Research Society*, *73*(1), 191–206. https://doi.org/10.1080/01605682.2020.1865847

Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*. https://doi.org/10.1145/1143844.1143874

Diaz-Serrano, L. (2005). Income volatility and residential mortgage delinquency across the EU. *Journal of Housing Economics*, *14*(3), 153–177. https://doi.org/10.1016/j.jhe.2005.07.003

Dong, G., Lai, K. K., & Yen, J. (2010). Credit scorecard based on logistic regression with random coefficients. *Procedia Computer Science*, *1*(1), 2463–2468. https://doi.org/10.1016/j.procs.2010.04.278

Emekter, R., Tu, Y., Jirasakuldech, B., & Lu, M. (2014). Evaluating credit risk and loan performance in online Peer-to-Peer (P2P) lending. *Applied Economics*, *47*(1), 54–70. https://doi.org/10.1080/00036846.2014.962222

Fisher, R. A. (2022b). *Statistical Methods for Research Workers, 12th Ed. Rev.* (Twelfth Edition). Oliver and Boyd.

George, N. (2018). *All Lending Club loan data 2007 through current Lending Club accepted and rejected loan data*. Kaggle. https://www.kaggle.com/wordsforthewise/lending-club?select=accepted_2007_to_2018Q4.csv.gz

Kim, H., & Devaney, S. A. (2001). The Determinants of Outstanding Balances Among Credit Card Revolvers. *Journal of Financial Counseling and Planning, 12*(1).

Meyer, T. (2007, July 10). *Online P2P lending nibbles at banks' loan business*. Retrieved from http://www.venturewoods.org/wp-content/uploads/2007/11/p2p-lending.pdf

Namvar, E. (2013). An Introduction to Peer to Peer Loans as Investments. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2227181

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

Scully, M. (2017, June 14). Biggest online lenders don't always check key borrower data. Retrieved August 29, 2022, from https://www.bloomberg.com/news/articles/2017-06-14/biggest-online-lenders-don-t-always-check-key-borrower-details

Setiawan, N., Suharjito, & Diana. (2019). A Comparison of Prediction Methods for Credit Default on Peer to Peer Lending using Machine Learning. *Procedia Computer Science*, *157*, 38–45. https://doi.org/10.1016/j.procs.2019.08.139

Wang, H., Xu, Q., & Zhou, L. (2015). Large Unbalanced Credit Scoring Using Lasso-Logistic Regression Ensemble. *PLOS ONE*, *10*(2), e0117844. https://doi.org/10.1371/journal.pone.0117844

Wang, Z., Jiang, C., Ding, Y., Lyu, X., & Liu, Y. (2018). A Novel behavioral scoring model for estimating probability of default over time in peer-to-peer lending. *Electronic Commerce Research and Applications*, *27*, 74–82. https://doi.org/10.1016/j.elerap.2017.12.006

Yen, S. J., & Lee, Y. S. (2006). Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset. *Intelligent Control and Automation*, 731–740. https://doi.org/10.1007/978-3-540-37256-1_89

Zhou, J., Li, W., Wang, J., Ding, S., & Xia, C. (2019). Default prediction in P2P lending from high-dimensional data based on machine learning. *Physica A: Statistical Mechanics and Its Applications*, *534*, 122370. https://doi.org/10.1016/j.physa.2019.122370

Zhou, Y., & Wei, X. (2020). Joint liability loans in online peer-to-peer lending. *Finance Research Letters*, *32*, 101076. https://doi.org/10.1016/j.frl.2018.12.024

Appendix

Table A: Description of attributes from Lending Club 2007 to 2018 fourth quarter approved loan dataset.

Attributes	Description	Datatype
annual_inc	The self-reported annual income provided by the borrower during registration.	float64
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers	Object
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.	float64
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.	float64
fico_range_high	Highest FICO score value.	float64
fico_range_low	lowest FICO score value.	float64
grade	LC assigned loan grade	Object
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report.	Object
int_rate	Interest Rate on the loan	float64
loan_amnt	The listed amount of the loan applied for by the borrower.	float64

loan_status	Current status of the loan	Object
mo_sin_old_il_acct	Months since oldest bank instalment account opened	float64
mo_sin_old_rev_tl_op	Months since oldest revolving account opened	float64
mort_acc	Number of mortgage accounts.	float64
num_bc_tl	Number of bankcard accounts	float64
num_rev_accts	Number of revolving accounts	float64
open_acc	The number of open credit lines in the borrower's credit file.	float64
pub_rec	Number of derogatory public records	float64
pub_rec_bankruptcies	Number of public record bankruptcies	float64
purpose	A category provided by the borrower for the loan request.	Object
revol_util	Revolving line utilisation rate, or the amount of credit the borrower is using relative to all available revolving credit.	float64
tax_liens	Number of tax liens	float64
revol_bal	Total credit revolving balance	float64
term	The number of payments on the loan. Values are in months and can be either 36 or 60.	Object
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified	Object

Trends and Future Directions in Automated Ransomware Detection

[1,2*]Abayomi Jegede, [3]Ayotunde Fadele, [4]Monday Onoja, [5]Gilbert Aimufua and [6]Ismaila Jesse Mazadu

[1]Department of Computer Science, University of Jos, Nigeria
[2]Africa Centre of Excellence on Technology Enhanced Learning, National Open University of Nigeria, Abuja, Nigeria
[3]Department of Computer Science, Federal College of Education Zaria, Nigeria
[4]Department of Mathematics and Computer Science, Federal University of Health Sciences, Otukpo, Nigeria
[5]Department of Computer Science, Nasarawa State University, Keffi
[6]Department of Computer Science, Federal University, Wukari, Nigeria

email: [1,2*]jegedea@unijos.edu.ng, [3]ayotundefadele@yahoo.com, [4]mondiono@gmail.com, [5]aimufuagio@yahoo.com, [6]mazadujesse@gmail.com

**Corresponding author*

Received: 21 August 2022 | Accepted: 24 October 2022 | Early access: 28 October 2022

Abstract - *Ransomware attacks constitute major security threats to personal and corporate data and information. A successful ransomware attack results in significant security and privacy violations with attendant financial losses and reputational damages to owners of computer-based resources. This makes it imperative for accurate, timely and reliable detection of ransomware. Several techniques have been proposed for ransomware detection and each technique has its strengths and limitations. The aim of this paper is to discuss the current trends and future directions in automated ransomware detection. The paper provides a background discussion on ransomware as well as historical background and chronology of ransomware attacks. It also provides a detailed and critical review of recent approaches to ransomware detection, prevention, mitigation and recovery. A major strength of the paper is the presentation of the chronology of ransomware attacks from its inception in 1989 to the latest attacks occurring in 2021. Another strength of the study is that a large proportion of the studies reviewed were published between 2015 and 2022. This provides readers with an up-to-date knowledge of the state-of-the-art in ransomware detection. It also provides insights into advances in strategies for preventing, mitigating and recovering from ransomware attacks. Overall, this paper presents researchers with open issues and possible research problems in ransomware detection, prevention, mitigation and recovery.*

Keywords: machine learning, deep learning, neural network, security, ransomware attack, ransomware detection

1 Introduction

Ransomware is malware that hijacks data or systems and prevents legitimate owners of such data or systems from accessing them. Ransomware may encrypt data or lock the system using processes, tools and techniques which make the locking or encryption difficult for a computer expert to reverse. It may also steal sensitive data from victims' computers and networks. Ransomware targets personal computers, business systems (including their data and applications) and industrial control systems. It also attacks internet of things (IoT) spectrum sensors (Celdrán et al., 2022). A ransomware attack uses private key encryption to deny a legitimate user access to his system or data until he pays a ransom (money), usually in bitcoin (Richardson & North, 2017). Ransomware attacks may also involve data exfiltration, whereby attackers copy sensitive files from compromised devices with a threat to revel such files to the public if the owner fails to pay ransom. The malware spreads through email attachments, malicious advertisements and by clicking a link to a malicious website. It locates the drives on the victim's system or network and encrypts the files in each drive to deny the legitimate owners' access to such files (Morhurle &

Patil, 2017). The attacker also provides a file, (or files) which contains instructions for paying the ransom. The decryption key is made available to the victim once the attacker confirms the payment of the ransom. Files infected or encrypted by ransomware usually contain extensions such as .aaa, .micro, .encrypted, .ttt, .xyz, .zzz, .locky, .crypt, .cryptolocker, .vault, or .petya. The extension of each file determines the type of ransomware that infected the file. Examples of ransomware are Reveton, CryptoLocker, CryptoLocker.F and TorrentLocker, CryptoWall, CryptoTear, Fusob and WannaCry (Andronio et al., 2017). Ransomware can be grouped into (1) crypto ransomware, (2) locker ransomware and (3) scareware (Andronio et al., 2017). Figure 1 illustrates the operations of policing (locker) ransomware and encrypting (crypto) ransomware (F-Secure Labs, 2013).

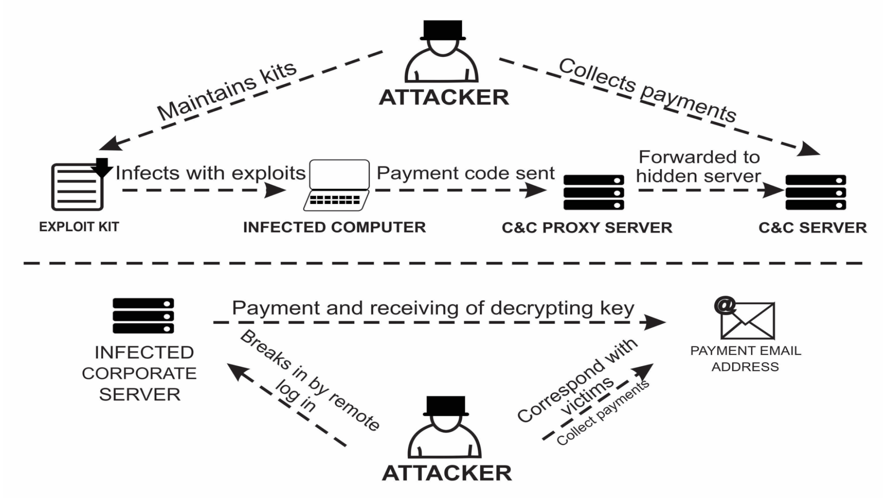

Figure 1: Encrypting ransomware vs. police ransomware operation flowchart

Crypto ransomware is the most common ransomware which attacks computer systems and networks. This category of ransomware uses symmetric and/or asymmetric cryptographic algorithm to encrypt files and data. Crypto ransomware renders encrypted data inaccessible even if the malicious software is removed from an infected device or a compromised storage media is inserted into another device. The infected device can still function and could be used to pay the ransom because the malware does not usually affect critical system files (Savage et al., 2015). Locker ransomware, on the other hand, locks a computer or any other device and prevents the owner from using it (Savage et al., 2015). Locker ransomware affects only the device, without rendering stored data inaccessible. There is also no alteration to the data after the removal of the malicious software. The data can often be recovered by inserting the infected storage device, such as a hard drive, into another system. This makes locker ransomware unattractive for extorting money from victims of attack. A scareware exploits its victims by displaying a warning on their computer screens that the systems have been infected and with a claim that a fake antivirus advertised by the attacker could be used to remove the ransomware (Brewer, 2016). The repeated display of the scareware alert prompts many innocent users to purchase and install the bogus antivirus. Other categories of ransomware include human-operated ransomware (Microsoft Ignite, 2022) and fileless ransomware (Crowdstrike, 2022a). Cyber criminals also use human-operated ransomware to penetrate networks or cloud infrastructure, perform privilege escalation and launch attacks against critical data. It is an active attack which targets an entire organization instead of a single system. Attackers usually leverage on incorrect security configurations to penetrate an entire IT infrastructure, perform lateral movement and exploit vulnerabilities. This results in unauthorized access to credentials of privileged users with the ultimate goal of launching ransomware attacks against IT infrastructures which support critical business operations. Fileless ransomware, on the other hand, uses native and legitimate system tools to launch attacks (Crowdstrike, 2022b). They are difficult to detect because the attack does not require the installation of any code on a victim's system. Hence, anti-ransomware tools do not find any suspicious file to track during an attack. Human-operated ransomware and fileless ransomware may be used to carry out file encryption, locking or data leak depending on the motive of an attacker.

Ransomware poses serious threats to files and devices used by businesses and individuals. It prevents innocent victims from accessing infected files or compromised devices until they pay ransom usually in the form of bitcoin. In many cases, hackers do not provide the decryption key even after a victim pays a ransom. At other times, an attempt to decrypt files using the key provided by an attacker causes further harm to files stored on the system. Technological innovations such as ransomware development kits, ransomware-as-a-service and bitcoins facilitate the persistent increase in ransomware attacks against personal computers, networks and mobile devices (Zetter, 2015). Businesses and individuals suffer losses to the tune of hundreds of millions of dollars annually due to ransomware attacks (Fitzpatrick et al., 2016). The huge amount of money which hackers make from ransomware attacks fuels the frequent development of new versions of the malware. In fact, multiple versions of ransomware have emerged each year since 2013. The evolution of different variants of ransomware which cannot be detected by conventional antivirus and other intrusion detection systems, as well as the huge losses which ransomware attacks inflict on individuals and businesses, highlight the need for innovative, efficient and reliable techniques for effective detection, prevention and mitigation of ransomware attacks.

The paper is novel in the following areas. Firstly, it presents a much more detailed and comprehensive history and chronology of ransomware than other related studies. A related work (Vehabovic et al., 2022) presents the history of ransomware from 2012 to 2021, while our work covers ransomware's history from its inception in 1989 to the latest attacks in 2021. The other study also presents high-level classification of existing ransomware detection methods into four broad categories, with few papers (about forty-seven) reviewed for all the categories, while our paper surveyed almost twice this number and provides much more detailed review of each paper. A significant number of the papers surveyed were published in 2022, unlike the other study which reviewed only a single 2022 paper. Secondly, our paper has a broader scope than the work of McIntosh et al. (2021), which focused primarily on ransomware mitigation, and Oz et al. (2021), whose focus is only on defence/prevention. Our work covers history, detection, defence/prevention, mitigation and recovery. Also, our paper provides an up-to-date review of ransomware attacks by surveying several 2022 papers, while almost all the papers reviewed in McIntosh et al. (2022), and Oz et al. (2021), were published before 2021. Finally, the focus of Dargahi et al. (2019) is completely different from that of our work. The paper presents a taxonomy of crypto-ransomware features using cyber-kill-chain, while the emphasis of our research is on history, detection, defence/prevention, mitigation and recovery. The rest of our paper is divided into the following sections. Section 2 presents the methodology used for the study, while Section 3 covers the historical background and chronology of ransomware attacks. Section 4 discusses the state-of-the-art in ransomware detection, while Section 5 is a review of some methods for preventing, mitigating and recovering from ransomware attacks. Section 6 presents suggestions for future research, while Section 7 is the conclusion of the study.

Stages in Ransomware Attack

Ransomware attack involves a number of phases. Figure 2 illustrates the flow of activities required to carry out such an attack.

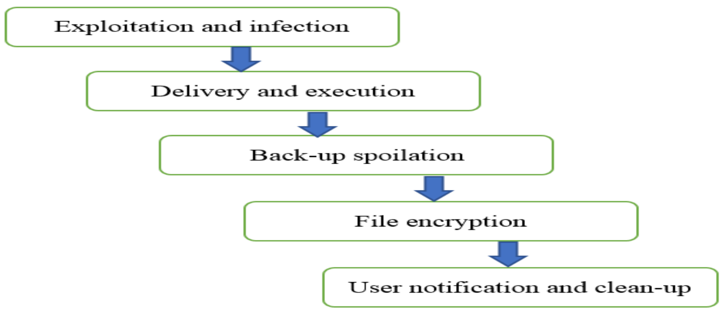

Figure 2: Phases of ransomware attack

An attacker uses exploitation and infection phase to identify vulnerabilities that can be used to launch an attack against a victim computer. The attacker may use a malicious email attachment or an exploit kit for this purpose. For example, the cryptolocker ransomware uses the Angler exploit kit to access and execute on victims' computers. The Angler exploit kit can exploit common vulnerabilities in Adobe Flash and Internet Explorer. The delivery and execution stage involves the installation and execution of the actual ransomware code on the victim's system once there are known vulnerabilities that can support the execution of the malicious payload. Once the file malicious payload executes, it establishes connection with the attacker via the command-and-control mechanism and continues to do further damage. Back-up spoliation involves identification and removal of the system's back-up files and folders to prevent restoration of infected files from back-up. This takes place few seconds after the execution of the ransomware. This is to ensure that victims cannot retrieve compromised files without paying ransom. For example, CryptoLocker and Locky uses vssadmin tool to execute a command that deletes the volume shadow copies from Windows systems. Other variants of these ransomware can identify and delete files from backup folders in order to make recovery a herculean task. File encryption occurs after the removal of backup folders. The process involves a secure key exchange with the command-and-control server to generate encryption keys that will be used to lock the files on the local system. Most modern ransomware variants use strong encryption algorithms such as AES 256 or RSA 1024 which makes it difficult for victims to decrypt infected files.

Ransomware variants such as SamSam performs file encryption locally (on victim systems) without any need to access a command-and-control server via the Internet. Finally, the hacker notifies the victim of the attack and presents instructions for payment of ransom. This occurs after the removal of the back-up files and encryption of the main files. The victim is often asked to pay a ransom within a few days and failure to do so results in an increase in the amount charged for ransom. The payment instructions are usually stored on the hard drive or in the folders containing infected files. At other times, they are saved in specific locations on the hard disk. The malicious executable file automatically deletes itself from the infected system to avoid recovery of useful forensic evidence that would reconstruct the attack and protect against the malware.

2 Methodology

The achievement of the overall objectives of the paper involved the following phases: data collection/information gathering, data extraction/analysis, information synthesis and reporting. Figure 3 is the research process flow, which illustrates the flow of activities involving the phases and the relationship between them.

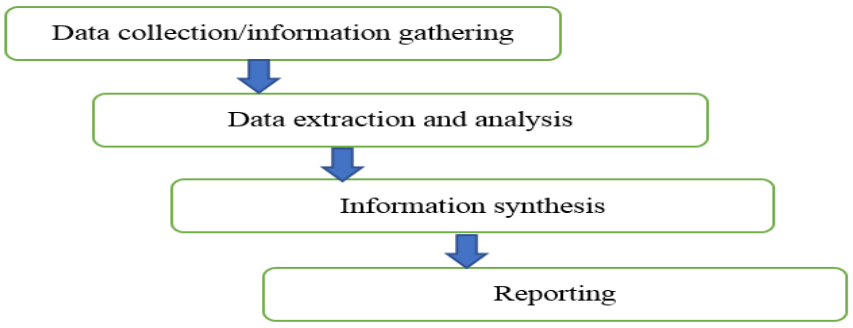
Data collection/information gathering
Data extraction and analysis
Information synthesis
Reporting

Figure 3: Research process flow

Data collection was performed by selecting relevant and up-to-date journal and conference papers from reputable databases such as IEEE, Springer, MDPI, Elsevier, IET and Archive.org. Other sources include university-based journals, thesis/dissertations and blogs published by reputable organizations such as Microsoft, Crowdstrike, Symantec and Techspot. The materials are then grouped into two main categories, namely, non-technical sources and technical sources. Non-technical sources include materials containing general information on ransomware and as such, provide reliable information for writing the sections on introduction and ransomware history/chronology of attacks. Technical papers that proposed solutions for ransomware attacks are divided into four groups: detection, prevention, mitigation and recovery. A paper is placed in a group depending on the nature or purpose of solution it proposes. Papers that focus on detection are further subdivided into artificial intelligence (AI)-based methods and non-artificial intelligence-based approaches. AI-based approaches are then classified into machine learning methods, deep learning approaches and artificial neural networks approaches, while papers which used non-AI approaches are grouped into packet and traffic analysis categories. Data extraction involved a detailed analysis and summary of each technical paper by identifying the problem the paper addressed, its objective(s), the method/technique used, achievements of the paper in terms of the results obtained, and limitations of the study. Information synthesis was applied to identify similarities or relationships among papers in each group and, if and how a study improved upon, or addressed the limitations of another work. The reporting phase placed papers which addressed similar problems or used similar techniques in the same group, and presented their reviews in the same paragraph. This provides a good flow of communication and enhances the readability of the paper. It also provides readers with a clear understanding of the concepts discussed in the study.

3 Historical Background and Chronology of Ransomware Attacks

Ransomware was first developed in 1989, when Dr. Joseph Popp created a malware called PC Cyborg or AIDS trojan. The malware attacked systems by hiding all folders and encrypting files on the hard disk. The ransomware spread via floppy disks and attackers used a script to request victims to send $189 to a post office box in Panama in favour of PC Cyborg Corporation [6]. The infection prevented users from accessing their computers until ransom was paid and attacks were reversed. The development of strong encryption algorithms has led to the emergence of many variants of the AIDS trojan, which makes it difficult for victims to recover encrypted files without paying ransom. The worst ransomware attack occurred in 2017 with the emergence of the WannaCry Ransomware. This malware encrypts files or systems, and denies legitimate users' access to files or entire devices. A victim can access his files or system only after a ransom is paid and the attacker releases a decryption key. The Wannacry ransomware affected more than 2 million victims cutting across health, business, education and government sectors. WannaCry encrypts user data and leaves only two files consisting of the encrypted file and a file containing instructions for payment of ransom. The second file also contains a threat that hijacked data will be deleted if the victim fails to pay ransom. The ransomware opens an original file, reads its contents, creates the encrypted version and closes the file (Scaife et al., 2016). India suffered the worst WannaCry ransomware attack with Madhya Pradesh, Maharashtra and Delhi recording 32.63%, 18.84% and 8.76% of total attacks on the country respectively (eScan, 2017). High net worth corporations like FedEx, Nissan, railway companies in Germany, Russian Railways, Megafor Telefonica were also not spared. Many NHS organisations in United Kingdom were severely hit. The attack also caused serious damages to computers belonging to universities and students in China. Well-known internet service providers like RailTel and Vodafone were the most severely affected (Mohurle & Patil, 2017).

Table 1 presents a chronology of major ransomware attacks. The table provides important information on ransomware evolution based on the year a ransomware emerged, the name of the ransomware, its mode of attack, how it spreads, encryption strategy and method used by victims to pay ra`nsom.

Table 1: Chronology of major ransomware attacks

Year	Ransomware Name	Attack mode	Mode of spread	encryption strategy	Ransom payment method
1989	AIDS Trojan	Encryption of file names	Infected floppy disk	Symmetric encryption	$189 postal order
2005	Trojan PGPcoder	File encryption	Spam email attachment	Asymmetric RSA-1024 encryption	N/A
2006	Trojan Cryzip	Creates password-protected archives of infected files	Spam email attachment	Password locking	No payment; malware code includes password
	Archievus	Encryption of **My Documents** folder	Phishing emails	Asymmetric RSA-1024 encryption	Purchase of 30-digit recovery password
2007	Locker	Display of pornographic image on the machine	Phishing attack	AES and RSA	SMS text message or calling a premium-rate phone number
2008	GPcode.AK	File encryption of subdirectory	Email phishing	Asymmetric RSA-1024 encryption	$100 to $200 in e-gold or Liberty Reserve
2011	60,000 new samples	Varying attack modes	Different modes of spread	Varying encryption and locking methods	Anonymous payment services
2012	Reveton	Password stealing	Clicking malicious link	Malicious JavaScript files	Around $300
	Trojan.Randsom.C	Device locking	N/A	N/A	calling a premium-rate phone number to reactivate Windows license
2013	CyptoLocker	File encryption	Gameover ZeuS banking Trojan botnet;	public and private cryptographic keys	Two Bitcoins (or $100), CashU, Ukash,

			malicious email		Paysafecard, and MoneyPak
	Locker	File encryption	Spam campaigns	AES	$150 via Perfect Money or QIWI Visa Virtual Card number
2014	CryptoDefens e	File encryption	Spear phishing email	RSA-2048	earned $34,000 in its first month
	CryptoWall	File encryption	Infected USB drive, email, malicious executables, malicious websites	RSA-2048	more than $1,000,000
2015	LockerPin	Device locking	Adult entertainment app	AES	$500
	Linux.Encoder .1	Encryption of data and web applications files	Exploits the flaw in Magento shopping cart software	AES and RSA	Unspecified amount in bitcoin
2016	Petya	File overwriting and full hard disk encryption	MEDoc tax and accounting software	Master boot record (MBR) and file encryption	$300
	KeRanger	File encryption	Infected web link	RSA	1 bitcoin
	Xbot	File encryption and stealing online banking details	SMS messages	N/A	$100
2017	WannaCry	File and device encryption	Unknown	Hybrid (AES and RSA)	$300 in bitcoin
	Bad Rabbit	Device locking	Drive-by-download on infected websites	Locks users' devices when they click on alicious Adobe Flash installer	$280 bitcoin
2018	GandCrab	File encryption	Infected phishing email, Microsoft Office macros, VBScript and ransomware-as-a-service	Installs on a device and encrypts user files when they access infected email	$500-$600
	Katyusha	File encryption	Malware trojan encrypts and adds 'Katyusha' extension to infected files	Infects networks using EternaBlue and DoublePulsar exploits	0.5 bitcoin

	Ryuk	File encryption	Massive spam attacks and exploit kits	Symmetric AES-256 and asymmetric RSA-2048 encryption	15-50 bitcoins
2019	Prolock/ PwndLocker	File lock/encryption	Qakbot Trojan	Asymmetric RSA-2048 encryption	Bitcoin
	LockerGoga	File encryption and file wiping	Logs users out of systems, encrypts files and deactivate devices	Cryptographic encryption and deletion of infected files	N/A
	PewCrypt	File encryption	Spam email messages	Symmetric 256-bit Advanced Encryption Scheme (AES-256)	Free
	Dharma v2019	File encryption	Malicious email	Symmetric AES-256 algorithm	N/A
2020	Nefilim	File encryption	Remote desktop protocol (RDP) attack	AES-256 encryption for victim's files; RSA-2048 algorithm to encrypt the AES-256 keys	Via email communication
	Ransomware Name	Attack mode	Mode of spread	encryption strategy	Ransom payment method
	Paradise v2020	File encryption	Spam message containing internet query attachments	RSA-1024 and RSA-2048 algorithms	No ransom. Tools are available to retrieve encrypted files
	Maze	File encryption	Exploit kits such as Fallout and Spelva	RSA and ChaCha20 stream cipher	$6m - $15m
	REvil	File encryption/file blocking	Phishing email and malicious attachment	AES or Salsa20	$70m in bitcoin
	Tycoon	Password exploitation of file servers and domain controllers	Insecure connection to an RDP server and a malicious (trojanized) Java Runtime Environment	RSA	N/A
	NetWalker	Full Windows device encryption	Network-wide executable files and VBS script attachments in Corona virus phishing emails.	Salsa20	More than $30m total ransom since March 2021
2021	Dark side	File encryption and data exfiltration	VPN password	Lightweight Salsa20 with RSA-1024	75 bitcoin or $4.4m

	ReVil	File encryption/file blocking	Vulnerability in Microsoft Exchange servers	AES or Salsa20	$50m in Monero cryptocurrency demanded
	Phoenix locker	File encryption on desktop and network shares	Spam emails	RSA-2048 algorithm	$40m
	ContiLocker	File encryption and data exfiltration	Via unprotected remote desktop protocol (RDP) port	RSA-4096 and AES-256-CBC	$2.6m
	Avaddon	File encryption, data exfiltration and DDoS	Malicious JavaScript files	AES-256	$40,000 or its equivalent in bitcoin

The table shows that the development of ransomware and deployment of ransomware attacks have been on the rise since 1989 when the first known ransomware emerged. Most ransomware attack involves encryption of files and sub-directories. The devices can still function, but the infected files are inaccessible to legitimate users. A less common form of attack involves blocking users from gaining access to their devices, even if the files stored on such devices are accessible. New variants of malware have also emerged each year since 2013. This is because of the availability of sophisticated tools that enable attackers to easily craft ransomware scripts as well as huge amounts of money hackers make from ransom payment. Maze, REvil, Ryuk, Tycoon and NetWalker are currently the five most dangerous ransomware attacks (Ransomware Attacks, 2021). Several factors enhance the growth of ransomware and persistent increase in ransomware attacks. These include easy procurement of powerful encryption (symmetric and asymmetric) algorithms, which enables attackers to easily craft a ransomware tailored for a specific attack, or environment and availability of effective infection vectors such as spam email and malvertising, which ensure that a ransomware spreads rapidly to as many users as possible (Adamov & Carlson, 2017). Other factors are easy accessibility of victims to cryptocurrency for ransom payments (including the ease with which attackers can convert cryptocurrency to cash without any trace) and the availability of Ransomware as a Service (RaaS) also enables unskilled and less knowledgeable attackers obtain customize ransomware and track victims via a user interface (Gellegos-Segovia et al., 2017). The creators of RaaS earn a percentage of profits from ransomware attacks launched via their platforms.

4 Ransomware Detection

Research show that ransomware attacks are on the rise and have doubled in the first quarter of 2020 due to increase in remote working culture imposed by COVID-19 pandemic. Many individuals who work from home do not practice the same cybersecurity measures commonly imposed in the office environment. Also, most remote workers use personal devices which are not adequately equipped with security mechanisms such as antimalware packages, firewall, intrusion detection/prevention systems, password management tools and encryption software. Ransomware leverages on new vulnerabilities found in systems and networks, using attacks focus on both small, medium and big companies who imbibe the remote working culture. Apart from encrypting files and locking devices, ransomware can also use sophisticated techniques to carry out data exfiltration. This resulting exposure of sensitive information may lead to severe security concerns and privacy violations. This is addition to financial losses and reputation damage suffered by victims. Ransomware attack against a health facility may result in loss of life such as in the case of a Dusseldolf University hospital patient where an attack interrupted emergency services and the hospital management had to send the patient to another hospital 17 miles away (Fingers, 2020). The patient eventually died as a result of delay in treatment. Ransomware payment is also a means by which attackers extort several millions of dollars from innocent victims every year (Symantec Corporation, 2016) Ransomware attacks account for more than 41% of cyber insurance claims in 2020 and it is projected that total losses which have organizations suffer from ransomware attacks may hit $20 billion at the end of 2020 (Potoroaca, 2020). The money which organizations use to pay ransom can be channeled to other productive ventures resulting in the overall growth of the business. These concerns highlight the need for efficient and reliable methods for ransomware detection, prevention, mitigation and recovery. Ransomware detection methods are generally categorized into automated and manual. Automated approaches rely on the use of tools to detect and report ransomware attacks. Such tools are usually software packages which may also possess the ability to block attacks. Manual detection methods focus on regular inspection of files and devices for obvious signs of attacks. This includes checking for changes in file extensions and whether authorized users can access files and devices. That is, checking whether a malware attack has not modified files and authorized users have not been blocked from accessing their devices and files. The flow of presentation in this section is illustrated in Figure 4.

4.1 Automated Ransomware Detection

Existing approaches for ransomware detection predominantly focus on system level monitoring, for instance, by tracking the file system characteristics. Automated ransomware detection approaches can be divided into two major categories namely, artificial intelligence (AI)-based methods and non-artificial intelligence (non-AI)-based methods. AI-based methods commonly use techniques such as machine learning (ML), deep learning (DL) and artificial neural network (ANN) for ransomware detection. Some tools apply variants of these techniques or a hybrid approach using a combination of two or more techniques to address the menace of ransomware attacks. Non-AI methods use approaches such as packet inspection and traffic analysis to detect ransomware. A major strength of automated approaches is their ability to detect, block and recover from ransomware attack without human intervention. The tools also possess high level accuracy and reliability in terms of ransomware detection, prevention and recovery.

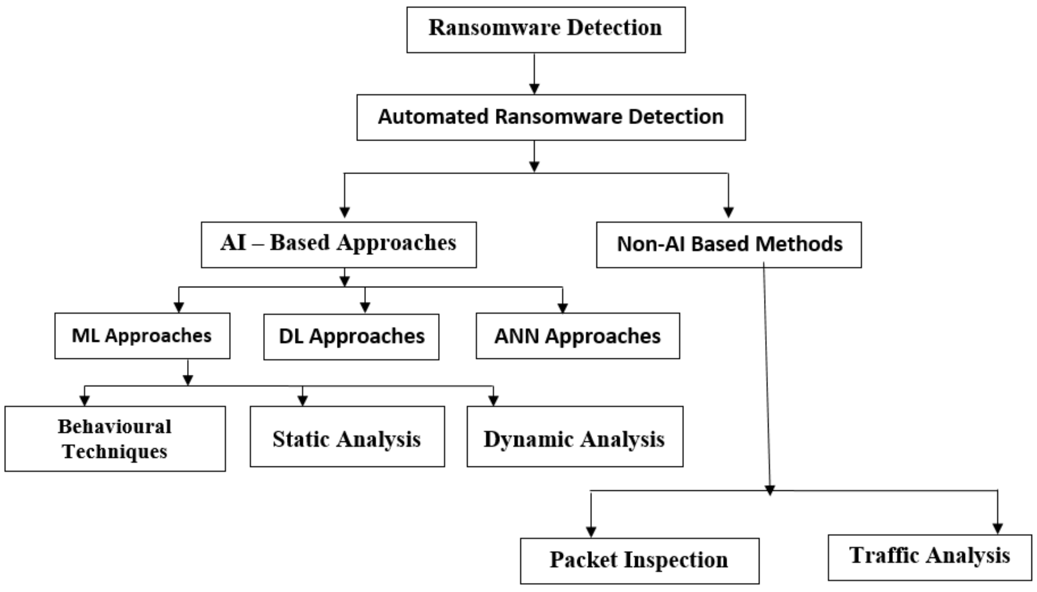

Figure 4: Flow of presentation on ransomware detection

4.1.1 Artificial Intelligence-Based Methods

Artificial intelligence-based methods use machine learning (such as behavioural techniques and static and dynamic analysis), deep learning and artificial neural network to perform automated detection of ransomware attacks.

4.1.1.1. Machine Learning Approaches

Machine learning (ML) is a branch of artificial intelligence which provides systems with the ability to learn from, and detect patterns in existing data, while making decisions with little or no human intervention (Dontov, 2019). It is a method commonly used to automate analytical model building. ML techniques enable computers to make predictions based on patterns found in large datasets. The algorithms are able to adapt to changes and make improvements as the size of the dataset increases. The ability of ML to make predictions based on file behaviour as well as known and unknown datasets makes it a viable tool for detecting previously unknown ransomware variants. However, machine learning techniques require a minimum of between 50 and 1,000 data points to make reliable prediction. Few samples may result in overfitting and biased prediction. Also, training machine learning algorithms require significant amount of time. File behaviour detection is the major application of machine learning to ransomware detection. ML algorithms use specialized analysis (such as interactive debugging or post mortem code execution analysis) to extract large amount of salient and discriminant information in order to learn the behaviour of a legitimate or normal application. ML-based ransomware detection tools perform detailed analysis of legitimate code execution and are able to identify malicious applications. Such tools make intelligent decisions and prompts specific actions by leveraging on their ability to distinguish between normal and abnormal program execution. The machine learning approaches explored in this study are behavioural techniques as well as static and dynamic analysis.

Behavioural Techniques

A normal application behaviour is measured from both user perspective and resource perspective. A normal behavioural baseline is established based on what represents normal or routine operations of a computer system or network. Such operations may include logins, file access, user and file behaviors, resource utilization, and other important signs of normal activity (Acronis International, 2021). The duration of the learning process depends on the amount of data needed to establish a baseline to represent normal system behaviour. The tool identifies and scrutinizes behavioural anomalies which do not fall within the normal behavioural pattern represented by the baseline. (Juan et al., 2017) proposed a ransomware detection and prevention model for unstructured dataset extracted from Ecuadorian control and regulatory institution (EcuCERT) logs. The approach uses machine learning techniques to detect abnormal behavioral patterns associated with Microsoft Windows-based ransomware. Feature selection was applied to the Log data to extract the most useful and discriminating information that represents a ransomware threat. The extracted information represents the feature set which serves as input for automatic learning algorithms. The algorithms use the input feature set to model abnormal behavioral patterns in order provide timely and reliable detection of ransomware. There was an attempt to address the limitations of signature-based methods in detecting ransomware attacks which evolve daily due to availability of code obfuscation techniques and creation of new polymorphic variants (Shaukat & Ribeiro, 2018). This is necessary because generic malware attack vectors do not adequately capture the specific behavioral patterns of cryptographic ransomware and as such, not sufficient or reliable enough for ransomware detection. The proposed approach known as RansomWall is a layered and hybrid mechanism based on the application of static and dynamic analyses to generate a new set of features that model ransomware behavior. The approach uses a strong trap layer for early detection of ransomware and is suitable for detecting zero-day attacks. An evaluation of RansomWall and Gradient Tree Boosting Algorithm on 574 samples of 12 Microsoft Windows operating system-based cryptographic ransomware produced 98.25% detection rate and very low (almost zero) false positives. It is also able to detect 30 zero-day attack samples, with less than 10% detection rate compared to 60 VirusTotal security engines. CryptoDrop was developed to provide early detection of ransomware based on suspicious file activity (Scaife et al., 2016). It uses a set of behavioral features to terminate any process that alters a large amount of the user's data. CryptoDrop can integrate common ransomware features to support rapid detection with low false positives. Experimental analysis shows that CryptoDrop is an efficient tool for ransomware detection and prevention. It is able to prevent execution of ransomware files with a median loss of only 10 files out of almost 5,100 tested files. Overall, the approach leverages on behavioral analysis to minimize data loss due to ransomware attacks. A limitation of CryptoDrop is its inability to determine the intent of attack indicated by changes in file behaviour. An example is a situation where the tool cannot determine whether a set of documents is encrypted by the user or ransomware. The system simply notifies the user who decides whether a suspicious activity is desirable or not. CryptoDrop flags legitimate activities such as compression whose behavior is normal, expected, desirable, and not actually invasive. It is necessary for future versions to possess the ability to distinguish legitimate bulk transformation activities such as file compression from malicious attacks.

Table 2 presents a summary of previous studies on behavioural techniques for ransomware detection.

Table 2: Summary of related works (behavioural techniques)

Author	**Problem addressed**	**Method used**	**Result**	**Limitation**
Shaukat & Ribeiro (2018)	Ransomware detection	Layered and hybrid mechanism (RansomWall)	Suitable for detecting zero-day attacks	N/A
Scaife et al. (2016)	Ransomware detection	Evaluation of RansomWall and Gradient Tree Boosting Algorithm (CryptoDrop)	Median loss of only 10 files out of almost 5,100 tested files	Inability to determine the intent of attack indicated by changes in file behavior

Makinde et al. (2019)	To detect the susceptibility of a real network system to ransomware attack	Machine Learning	Correlation above 0.8	It simulated the behaviour of few users
Ahmad et al. (2019)	To distinguish members of the Locky ransomware	Behavioural ransomware detection approach (parallel classifiers)	Highly accurate detection with low false positive rate	N/A
Zahra & Sha (2019)	Detecting Cryptowall ransomware attack	Command and control (C&C) server black listing	Extracts TCP/IP header from web proxy server which serves as the gateway to TCP/IP traffic.	The model was not implemented to demonstrate its accuracy and effectiveness in detecting ransomware and their modes of attack against different operating system environments
Singh et al., (2022)	Detection of previously unknown ransomware families and classification of new ransomware attacks	Examines access privileges in process memory to achieve easy and accurate detection of ransomware	accuracy ranges between 81.38% and 96.28%.	N/A

A variant of behavioural detection approaches used a machine learning baseline model for simulating and predicting the individual network user behaviour pattern at the micro level in order to detect possible scenarios that may indicate a vulnerability or an actual ransomware attack (Makinde et al., 2019). The goal was to detect the susceptibility of a real network system to ransomware attack. A comparative evaluation of the results obtained from the simulated network and the log data obtained from the server in the real-life network system indicates a realistic model with a correlation above 0.8. A limitation of this approach is that it simulated the behaviour of few users. Future works should focus on using tools for big data analytics to simulate the behaviour of a large number of users. A more recent behavioural ransomware detection approach used two parallel classifiers to distinguish members of the Locky ransomware family according to their types (Ahmad et al., 2019). The method focused on early detection based on behavioural analysis of ransomware network traffic in order to prevent a ransomware from connecting to command-and-control servers and executing harmful payloads. The study used a dedicated network to collect network information and extract relevant features of network traffic. The extracted features of the Locky ransomware family are processed by two independent (parallel) classifiers working on data at packet and datagram levels. Experimental results show that the method is able to extract valid features and provides a high level of effectiveness in tracking the activities of ransomware on the network. It also offers highly accurate detection with low false positive rate. Zahra and Sha (2019) proposed a domain-specific framework for detecting Cryptowall ransomware attack based on the communication and behavioral analysis of the ransomware in an IoT environment using command and control (C&C) server black listing to detect ransomware attacks. The method extracts TCP/IP header from web proxy server which serves as the gateway to TCP/IP traffic. It also extracts source and destination IP addresses and compares them with blacklisted IP of Command-and-Control servers. A ransomware is detected if the source or destination IP matches ransomware attack for IoT devices. However, the model was not implemented to demonstrate its accuracy and effectiveness in detecting ransomware and their modes of attack against different operating system environments. A very recent approach to behavioural-based detection leverages on access privileges in process memory to achieve easy and accurate detection of ransomware (Singh et al., 2022). The method can also detect previously unknown ransomware families and classify new ransomware attacks using the access privileges a file or an application possesses and the area of memory it intends to access. The helps to identify the behaviour of an executable, and detect its intent before it causes serious damage to legitimate files and applications. Experimental results based on these multiple algorithms produced good detection accuracy which ranges between 81.38% and 96.28%.

Static and Dynamic Analysis

A novel detection technique based on static analysis extracts features directly from raw ransomware binaries using frequent pattern mining (Khammas, 2020). It also uses Gain Ratio technique to select 1000 features for optimal ransomware detection. Random forest classifier was used to analyze the impact of trees seed numbers on the detection process. Experimental results show that the detection rate of proposed approach is 97.74%. Direct extraction of raw ransomware binaries results in a remarkable increase in the speed of detection. An enhanced approach to ransomware detection integrates dynamic analysis with machine learning (Hwang et al., 2020). It is a hybrid ransomware detection model based on Markov model and Random Forest model. The approach uses Windows API call sequence pattern to build a Markov model which extracts the unique features of ransomware. This is followed by using Random Forest to model the remaining data in order minimize error rates. The two-stage mixed detection technique achieved good detection rates with an overall accuracy of 97.3%, 4.8% FPR (false positive rate) and 1.5% FNR (false negative rate). A similar approach known as *EldeRan* uses dynamic analysis to detect ransomware at run-time (Sgandurra et al., 2016). The technique leverages on the fact that run-time features exhibited by ransomware samples are similar for all ransomware families. *EldeRan* performs dynamic analysis and ransomware detection by monitoring the actions carried out by applications when they are first installed and checking for obvious signs of ransomware. The result of experiments carried out on a dataset of 582 ransomware and 942 goodware applications, shows that the approach achieves an area under the ROC curve of 0.995. A major strength of the *EldeRan* lies in its ability to perform dynamic analysis and ransomware detection even if the entire dataset of a ransomware family is not available. This supports early detection of new ransomware variants.

An improved technique for ransomware detection used an integrated approach, which combines static and dynamic analysis (Bazrafshan et al., 2013). It is an analysis framework based on support vector machines, which uses “run-time” and “static code” features for early detection of known and previously unknown ransomware variants. The results of experiments based on a wide array of ransomware types suggest that the integrated approach provides better ransomware detection than using either static analysis or dynamic analysis individually. The integration of static and dynamic analysis has also been used to analyze ransomware threats against mobile devices and perform mobile ransomware detection (Yang et al., 2015). The proposed approach combines the results of static and dynamic analysis for detecting ransomware threats and attacks against mobile applications. It is a two-phase approach which integrates data states and software execution on the critical test path of the Android API. The first phase is static analysis which detects the likelihood of an attack by using API, existing attack patterns and dynamic analysis to execute a program in a limited and restricted scope and comparing whether the detected path conforms with existing attack patterns. The second phase (which is runtime dynamic analysis) uses dynamic inspection to detect the nature of attack and possible violation of data confidentiality (such as web browser cookie) without compromising sensitive and secured data sources in mobile device. A related work detects unknown ransomware by using the most discriminating API calls to train a classifier (Sheen & Yadav, 2018). The approach was applied on an imbalanced dataset consisting of unequal amounts of ransomware and benign data. Experimental results show that the approach is more suitable for random forest than decision tree or KNN. Random forest produced the best detection rate of over 98% because it is more robust against class imbalance than decision tree and KNN. A limitation of this study is class imbalance in the dataset due to the difference in the number of samples in the ransomware class and benign class. A future work should apply the same technique on a balanced dataset using the same classifiers and observe the outcome. An improved approach integrates feature generation engines and machine learning for analyzing malware samples obtained from raw binaries, assembly codes, libraries, and function calls in order to identify the goal malicious codes intend to achieve. Poudyal et al. (2018) applied different supervised ML techniques on features extracted from ransomware and benign binaries. Performance evaluation results show that the approach has detection accuracy which ranges from 76% to 97% depending on the ML classifier used. Seven out of the eight classifiers achieved a detection rate of at least 90%. The study also revealed better ransomware detection rates when static level analysis is applied to data obtained by integrating ASM-level and DLL-level features. Similarly, Dehghantanha et al. (2018) proposed a Decision Tree (J48) classifier known as NetConverse, for high speed and reliable detection of Windows ransomware. Experimental results based on conversation-based network traffic features dataset show a true positive detection rate of 97.1% using the Decision Tree (J48) classifier. Static and dynamic techniques can also be used for real time detection and prevention of ransomware attack (Lalson et al., 2019). The technique offers a robust and an effective protection against a variety of ransomware. The approach halts attacks before the system or network experiences a significant damage. However, the proposed method cannot perform the recovery of infected files. It is also possible for a ransomware to encrypt some files before it is actually detected or blocked. Lee et al. (2022) addressed the ineffectiveness of static analysis against obfuscating ransomware, which hides their behaviour to evade detection and low-speed detection of dynamic analysis by proposing a statistical analysis which uses heuristics to distinguish between normal files and those attacked by ransomware. The approach provides real-time detection of known crypto-ransomware variants. It is also efficient with about 13% overhead required during the detection process.

Recent ML approaches such as the one proposed by Rani and Dhavale (2022) used a number of machine learning models such as decision tree, random forest, KNN, SVM, XGBoost and Logistic Regression to build an effective proof of concept for a product specific ransomware. The proposed solution is efficient and reliable with an accuracy of 98.21%. Similarly, three different machine learning algorithms namely decision tree (J48), random forest and radial basis function (RBF) were applied on 1000 dominant features obtained from raw, byte-level ransomware data using the gain ratio feature selection method (Khammas, 2022). The results from experiments show that random forest is the most effective of the threes algorithms with ~ 98% accuracy and the most suitable feature size is 1000 attributes. An enhanced approach integrates ensemble learning with voting-based method, monitors memory usage, system call logs, CPU usage and performs static and dynamic analysis of text, permissions and network-based features (Ahmed et al., 2022). Experimental results based on malicious and benign features (static and dynamic) obtained from Android malware applications show that the proposed technique can detect unknown ransomware attacks based on the behaviour of malicious applications. The technique is also robust against adversarial evasion attacks as demonstrated by its high detection accuracy when tested with 1-bit, 10-bit, 20-bit, 30-bit and 40-bit crafted ransomware data. Talabani and Abdulhadi (2022) proposed two rule-based models to address the low accuracy of ransomware detection tools which use data mining and machine learning techniques. The models known as Partial Decision Tree (PART) and Decision Table were applied to bitcoin dataset consisting of 61,004 samples of 29 ransomware families with ten descriptive and decision attributes. Experimental results show that the PART algorithm provides better performance in terms of accuracy (96.01%), recall (96%), precision (95.9%) and F-Measure (95.6%) than Decision Table. Experimental results show that it is necessary to carry out additional investigation on the application of PART to predictive modelling tasks in ransomware detection experiments.

A summary of previous studies which used static and dynamic analysis for ransomware detection is presented in Table 3.

Table 3: Summary of related works (static and dynamic analysis)

Author	**Problem addressed**	**Method used**	**Result**
Khammas (2020)	Ransomware detection	Random forest technique	Detection rate is 97.74%.
Hwang et al. (2020)	An enhanced approach to ransomware detection.	Markov model and random forest model	Overall accuracy of 97.3%, 4.8% FPR (false positive rate) and 1.5% FNR (false negative rate
Dehghantanha et al. (2018)	High speed and reliable detection of windows ransomware	Netconverse (decision tree (j48) classifier)	True positive detection rate of 97.1%
Rahman & Hasan (2019)	Improved technique for ransomware detection	Analysis framework based on support vector machines	Integrated approach provides better ransomware detection than using either static analysis or dynamic analysis individually.
Jasmin (2019)	Distinguishing ransomware traffic from normal traffic	Random forest, support vector machine and logistic regression algorithms	Random forest has the best detection rate of 99.9% and a false positive rate of 0%.
Ameer (2019)	Ransomware detection	Static and dynamic analysis	Detection and classification accuracy of 100%

Talabani & Abdulhadi (2022)	Low accuracy of ransomware detection tools which use data mining and machine learning techniques	Partial Decision Tree (PART) and Decision Table	accuracy (96.01%), recall (96%), precision (95.9%) and F-Measure (95.6%)

Several enhanced machine learning techniques have been proposed for effective and reliable detection of ransomware. These techniques are meant to address the weaknesses in existing ML-based ransomware detection methods. One of such improvements addressed the limitation of detection techniques (such as sandbox analysis and pipelines) due to their inability to isolate a sample and handle the delay in analyzing isolated ransomware samples (Adamu, 2019). The approach predicts ransomware using a dataset consisting of 30,000 attributes which serve as independent variables. Feature selection was used to obtain five attributes used as input to support vector machine algorithm. The method has promising ransomware detection rate with accuracy of 88.2%. Another improvement focused on detecting ransomware in cloud storage instead of the local system (Matthias, 2018). It is a hybrid technique which integrates 'guilt by association' assumption with content-based, metadata-based and behaviour-based analysis to minimize the false positive rate. This involves the use of file versioning of the cloud storage to delay the recovery and transferring the supervision of the recovery to the end user. The only responsibility of the end-user is to supervise the recovery. Users are provided with classification information which allows them make informed decisions and prevent false positives. The approach provides improved detection accuracy and reliable recovery. A novel approach used network connection information, certificate information and machine learning for network-level ransomware detection (Jasmin, 2019). The method can be used in conjunction with system-level detection to provide early detection of ransomware attacks. The technique extracts and models ransomware features based on three major characteristics of network traffic namely, connection-based, encryption-based, and certificate. It is a feature model which used random forest, support vector machine and logistic regression algorithms to distinguish ransomware traffic from normal traffic. Experimental results based on a variety of datasets showed that random forest has the best detection rate of 99.9% and a false positive rate of 0%. Another enhanced detection approach is a decision tree model based on big data technology, which exploits Argus for packet preprocessing, merging, and labeling malware file (Wan et al., 2018). Biflow was used to replace the packet data and reduce the data size by a factor of 1000 (that is, 1000:1). Feature selection and feature concatenation were employed to extract and combine the characteristics of a complete network traffic. The method used six feature selection algorithms in order to achieve better classification accuracy. A recent and an innovative ransomware detection method used machine learning to monitor power consumption of Android devices (Azmoodeh et al., 2018). The proposed approach distinguishes ransomware from benign applications by monitoring the energy consumption patterns of various Android processes. This is achieved by collecting and analyzing the unique local fingerprint of ransomware's energy consumption. Experimental results show that the method achieved high detection and precision rates of 95.65% and 89.19% respectively. It also has better accuracy, recall rate, precision rate and F-measure than K-Nearest Neighbor, Neural Network, Support Vector Machine and Random Forest. Another enhanced solution is a novel lightweight approach known as RanDroid for automated detection of polymorphic ransomware (Alzahrani et al., 2018). The technique detects new ransomware variants on Android platforms using the structural similarity measures between features extracted from an application and a set of threat data extracted from known ransomware variants. The similarity measures used are Image Similarity Measurement (ISM) and String Similarity Measurement (SSM). Further information was extracted by applying linguistic analysis on the app's code behavioural features and image textural strings. The approach addressed the limitations of static analysis by performing dynamic and static analyses in order to mitigate ransomware attacks without modifying the Android OS and its underlying security module. An evaluation of RanDroid based on 950 ransomware samples showed that the approach can detect ransomware based on evasive techniques such as sophisticated codes or dynamic payloads. A related work proposed a hybrid solution based on the integration of static and dynamic analysis for detecting Android ransomware and distinguishing ransomware from other malware (Ameer, 2019). The approach applied static analysis on permissions, text, and network-based features. It also applied dynamic analysis on the memory usage, system call logs, and CPU usage. The results of experiments based on features extracted from ransomware and benign samples show that technique can mitigate evasive ransomware attacks. It is also able to detect and classify unknown ransomware with 100% accuracy.

4.1.1.2 Deep Learning Approaches

Deep learning techniques are aimed at addressing the shortcomings of conventional supervised ransomware detection tools. The goal is to enhance the accuracy and reliability of results obtained from a ransomware detection activity. Deep learning techniques perform automatic feature generation and are very suitable for unstructured datasets. The techniques also require very little or no human intervention (good self-learning capabilities). Deep learning algorithms are very suitable for classifying audio, text and image data. This enhances their effectiveness at detecting textual and image ransomware data. However, training deep learning algorithms requires a very large amount of data. This makes the algorithms unsuitable for general purpose applications especially those requiring small data points or sizes. Other limitations of deep learning include the need for high processing (CPU) power and inability to easily adapt to real life datasets. A recent application of this approach is a deep learning based semi-supervised framework, which extracts inherent, unlabeled and previously unknown features of new ransomware variants (Sharmeen et al., 2020). The framework also provides an adaptive detection model by integrating the unsupervised learned model with supervised classification. Experimental results based on real ransomware data with a dynamic analysis testbed shows that the method is highly accurate at detecting different kinds of ransomware compared to existing supervised approaches. Another deep learning approach for automated behavioural-based ransomware detection applied dynamic analysis on data obtained from Application Programming Interface (API) calls made by the executable (Maniath et al., 2017). This approach uses a word sequence to represent the list of API calls made by an executable file. It applied Long-Short Term Memory (LSTM) networks for binary sequence classification of application programming interface (API) calls a suitable method for detecting ransomware behavior. The approach detects ransomware behaviour using API calls obtained from systems logs of modified sandbox environment. It is a suitable method for reliable analysis and detection of large malwares samples. A related study proposed a deep learning technique based on features extracted from permissions and API calls for detecting Android ransomware (Wongsupa, 2018). AndroGuard (a python library) was used for feature extraction, while the ransomware detection framework was implemented on Keras, using multilayer perceptron (MLP) with back-propagation and supervised learning algorithm. The results of experiments on real-world applications show that the accuracy is 98% for MLPs with more than 3 hidden layers and moderately sized neurons. However, the use of MLPs with two hidden layers and large number of neurons results in low detection accuracy of between 45% and 60%. A novel deep learning approach to ransomware detection extracts salient behavioral features from labeled ransomware data (Aragom et al., 2016). It is a novel architecture which combines deep packet inspection with machine learning. The model can detect and prevent various types cryptographic ransomware. Experimental results show that the deep learning model achieved a detection accuracy of 93.92%, which makes it suitable for timely detection of unknown ransomware in high-speed network. Table 4 is the summary of related works which used deep learning techniques to implement automated ransomware detection systems.

Table 4: Summary of related works (deep learning approaches)

Author	**Problem addressed**	**Method used**	**Result**	**Limitation**
Sharmeen et al. (2020)	To enhance the accuracy and reliability of results obtained from a ransomware detection activity	Deep learning based semi-supervised framework	Highly accurate at detecting different kinds of ransomware compared to existing supervised approaches	N/A
Maniath et al. (2017)	Automated behavioural-based ransomware detection	Deep learning techniques	The approach detects ransomware behaviour using API calls obtained from systems logs of modified sandbox environment	N/A
Wongsupa (2018)	Detecting Android ransomware.	Deep learning technique and Supervised learning algorithm	The results of experiments on real-world applications show that the accuracy is 98% for mlps with more than 3 hidden layers and moderately sized neurons	The use of mlps with 2 hidden layers and large number of neurons results in low detection accuracy of between 45% and 60%.
Aragom et al. (2016)	Detection and prevention of various types cryptographic ransomware.	Combines deep packet inspection with machine learning	Detection accuracy of 93.92%,	N/A

Vinayakumar et al. (2017)	Effective detection and classification of ransomware	Enhanced deep learning technique	Classification accuracy of 0.98 (98%)	The experimental results do not represent the actual situation involving more complex architecture settings
Olani et al. (2022)	Detection of ransomware by monitoring and analyzing changes in the distribution hardware performance counter data.	Deep learning	Classification accuracy of 98.6% and recall score of 84.41%.	N/A

An enhanced deep learning technique applied shallow and deep networks on features extracted from API calls for effective detection and classification of ransomware (Vinayakumar et al., 2017). The study explored a number of network parameters and structures to obtain the best architecture for the multi-layer perceptron (MLP). This involved up to 500 epochs with a learning rate between 0.01 and 0.5. The results of various experiments showed that MLP has very high accuracy of 1.0 (100%) in distinguishing ransomware from benign samples. It was also able to classify ransomware into their families with an accuracy of 0.98 (98%). This shows that the approach can detect and classify ransomware better than other classical machine learning techniques. However, the proposed approach is a very simple MLP network, which does not impose high computational burden on hardware and monolithic training environment. The experimental results do not represent the actual situation involving more complex architecture settings. A future work should focus on using more complex MLP network to perform the same experiments on state-of-the-art hardware in a distributed environment. A recent a deep learning model monitors changes in the distribution hardware performance counter data across the system and analyzes relevant information to achieve effective and efficient detection of ransomware (Olani et al., 2022). The information extracted is specifically related to events which indicate behaviour that distinguishes a ransomware from a benign application. The results of experiment based on different ransomware families show that the model is effective with ransomware classification accuracy of 98.6% and recall score of 84.41%. The model is also effective for detecting zero-day attack as demonstrated by experiments based on previously unknown CoronaVirus, Ryuk, and Dharma ransomware variants.

4.1.1.3 Artificial Neural Network Approaches

Neural networks have wide applications which makes them suitable for detecting different types of ransomware data (text or image) and ransomware variants. The ability of neural networks to perform continuous learning makes them suitable for adapting to new ransomware data and detecting zero-day ransomware attacks. However, neural network techniques are hardware dependent and susceptible to data dependency. They also deny human analysts from tracking data processing tasks and checking for deviations (black box nature). Agrawal (2019) proposed an enhanced technique which leverages on the ability of recurrent neural networks to establish a relationship among events which follow a particular sequence. The technique known as Attended Recent Inputs-Long Short-Term Memory (ARI-LSTM) uses attention mechanisms to extract the pattern of events created by ransomware sequences. The approach leverages on the ability of recurrent neural networks to provide high detection accuracy for sequence learning models. An LSTM is a type of recurrent neural network which possesses the ability to establish a relationship among a sequence of events caused by ransomware attack (Shmidhuber & Cummins, 1997; Gers, 2000). ARI enhances neural cells by incorporating attention in learning from ransomware sequences. It uses the concept of a subsequence to extract local event patterns in ransomware sequences to learn from a recent history of ransomware. An evaluation of ARI-LSTM using ransomware and benign executables captured from Windows operating system showed that the technique has better detection rate than LSTM. A much finer scale evaluation of detection accuracy showed that the technique has a False Positive Rate (FPR) set of 2%. Generally, ARI-LSTM possesses much better performance accuracy (or detection rate) of 91% with low values of FPR thus establishing the potency and efficiency of attention mechanisms in learning local patterns. Similarly, identification of important and unique features of ransomware can be used to detect an attack (Arslan, 2020). This is achieved by using transfer learning based deep convolutional neural networks to perform feature engineering in order to analyze important properties and behaviors of a ransomware. The technique leverages on the ability of neural networks to detect some attributes, states, and patterns of ransomware files. Feature engineering and analysis were performed on static and dynamic datasets consisting of 3646 samples (1700 Ransomware and 1946 Goodware) and 3444 (1455 Ransomware and 1989 Goodware) samples respectively. Experimental results show that relevant features for ransomware detection are registry changes, application programming interface (API) calls, and dynamic link libraries (DLLs). It was also observed that N Gram technique can be used to detect important sequences in a ransomware attack. For example, a Registry Delete operation, whereby a malicious file attempts to delete registries, follows a particular and repeated sequence. A different observation involving benign files showed that Registry Delete operation does not follow any particular or repeated sequence. A reliable and efficient ransomware detection leverages on the nonexistence of a common Registry deleted sequence used by both malicious and benign files. Table 5 summarizes previous researches which proposed artificial neural network approaches for ransomware detection.

Table 5: Summary of related works (artificial neural network approaches)

Author	**Problem addressed**	**Method used**	**Result**
Agrawal (2019)	Establishing a relationship among events which follow a particular sequence	Attended Recent Inputs-Long Short-Term Memory (ARI-LSTM)	High detection accuracy for sequence learning models
Schmidhubar & Cummins, (1997; Gers (2000)	Establishing a relationship among events which follow a particular sequence	Attended Recent Inputs-Long Short-Term Memory (ARI-LSTM)	Have better detection rate than LSTM. False Positive Rate (FPR) set of 2%.
Arslan (2020)	Using unique features of ransomware to detect an attack	Transfer learning based deep convolutional neural networks	Detects some attributes, states, and patterns of ransomware files.

4.1.2 Non-Artificial Intelligence-Based Methods

Non-AI methods use approaches such as packet inspection and traffic analysis to detect ransomware. One of such methods aims at detecting ransomware using a network of decoy and bogus computer systems known as honeypot. The goal was to create and monitor honeypot folder for changes that could be used to detect the presence of ransomware (Moore, 2016). The study performed the manipulation of the Windows Security logs using the File Screening service of the *Microsoft File Server Resource Manager* feature and *EventSentry*. Although honeypot is a useful tool for tracking network activity, the method offers a limited view of ransomware and their activities on the network as the absence of attack alerts does not mean that a honeypot is not a target of ransomware attack. A related work proposed an algorithm that probes networks for passive monitoring of traffic in order to detect the presence of ransomware (Morato et al., 2018). Experimental analysis using 19 different ransomware families show that it takes the proposed algorithm less than 20 seconds to detect the presence of ransomware. It was also observed that not more than 10 files are lost within the 20 second duration. The method allows recovery of lost files as their contents were stored in the network traffic. It also has low false positives based on experiments conducted on traffic data from real-life corporate networks. A very recent neural network approach to ransomware detection is the novel Bayesian Neural network known as the Radial Spike and Slab Bayesian Neural Network (Nazarovs et al., 2022). The proposed solution is suitable for large and/or complex architectures as it provides better performance than the generic Bayesian Neural Network and other deep learning techniques. It also provides enough information to trigger the suspicion of investigators and confirm whether an incident is actually a ransomware attack or not. Overall, the technique helps to overcome the limitation of insufficient ransomware datasets for deep learning experiments by eliminating the likelihood of overfitting even if small-sized samples are used for training and classification. A limitation of the approach is the need for human intervention to disable systems and prevent network access in the event of a suspected ransomware attack. A summary of previous studies based on non-AI techniques is presented in Table 6.

Table 6: Summary of related works (non-artificial neural network approaches)

Author	**Problem addressed**	**Method used**	**Result**	**Limitation**
Moore (2016)	Ransomware Detection	Honeypot	N/A	Method offers a limited view of ransomware and their activities on the network
Morato et al. (2018)	Detecting the presence of ransomware and preventing attacks	N/A	Less than 20 seconds to detect the presence of ransomware.	N/A
Cabaj et al. (2017)	Software-Defined Network (SDN) environment	Rapid response to ransomware threats	Detection rates of between 97% and 98% as well as 4–5% false alarm rates	N/A
Chen et al. (2018)	Systematic characterization and real-time detection of Android ransomware	Novel technique for real time detection of encrypting ransomware	Abnormal encryption activities can be detected before a ransomware causes significant damages. The analysis of runtime performance also demonstrated the usability of ransomprober	Attempt at detecting mobile ransomware is constrained by the unavailability of a comprehensive dataset and limited understanding of real-time ransomware attack.
(Kharra zet al., 2015)	HELDROID	Distinguish known and unknown scareware and ransomware samples from goodware	Provides reliable protection against many zero-day ransomware attacks	N/A

A slightly different detection method used the modes of ransomware communication in a Software-Defined Network (SDN) environment to provide a rapid response to ransomware threats (Cabaj et al., 2017). The proposed method observes the network communication patterns of CryptoWall and Locky ransomware families between an infected host and an attacker's command and control server. Threat detection involves an analysis of the HTTP message sequences and the sizes of their respective contents. The results of experiments based on actual ransomware data showed high detection rates of between 97% and 98% as well as 4–5% false alarm rates. This shows that the approach is simple, realist and effective in preventing ransomware attacks. Chen et al., (2018) proposed a novel real-time detection system called RansomProber, which analyzes the user interface widgets of related activities and the coordinates of users' finger movements. The technique is suitable for a systematic characterization and real-time detection of Android ransomware. The results of the analysis of these samples from different perspectives revealed details such as the ransomware scale, classification, and features. The study also designed a novel technique for real time detection of encrypting ransomware. The goal is to monitor a device's sensitive files and determine the user's intention. The technique can accurately and reliably detect whether a file encryption activity initiated by users or ransomware. Experimental evaluation showed that proposed method can detect abnormal encryption activities before a ransomware causes significant damages. The analysis of runtime performance also demonstrated the usability of RansomProber. However, attempt at detecting mobile ransomware is constrained by the unavailability of a comprehensive dataset and limited understanding of real-time ransomware attack. A related work (Kharraz et al., 2015) proposed a mobile ransomware detection approach known as HELDROID to distinguish known and unknown scareware and ransomware samples from goodware in a quick, efficient and fully automated manner. The approach monitors abnormal file system behaviour to offer protection against a large number of ransomware. It also provides reliable protection against many zero-day ransomware attacks by examining I/O requests and protecting master file table in the NTFS file system. A very recent non-AI technique addressed the limitations of entropy-based ransomware detection such as misclassification due to high-level entropy of some legitimate files and impracticality of a wholesome evaluation of large files to detect ransomware due to high cost of such effort (Kim et al., 2022). This was achieved by proposing EntropySA and DistSA as byte-frequency-based indicators which explore the properties of “sample areas" (SAs) of suspicious files. The discriminant feature used to distinguish a benign file from an infected file is the degree of randomness of information in the sampled sub-area of the files. An experimental evaluation of the proposed method showed that benign files whose sampled area includes information such as file header have relatively low degree of randomness despite the high level of randomness exhibited by the entire file. The main advantage of the approach is its ability to detect a ransomware based on each file it attacks. This makes the technique able to provide effective and accurate detection of ransomware attacks irrespective of the order in which a ransomware attacks files in the system. It is also robust against obfuscating ransomware which hide their behaviour to evade detection. However, the approach is unable to record 100% detection of files attacked by the DMA Locker2 ransomware because the ransomware places a unique signature string at the beginning of a file in order to evade detection. It is also unable to detect smaller (less than 256 bytes) files.

5 Prevention, Mitigation and Recovery Strategies

It is not only necessary to detect a ransomware attack after it has caused significant damages to data and systems, but also important to put strategies in place to prevent attacks from occurring. This makes it critical to devise approaches for preventing the occurrence of ransomware attacks and mitigating potential damages caused by ransomware. It is also important to ensure recovery of files and systems after attacks without any need to pay ransom. One of such methods focuses on preventing ransomware and protecting computer systems by identifying and blocking an attack (Patel & Tailor, 2020). The strategy involves fooling an attacker to encrypt a large dummy file over a long period of time. This provides sufficient time to render the remaining contents of the file system inaccessible to the ransomware. Performance evaluation of the proposed technique in a real-time environment showed that the approach is effective against ransomware attacks. A similar study used the behaviour of a system under advanced Petya ransomware attack to propose strategies for minimizing the susceptibility of systems and organizations to ransomware attacks (Aidan et al., 2018). The approach prevents Petya ransomware attack by blocking the server message block (SMB) ports (that is, UDP port 137, 138 and TCP 139, 445) or disabling SMBv1. Additional measure includes preventing the execution of perfc.dat and psexec.dat files from sysinternals. Perfc.dat and psexec.dat files are created as a result of ransomware attack. It is possible to prevent the creation of the ransomware files by self-creating perfc.dat and psexec.dat files and changing their access permissions to READONLY. Other mitigation strategies include using Software Restriction Policies (SRP) to disable binaries from executing %APPDATA%, %PROGRAMDATA% and %TEMP% paths, as well as restricting malicious files by deploying email and web filtering on the network. File- and behavior-based detection methods do not have the ability to detect or prevent previously unknown ransomware variants and ransomware which attack cloud-based data storage. This challenge was addressed by proposing a machine learning technique for ransomware prevention known as file entropy analysis (Lee et al., 2019). The method can retrieve infected files that have been synchronized to the backup server whether or not the host system is infected by ransomware. Similarly, Du et al. (2022) presented a number of defensive strategies which are able to detect a ransomware before it actually attacks an endpoint system. One of such is a hybrid machine learning solution based on intelligent KNN and density-based algorithms. The approach integrates data pre-processing and feature engineering techniques with KNN algorithm. It has high ransomware attack prediction accuracy of 98%, which makes it a suitable anti-malware and anti-ransomware solution. Another method used in the study is random forest which records a good accuracy of 99%. The study proposed K-means and DBSCAN clustering algorithms to provide effective detection of previously unknown ransomware variants. A very recent preventive solution is the system-architecture-based risk transference which relocates sensitive data from the system to highly protected storage locations (Sreejith Gopinath & Aspen Olmstead, 2022). This minimizes the susceptibility of such data to ransomware attacks. The information is also stored in a context-free manner in order to discourage attackers from attempting to hold such data hostage. Experimental results show that the proposed architecture allows for easy recovery of a system under ransomware attack.

The method proposed by Gómez-Hernández et al. (2022) supports early reaction to ransomware incidents and reduces damage to files during an attack. It is an enhanced tool which deploys a large number of “honey files” in close proximity to sensitive system and application files in order to achieve early detection of and timely response to ransomware attacks. The capability of the tool was extended by adapting it to Windows platforms and improving the system-wide management of the “honey files” to provide adequate protection of system files. Additional enhancements include semi-automation of defence mechanisms against ransomware using dynamic white-/black-lists, which minimizes the need for human intervention in the event of an attack. WmRmR (weighted minimum Redundancy maximum Relevance) is a mitigation strategy used to estimate the importance of dominant or most discriminating features in data captured at the onset of ransomware attacks (Ahmed et al., 2022). It is a hybrid solution based on the integration of two different techniques namely, enhanced minimum redundancy maximum relevance (EmRmR) and Term Frequency-Inverse Document Frequency (TF-IDF). The approach uses TF-IDF to evaluate weights generated by EmRmR algorithm and eliminate noisy features that may impair performance. The results of experiments show that the proposed solution has simple implementation, low false positive rate and is effective for early detection of ransomware attacks.

A simple technique for easy recovery from ransomware attacks irrespective of the availability of attacker's tools on the victim system to prevent recovery from such attacks has also been proposed (Wecksten, 2016). The study revealed that common cypto ransomware attack involves the installation of tools on a victim's device to make recovery from ransomware attack a herculean task. Hence the need for a technique to provide easy recovery from ransomware infections by renaming the system tool which handles shadow copies of files. A similar strategy for ransomware recovery proposed by Kim et al. (2022) enables a partial (95%) recovery of the master key used by attackers to launch Hive ransomware. This was achieved by analyzing the encryption process used by Hive ransomware and discovering its vulnerabilities. The result of this effort is the creation of a decryption key for recovering files held by the ransomware without the need to obtain the attacker's RSA private key or pay a ransom to the attacker. A very recent recovery method is the novel framework proposed as an efficient technique for recovering XML documents that have been compromised by ransomware attack (Al-Dwairi et al., 2022). The approach uses the concepts of links to support the distributed storage of different versions of the same file. Adequate access control is also put in place to prevent the file versions from unauthorized encryption or deletion. Experimental results show that the time required for decrypting an encrypted XML file is directly proportional to the actual size of the file before encryption. Generally, files that less than 1 MB requires less than 120 ms and decryption of bzip2 encrypted files required the highest CPU utilization. Decrypting zip and gzip encrypted files requires almost the same amount of memory (~ 6.8 KBs), while decryption of bzip2 encrypted files increases the memory usage to 28 KBs. Overall, the approach is efficient in terms of storage overhead, processing time, CPU utilization, and memory usage.

6 Future Research Directions

Path enumeration for creation of decoy file proposed by Lalson et al. (2019) takes several hours in very large file systems. Hence, there is a need to maintain a balance between the file size and the computation time for creating large decoy files. The threshold can be tweaked to suit the peculiarities of each system. For example, a high threshold may be used in critical systems to minimize the false positive rate, while home systems may have threshold values lower than those of critical systems. Future research works should also consider enhancing the technique for detecting multi-stage crypto ransomware attacks suggested by Zimba et al. (2018) to prioritize the security of production network devices using a cascaded network segmentation approach. Research effort should also concentrate on detecting network-level ransomware attacks because many ransomware now communicate with the command-and-control server via encrypted channels such as the HTTPS protocol. The work of Makinde et al. (2019) is limited by the fact that the simulation involved few users. A future work should focus on using tools for big data analytics to simulate the behaviour of large number of users. The solution proposed by Sheen and Yadav (2018) applied class imbalance due to unequal number of samples in ransomware dataset and benign dataset. The same technique should be applied on a balanced dataset using the same classifiers and observe the outcome.

Although the Deep Packet Inspection technique proposed in Aragom et al. (2016) has 93% accuracy, the model currently supports static analysis. It can be extended to handle dynamic analysis by implementing it on a software defined network to support real time ransomware detection. Another possible extension is to improve the feature selection process applied to pcap files, such that the enhanced method compares the extracted features with those obtained from preceding or successive packets in order to obtain a better detection rate. The results obtained from the simple MLP network proposed in Vinayakumar et al. (2017) do not represent the actual situation involving more complex architecture settings. Future works should focus on using more complex MLP networks to perform the same experiments on state-of-the-art hardware in a distributed environment. Zahra and Sha (2017) proposed an IoT ransomware detection technique without actual implementation and deployment in a real-world environment. The proposed technique should be prototyped and deployed in a real-world IoT environment in order to evaluate and refine it. A future work should focus on increasing the accuracy of Randroid proposed by Alzahrani et al. (2018) by adding of more samples of malicious images and strings to the ISM database and the SSM database respectively. The new images should include logos of governments and icons of law enforcement agencies. This will help detect ransomware variants that exploit false positives such as fake FBI notes. Additional consideration should be given to the application of text recognition techniques on more images and texts to verify the ability of the dynamic analysis component to detect dynamic payloads. Chen et al. (2018) suggested that detecting mobile ransomware is constrained by the unavailability of a comprehensive dataset and limited understanding of real-time ransomware attack. Future research should consider creating a comprehensive and up-to-date dataset of mobile ransomware and developing a deep understanding of real-time ransomware attack against mobile devices.

Recent studies also have limitations and gaps which may be explored by future research. Future research based on the work of Rani and Dhavale (2022) should consider the integration of the model with Elasticsearch Logstash Kibana (ELK) to develop a practical tool for real-life ransomware detection. ELK can serve as the backend for filtering and collecting useful log data for the ransomware detection system. The detection system will then process logs of suspicious activities to determine whether such events are actually ransomware attacks. The work of Ahmed et al. (2022) focused only on the use of static and dynamic features for detecting unknown attacks by malicious Android malware. The study can be extended to explore distinct and detailed features of known ransomware samples, attacks that can be launched by such ransomware samples, qualitative and economical strategies for feature extraction, and malicious feature estimation. Researchers may also propose suitable metrics to determine the resistance of ransomware against countermeasures as well as the performance of defence mechanisms against ransomware attacks. The inability of the byte-frequency-based indicators proposed by Kim et al. (2022) to detect smaller (less than 256 bytes) files also represents an important research problem. This is because attackers may evade detection by using small-sized ransomware files to exploit computer systems. The approach can also be enhanced to address its inability to detect the DMA Locker2 ransomware. The novel Bayesian Neural network known as the Radial Spike and Slab Bayesian Neural Network (Nazarovs et al., 2022) requires human intervention for disabling and isolating systems in the event of ransomware attack. Future works should explore an enhanced solution which automatically disables systems and prevent access to the network once there is a suspected ransomware attack. Techniques which use enhanced feature extraction methods for better ransomware detection also require improvements. The two-stage particle swarm optimization proposed by Abbasi et al. (2022) requires improvements such as the use of more feature sets in the experimental dataset to capture additional behavioral traits such as communication involving critical servers or command and control centre. Also, certain future sets may be removed from the dataset and observe the impact of such removal on performance. In addition to this, intending researchers may explore the use system call sequences as additional features for classifying ransomware into families.

7 Conclusion

Ransomware attacks have done and are still doing significant damages to computers as well as data and information they process. These include unauthorized access, disclosure and destruction of vital, sensitive and critical computer and hardware resources. Both individuals and corporations have suffered grave financial losses and reputational damages due to ransomware attacks. Hence, several methods have been proposed for accurate, timely and reliable ransomware detection techniques. The background discussion on ransomware as well as the historical background and chronology of ransomware attacks presented in this study provide readers with the much-needed introduction to ransomware detection. The detailed and critical review of recent papers provide readers with an up-to-date knowledge of the current trends in automated ransomware detection. This will equip readers with the knowledge of the state-of-the-art in automated ransomware detection, prevention, mitigation and recovery. Also included in this study is an exposé on future research directions to provide intending researchers with open issues and possible research problems in detection, prevention, mitigation of and recovery from ransomware attacks.

References

Acronis International (2021). How machine learning can be used to prevent ransomware. Retrieved from https://www.acronis.com/en-eu/articles/machine-learning-prevent-ransomware.

Adamov, A. & Carlsson A. (2017). The state of ransomware. Trends and mitigation techniques. IEEE East-West Design & Test Symposium (EWDTS), 1-8, doi: 10.1109/EWDTS.2017.8110056.

Adamu, U. & Awan, I. (2019). Ransomware prediction using supervised learning algorithms. FiCloud 2019, Istanbul, Turkey, 57–63. doi: 10.1109/FiCloud.2019.00016.

Agrawal R., Stokes J.W., Selvaraj K. & Marinescu, M. (2019). Attention in recurrent neural networks for ransomware detection. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 3222-3226, doi: 10.1109/ICASSP.2019.8682899.

Ahmad, A., Kaiiali, M., Sezer, S. & O'kane P. (2019). A multi-classifier network-based crypto ransomware detection system: a case study of locky ransomware. IEEE Access, vol. 7, doi: 10.1109/ACCESS.2019.2907485.

Ahmed, U., Lin J.C.W. & Srivastava, G. (2022). Mitigating adversarial evasion attacks of ransomware using ensemble learning. Computers and Electrical Engineering, 100 (2022) 107903.

Ahmed Y.A., Huda S., Al-rimy B.A.S., Alharbi N., Saeed F, Ghaleb F.A. & Ali I.M. (2022). A weighted minimum redundancy maximum relevance technique for ransomware early detection in industrial iot sustainability. MDPI. 14(1231), 1-15. Retrieved from https://doi.org/10.3390/su14031231.

Aidan J., Zeenia, S. & Garg, U. (2018). Advanced petya ransomware and mitigation strategies. First International Conference on Secure Cyber Computing and Communication (ICSCCC). 23-28, doi: 10.1109/ICSCCC.2018.8703323.

Al-Dwairi M., Shatnawi A.S., Al-Khaleel, O. & Al-Duwairi, B. (2022). Ransomware-resilient self-healing XML documents. Future Internet, 14(115), 1-19. Retrieved from https://doi.org/10.3390/fi14040115.

Alzahrani A. (2018). RanDroid: structural similarity approach for detecting ransomware applications in android platform. IEEE International Conference on Electro/Information Technology (EIT), 0892-0897. doi: 10.1109/EIT.2018.8500161.

Ameer, M. (2019). Android Ransomware Detection using Machine Learning Techniques to Mitigate Adversarial Evasion Attacks. (Capital University of Science and Technology, Islamabad, Pakistan).

Andronio N., Zanero S. & Maggi F. (2015). HelDroid: dissecting and detecting mobile ransomware. In Research in Attacks, Intrusions, and Defenses. Lect. Notes Comput. Sci., vol. 9404, 382–404.

Aragorn, T., Yun-chun, C., YiHsiang, K., & Tsungnan, L. (2016). Deep learning for ransomware detection. Retrieved from https://www.semanticscholar.org/paper/Deep-Learning-for-Ransomware-Detection-Aragorn-Yun-chun/cc3a41b37230861cfe429632744e0d1db19256b7.

Arslan A., Abdul A., Umme Z., & Asifullah, K. (2020). Ransomware analysis using feature engineering and deep neural networks. Retrieved from https://arxiv.org/abs/1910.00286v2.

Azmoodeh A., Dehghantanha A., Conti M, & Choo K. R (2018). Detecting crypto Ransomware in IoT networks based on energy consumption footprint. Ambient Intell Human Comput 9, 1141–1152, Retrieved from https://doi.org/10.1007/s12652-017-0558-5.

Bazrafshan, Z., Hashemi, H, Fard, S.M.H. & Hamzeh, A. (2013). A survey on heuristic malware detection techniques. The 5th Conference on Information and Knowledge Technology, 113-120, doi: 10.1109/IKT.2013.6620049.

Brewer, R. (2016), Ransomware attacks: detection, prevention and cure. Netw. Secur, 1–6.

Cabaj, K., Gregorczyk, M., & Mazurczyk, W. (2017). Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. Comput. Electr. Eng., 353-368.

Celdrán A.H, Sánchez P.M.S, Castillo M.A, Gérôme B, Gregorio M.P. & Burkhard S (2022). Intelligent and behavioral-based detection of malware in IoT spectrum sensors. Int. J. Inf. Secur, 1-21. Retrieved from https://doi.org/10.1007/s10207-022-00602-w.

Chen, J., Wang, C., Zhao, Z., Chen, K., Du, R. & G.-J. Ahn (2018). Uncovering the face of android ransomware: characterization and real-time detection. IEEE Trans. Inf. Forensics Secur. 1286–1300.

Crowdstrike (2022a). How ransomware works. Retrieved from https://www.crowdstrike.com/resources/infographics/

how-fileless-ransomware-works/

Crowdstrike (2022b). Fileless Malware Explained. Retrieved from https://www.crowdstrike.com/cybersecurity-101/malware/fileless-malware/

Dargahi, T., Dehghantanha, A., Bahrami, P. N., Conti, M., Bianchi, G., & Benedetto, L. (2019). A cyber-kill-chain based taxonomy of crypto-ransomware features. Journal of Computer Virology and Hacking Techniques, 15(4), 277-305. Retrieved from https://doi.org/10.1007/s11416-019-00338-7.

Dehghantanha, A., Baldwin, J., & Alhawi. O. M. K. (2018). Leveraging machine learning techniques for windows ransomware network traffic detection. Retrieved from https://doi.org/10.1007/978-3-319-73951-95.

Dontov, D. (2019). Ransomware detection using machine learning. Retrieved from https://spinbackup.com/blog/ransomware-detection-using-machine-learning/

Du, J., Raza, S.H., Ahmad, M., Alam, I., Dar, S.H, & Habib, M.A, (2022). Digital forensics as advanced ransomware pre-attack detection algorithm for endpoint data protection. Security and Communication Networks. 1-16. Retrieved from https://doi.org/10.1155/2022/1424638.

eScan (2017). Antivirus reports.

F-Secure Labs (2013). Threat Report H1, Helsinki, Finland.

Fingers, J. (2020). Ransomware may have led to the death of a German hospital patient. Retrieved from www.google.com/amp/s/www.engadget.com/amp/ransomware-death-at-german-hospital-210309749.html.

Fitzpatrick, D. & Griffin, D. (2016). Cyber-extortion losses skyrocket, says FBI. Retrieved from http://money.cnn.com/2016/04/15/technology/ransomwarecyber-security.

Gallegos-Segovia, P.L., Bravo-Torres, J.F., Larios-Rosillo, V.M., Vintimilla-Tapia, P.E., Yuquilima-Albarado, I.F.

& Jara-Saltos J.D. (2017). Social engineering as an attack vector for ransomware. CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 1-6, doi: 10.1109/CHILECON.2017.8229528.

Gers, F.A., Schmidhuber, J. & Cummins, F.A (2000). Learning to forget: Continual prediction with lstm, Neural Computation. Neural Comput 2000. 12(10) 2451–2471. Retrieved from https://doi.org/10.1162/

089976600300015015

Gómez-Hernández, J.A., Sánchez-Fernández, R. & García-Teodoro, A. (2022). Inhibiting crypto-ransomware on windows platforms through a honeyfile-based approach with R-Locker. IET Inf. Secur. 16(1), 64–74. Retrieved from https://doi.org/10.1049/ise2.12042.

Gopinath, S. & Olmstead, A. (2022). Mitigating the effects of ransomware attacks on healthcare systems.

Hwang J, Kim J, L. S, & Kim K (2020). Two-stage ransomware detection using dynamic analysis and machine learning techniques. Wireless Pers Commun 112, 2597–2609, Retrieved from https://doi.org/10.1007/s11277-020-07166-9.

Jasmin, M. (2019). Detecting ransomware in encrypted network traffic using machine learning. (University of Victoria, Canada). Retrieved from http://hdl.handle.net/1828/11076.

Juan, A., Silver, H., & Hernández-Alvarez, M. (2017). Ransomware detection by cognitive security, IEEE, 346–363.

Khammas, B. (2020). Ransomware detection using random forest technique. ICT Express, 6(4), 325–331.

Khammas, B.M. (2022). Comparative analysis of various machine learning algorithms for ransomware detection. TELKOMNIKA Telecommunication Computing Electronics and Control, 20(1), 43~51.

Kharraz A., Robertson W, Balzarotti D, Leyla Bilge & Kirda E (2015). Cutting the gordian knot: a look under the hood of ransomware attacks In: M. Almgren., V. Gulisano, F. Maggi. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA Lecture Notes in Computer Science, vol 9148. Springer, Cham. Retrieved from https://doi.org/10.1007/978-3-319-20550-2_1.

Kim, G., Kim, S., Kang, J. & Kim, J. (2022). A method for decrypting data infected with hive ransomware. arXiv:2202.08477v1 [cs.CR], 1-23.

Kim, G.Y., Paik J.Y. & Kim Y. (2022). Byte frequency-based indicators for crypto-ransomware detection from empirical analysis. Journal of Computer Science and Technology, 37(2). DOI 10.1007/s11390-021-0263-x.

Lalson, E.R., Shony, K.M, & Netto, D.F. (2019). An integrated approach for detecting ransomware using static and dynamic analysis. FiCloud 2019, 410–414. doi: 10.1109/FiCloud.2019.00016.

Lee, K., Lee, S,, & Yim, K, (2019). Machine learning based file entropy analysis for ransomware detection in backup systems. IEEE Access, 110205–110215, doi: 10.1109/ACCESS.2019.2931136.

Lee, S., Jho, N., Chung D, Kang, Y. & Kim, M. (2022). Rcryptect: real-time detection of cryptographic function in the user-space filesystem. Computers & Security. 112, 1-13.

Makinde, O., Sangodoyin, A., Mohammed, B., Neagu, D., & Adamu, U. (2019). Distributed network behaviour prediction using machine learning and agent-based micro simulation. FiCloud 2019, 182-188.

Maniath S, Ashok A., Poornachandran P., Sujadevi G., Sankar,. A.U. & Jan, S (2017). Deep learning LSTM based ransomware detection. Recent Dev. Control Autom. Power Eng., 442–446, doi: 10.1109/RDCAPE.2017.8358312.

Matthias, H. (2018). Detecting ransomware. (Universität Konstanz).

McIntosh, T., Kayes, A.S.M., Chen, Y.P.P., Ng, A. & Watters, P, (2021). Ransomware mitigation in the modern era: a comprehensive review, research challenges, and future directions. ACM Computing Surveys (CSUR), 54(9), 1-36. Retrieved from https://doi.org/10.1145/3479393.

Microsoft Ignite (2022). What is ransomware? Retrieved from https://docs.microsoft.com/en-us/security/compass/human-operated-ransomware.

Mohurle, S., & Patil, S. (2017). Brief study of wannacry ransomware attack. Int. J. Adv. Res. Comput. Sci., vol. 8, 1938–1940.

Moore, C. (2016), Detecting ransomware with honeypot techniques. Cybersecurity and Cyberforensics Conference (CCC). 77-81. doi: 10.1109/CCC.2016.14.

Morato, D., Berrueta, E., Magaña E., & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. J. Netw. Comput. Appl., 14–32.

Nazarovs, J., Stokes, J.W, Turcotte, M., Carroll, J. & Grady, I. (2022). Radial spike and slab bayesian neural networks for sparse data in ransomware attacks. arXiv:2205.14759v1 [cs.CR] 1-17.

Olani, G., Wu, C-F. & Chang, Y-H. (2022). DeepWare: imaging performance counters with deep learning to detect ransomware. IEEE Transactions on Computers, Vol. X, No. X, XXX 20XX, pp. 1-15.

Oz, H., Aris, A., Levi, A., & Uluagac, A. S. (2021). A survey on ransomware: evolution, taxonomy, and defense solutions. ACM Computing Surveys (CSUR). Retrieved from https://doi.org/10.1145/3514229.

Patel, A. & Tailor, J, (2020). A malicious activity monitoring mechanism to detect and prevent ransomware. Comput. Fraud Secur, 14–19.

Potoroaca, A. (2020). Over 41% of cyber insurance claims in 2020 came from ransomware attacks. Retrieved from https://www.techspot.com/amp/news/86714-over-41-percent-cyber-insurance-claims-2020-came.html.

Poudyal, S., Subedi, K.P. & Dasgupta, D. (2018). A framework for analyzing ransomware using machine learning. IEEE Symposium Series on Computational Intelligence (SSCI), 1692-1699. doi: 10.1109/SSCI.2018.8628743.

Rahman, M. & Hasan, M. (2017). A support vector machine-based ransomware analysis framework with integrated feature set. 20th International Conference of Computer and Information Technology, Dhaka, 1–7. doi: 10.1109/ICCITECHN.2017.8281835.

Rani, N. & Dhavale, S.V. (2022). Leveraging machine learning for ransomware detection. arXiv:2206.01919v1 [cs.CR], 1-13.

Ransomware attacks. (2021). Top 5 ransomware attacks to watch out for in 2020-2021. Retrieved from https://www.google.com/amp/s/top-5-ransomware-attacks-to- watch-out-for-in-2020-2021/amp.

Richardson, R. & North, M. (2017). Ransomware: evolution, mitigation and prevention. Int. Manag. Rev., vol. 13, 10–21.

Savage, K., Coogan P, & Lau, H. (2015). The evolution of ransomware. Secur. Response, Symantec. Retrieved from https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware.pdf.

Scaife, N., Carter, H., Traynor, P, & Kevin, B. (2016). CryptoLock (and drop it): stopping ransomware attacks on user data. IEEE 36th Int. Conf. Distrib. Comput. Syst.

Schmidhuber, J. & Sepp, H. (1997). Long short term memory. Neural Computation. 1735–1780.

Sgandurra D., Muñoz-González, L., Mohsen, R., & Lupu, E. (2016). Automated dynamic analysis of ransomware: benefits, limitations and use for detection. Retrieved from https://arxiv.org/abs/1609.03020, 1–12.

Sharmeen, S., Ahmed, Y.A., Huda, S., Koçer, B.S., & Hassan, M.M. (2020). Avoiding future digital extortion through

robust protection against ransomware threats using deep learning based adaptive approaches. IEEE Access, vol. 8, 24522–24534, doi: 10.1109/ACCESS.2020.2970466.

Shaukat, S., & Ribeiro, V. (2018). RansomWall: a layered defense system against cryptographic ransomware attacks using machine learning. 10th International Conference on Communication Systems and Networks, 356-363.

Sheen, S. & Yadav, A. (2018). Ransomware detection by mining api call usage. International Conference on Advances in Computing, Communications and Informatics (ICACCI), 983-987, doi: 10.1109/ICACCI.2018.8554938.

Singh, A., Ikuesan, R.A. & Venter, H. (2022). Ransomware detection using process memory. ICCWS 2022: 17th International Conference on Cyber Warfare and Security, 1-10.

Symantec Corporation (2016). Internet security threat report.

Talabani, H.S. & Abdulhadi, H.M.T. (2022). Bitcoin ransomware detection employing rule-based algorithms.

Science Journal of University of Zakho, 10(1), 5– 10.

Vehabovic, A., Ghani, N., Bou-Harb, E., Crichigno, J. & Yayimli, A. (2022). Ransomware detection and classification strategies. IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 316-324, doi: 10.1109/BlackSeaCom54372.2022.9858296.

Vinayakumar, R., Soman, K.P., Senthil, M., Velan, K. K. & Ganorkar, S. (2017). Evaluating shallow and deep networks for ransomware detection and classification. International Conference on Advances in Computing, Communications and Informatics (ICACCI), 259-265. doi: 10.1109/ICACCI.2017.8125850.

Wan, Y., Chang, J., Chen, R. & Wang, S. (2018). Feature-selection-based ransomware detection with machine learning of data analysis. 3rd International Conference on Computer and Communication Systems (ICCCS), 85-88, doi: 10.1109/CCOMS.2018.8463300.

Weckstén, M., Frick, J., Sjöström, A. & Järpe, E. (2016). A novel method for recovery from crypto ransomware infections. 2nd IEEE International Conference on Computer and Communications (ICCC). 1354-1358, doi: 10.1109/CompComm.2016.7924925.

Wongsupa, P. (2018). Deep learning for android application ransomware detection. MSc Dissertation. (Florida Atlantic University).

Yang, T., Yang, Y., Qian K., Lo, D.C, Qian, Y. & Tao, L. (2015). Automated detection and analysis for android

ransomware. IEEE 17th International Conference on High Performance Computing and Communications, IEEE 7th International Symposium on Cyberspace Safety and Security, and IEEE 12th International Conference on Embedded Software and Systems, 1338-1343, doi: 10.1109/HPCC-CSS-ICESS.2015.39.

Zahra, A. & Shah, M. (2017). IoT based ransomware growth rate evaluation and detection using command and control blacklisting. Proceedings of the 23rd International Conference on Automation & Computing, (University of Huddersfield, Huddersfield), 1–6.

Zetter, K. (2015). Hacker lexicon: A guide to ransomware, the scary hack that's on the rise. Retrieved from: https://www.wired.com/2015/09/hacker-lexicon-guideransomware- scary-hack-thats-rise/

Zimba, A., Wang, Z., & Chen, H. (2018). Multi-stage crypto ransomware attacks: a new emerging cyber threat to critical infrastructure and industrial control systems. ICT Express, vol. 4, 14–18.

EEMDS: Efficient and Effective Malware Detection System with Hybrid Model based on XceptionCNN and LightGBM Algorithm

[1,2*]Monday Onoja, [2,3]Abayomi Jegede, [2]Nachamada Blamah, [4]Olawale Victor Abimbola and [1]Temidayo Oluwatosin Omotehinwa

[1]Department of Mathematics and Computer Scicence, Faculty of Science, Federal University of Health Sciences, P.M.B 145, Otukpo, Nigeria.
[2]Department of Computer Scicence, Faculty of Natural Science, University of Jos, P.M.B 2084 Jos, Nigeria.
[3]Africa Centre of Excellence on Technology Enhanced Learning, National Open University, Abuja, Nigeria.
[4]Creative Advanced Technologies, Dubai, UAE.

email: [1,2*]monday.onoja@fuhso.edu.ng; [2,3]jegedea@unijos.edu.ng; [2]blamahn@unijos.edu.ng
[4]abimbolaolawale41@gmail.com; [1]oluomotehinwa@gmail.com

**Corresponding author*

Received: 13 June 2022 | Accepted: 24 October 2022 | Early access: 28 October 2022

Abstract - *The security threats posed by malware make it imperative to build a model for efficient and effective classification of malware based on its family, irrespective of the variant. Preliminary experiments carried out demonstrate the suitability of the generic LightGBM algorithm for Windows malware as well as its effectiveness and efficiency in terms of detection accuracy, training accuracy, prediction time and training time. The prediction time of the generic LightGBM is 0.08s for binary class and 0.40s for multi-class on the Malimg dataset. The classification accuracy of the generic LightGBM is 99% True Positive Rate (TPR). Its training accuracy is 99.80% for binary class and 96.87% for multi-class, while the training time is 179.51s and 2224.77s for binary and multi classification respectively. The performance of the generic LightGBM leaves room for improvement, hence, the need to improve the classification accuracy and training accuracy of the model for effective decision making and to reduce the prediction time and training time for efficiency. It is also imperative to improve the performance and accuracy for effectiveness on larger samples. The goal is to enhance the detection accuracy and reduce the prediction time. The reduction in prediction time provides early detection of malware before it damages files stored in computer systems. Performance evaluation based on Malimg dataset demonstrates the effectiveness and efficiency of the hybrid model. The proposed model is a hybrid model which integrates XceptionCNN with LightGBM algorithm for Windows Malware classification on google colab environment. It uses the Malimg malware dataset which is a benchmark dataset for Windows malware image classification. It contains 9,339 Malware samples, structured as grayscale images, consisting of 25 families and 1,042 Windows benign executable files extracted from Windows environments. The proposed XceptionCNN-LightGBM technique provides improved classification accuracy of 100% TPR, with an overall reduction in the prediction time of 0.08s and 0.37s for binary and multi-class respectively. These are lower than the prediction time for the generic LightGBM which is 0.08s for binary class and 0.40s for multi-class, with an improved 100% classification accuracy. The training accuracy increased to 99.85% for binary classification and 97.40% for multi classification, with reduction in the training time of 29.97s for binary classification and 447.75s for multi classification. These are also lower than the training times for the generic LightGBM model, which are 179.51s and 2224.77s for the binary and multi classification respectively. This significant reduction in the training time makes it possible for the model to converge quickly and train a large sum of data within a relatively short period of time. Overall, the reduction in detection time and improvement in detection accuracy will minimize damages to files stored in computer systems in the event of malware attack.*

Keywords: Anomaly-based Detection, LightGBM, Machine Learning, Malware Detection, XceptionCNN.

1 Introduction

Malware, also known as malicious software can be described as any instruction set that is compromised to alter a computer system and impose harm on users and organizations (Abusitta, 2021). A malware is categorized based on its activities and execution process (Singh & Singh, 2020). The internet has made great contributions in communication, however, it has consequently led to the rise in malware distribution. The developments of web services with increasing speed have made productive advantage available to end-users. The number of people using the Internet was about two billion in 2010 (Chang et al., 2013). A report from Dasient, and cited by Chang et al. (2013), suggests that the number of malware delivering websites doubled between 2009 and 2010. “There were 3.424 billion people using the Internet by July 2018” (Wang, 2018). Carrying out activities on an infected website is a sufficient pathway for an attacker to take advantage of the weakness of a browser.

Malware analysis is essential in order to build successful malware detection techniques. The focus of malware analysis is to understand the intent and activities of malware (Wong et al., 2021). Malware analysis may be static, dynamic, or hybrid depending on the way and manner it is carried out. Analysts use static analysis, dynamic analysis or a combination of both methods (hybrid) to understand and explain the mode of operation of malware and the effects on the system (Wong et al., 2021).

Malware detection is the process of recognizing malicious sets of instruction from benign ones, so that a defense can be built, in order that the system can be protected or recovered from any malicious effects (Landage & Wankhade, 2013). Malware detection techniques identify malicious codes and prevent the system from its effect and possible loss of information. A malware detector uses malware detection techniques to identify activities of malware. Figure 1 shows malware detection techniques and approach as presented by Kumar (2017).

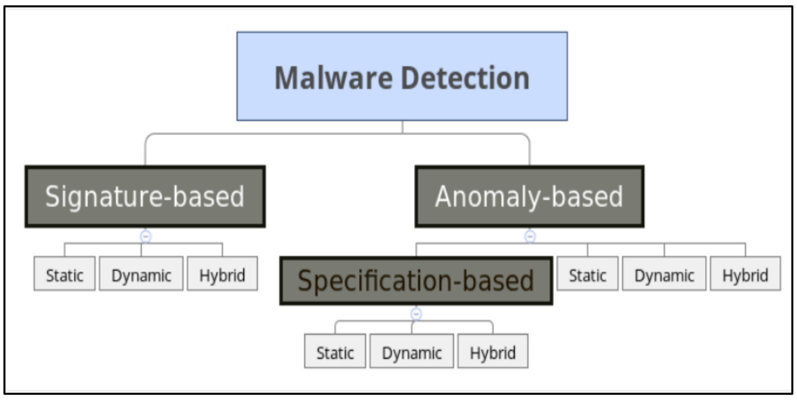

Figure 1: Malware detection techniques (Kumar, 2017)

1.1 Signature-Based Detection

A signature is a chain of information that describes the activities of a particular malware (Damodaran et al., 2017). In Signature-Based Detection approach, unique signatures are detached from captured malware files. The signatures are further used to detect malware with similar characteristics. Signature-based approach is suitable for generic malware which do go through significant behavioural modification (Abusitta, 2021). However, attackers easily manipulate malware signatures in order not to be detected by antivirus software (Abusitta, 2021). Signature-based models are very effective in known malware detections, but, it is unable to identify new ones (Bazrafshan et al., 2013).

1.2 Anomaly-Based Detection

In Anomaly-based detection; the behaviors of malware during runtime are studied in a training phase, after which the executable is tagged as malicious or benign during testing phase based on extracted patterns in the training phase (Damodaran et al., 2017). Behavior-based method is capable of detecting new and unknown malware and malware that uses obfuscation techniques. The main limitations of the behavior-based detection are: a substantial False Positive Rate (FPR) and unnecessary testing time (Bazrafshan et al., 2013).

Malware issue has developed into a serious issue in computing. According to Gibert et al. (2020), machine learning technique is the best technique that is needed to protect a computer system due to rise in malware attack. Using malware images makes malware classification easier (Pant & Bista, 2021). Image-based techniques are robust against many types of obfuscations (Bhodia et al., 2019). Omitting irrelevant features fasten and make algorithm to perform better (Şahin et al., 2021). While LightGBM technique is the best of the Gradient Boosting Decision Tree Algorithms (GBDT) and has demonstrated its suitability for malware detection (Abbadi et al., & Pan et al., 2020), XceptionCNN is an effective and less complex neural network for robust feature extraction (Shaheed & Zhang, 2022).

We conducted a preliminary study which reveals the prediction time of the generic LightGBM to be 0.08s for binary class and 0.40s for multi-class on the Malimg dataset with 10,381 malware samples. The preliminary study further reveals a classification accuracy of 99% (TPR), with training accuracy of 99.80% for binary classification on Malimg dataset and 96.87% for multi classification on the same malware samples. A growth in data size may lead to corresponding increase in time of prediction. Although the classification accuracy obtained from our preliminary experiment seems to be good, it could be further improved in order to enforce the effectiveness of the algorithm. The prediction time, performance accuracy, and training time obtained from the preliminary experiment also leave room for improvement. Hence, there will be a need to make the classification accuracy of the model better for effective decision making, reduce the prediction time for efficiency, and improve the performance and accuracy for effectiveness on larger samples. Our study, which hybridized XceptionCNN with LightGBM has the following contributions and novelty:

- Improved training accuracy of the model for effective decision making.
- Reduction in the detection time and improvement in detection accuracy, which will minimize damages to files stored in computer systems in the event of malware attack.
- Reduction in the training time, which will enable the model to converge quickly and train a large amount of data within a relatively short period of time.
- The proposed model can detect both known and new malware variants.
- The training accuracy of the proposed model is higher than those of the existing models.
- This study is the first to compute and improve the detection time of the LightGBM algorithm

This study proposes a hybrid model based on LightGBM and XceptionCNN algorithms. The aim is to improve the efficiency and effectiveness of Windows malware detection. Our preliminary study revealed that the LightGBM technique which is the best of the GBDT algorithm, has proven to be suitable for Windows malware detection (Abbadi et al., 2020; Pan et al., 2020) and can be improved for effective and efficient malware detection. ML-based classifiers use underlying features to distinguish between malicious and benign applications, and detecting changes in those features when malicious modifies itself. Malware possess certain features which Machine learning algorithm can learn and use to predict if an executable file is a malware or benign sample. Such that, if such an executable file behaves in a certain way or attempts to modify the system or access privilege instructions, they may be classified as malware or benign based on their activities. Our technique can take a number of these features as input, learn the properties of these features, then, build a model for prediction of new samples. Using the mathematical function$f: m \rightarrow x$, where m is the given malware and x is their corresponding malware family, the model can also detect new variants of malware. This study will contribute to advances in malware detection. The proposed solution can be applied to similar studies in future.

The rest of this paper is organized as follows: Section 2 presents related work in malware detection, while Section 3 discusses experiments and implementation details. Section 4 covers the findings of our study and the implications of those findings, while Section 5 summarizes the study and draws conclusions based on the achieved objectives.

2 Related Work

Ke et al. (2017) posited that GBDT is one of the machine learning classifiers which is extensively utilized as a result of its effectiveness. LightGBM is a new GBDT algorithm with Gradient-based One-Side Sampling (GOSS) and the Exclusive Feature Bundling (EFB) (Ke et al., 2017). The GOSS and EFB makes LightGBM considerably effective than eXtreme Gradient Boosting (XGBoost) with regards to speed and memory usage (Ke et al., 2017). In an attempt to study user's click for click fraud detection, Minastireanu and Mesnita (2019) used the experimental test for LightGBM using a public dataset available on Kaggle. Their results on the GBDT Algorithm achieved 98% accuracy. In a different study, using the N-Gram model, Venkat et al. (2020) proposed a medicine approval system. In an attempt to boost the efficiency of the medicine approval system, a LightGBM model was used to carry out medication examination. Ju et al. (2019) observed that single-convolution model was ineffective for wind power prediction; hence, they proposed a solution which integrates LightGBM algorithm with convolutional model to enhance the prediction accuracy and reliability. A study by Fonseca et al. (2017) showed that training LightGBM algorithm is faster than training XGBoost. The study did not compare the algorithms based on their classification time. In further study, using dataset made available through Kaggle's competition, Machado et al. (2019) evaluated the accuracy of two GBDT Models: XGBoost and LightGBM algorithm in predicting credit card customers' reliability status. The study assessed customer loyalty prediction accuracy through Root Mean Square Error and found that LightGBM achieved better than XGBoost. LightGBM-based method performed better than most generic methods (such as Support Vector Machine, XGBoost, or Random Forest) when applied fraud detection (Huang, 2020). Sun et al. (2020) also used LightGBM to combine the daily data of 42 kinds of primary crypto-currencies with key important pointers in order to predict the prices of crypto-currencies and obtain relevant information about the market. Their experimental results show that the robustness of the LightGBM model is better than the other models. The time complexity for the LightGBM is calculated as O(#Data x #Features) (Meidan et al., 2018). A malware classification approach converted malware binaries to grayscale images before using a trained CNN to build a model for classifying malware according to its family (Kalash et al., 2018). A deep learning architecture applied to Malimg malware dataset and Microsoft dataset has performance accuracy of 98.52% for malign dataset and 99.97% accuracy for Microsoft datasets (Kalash et al., 2018). In a similar study carried out by Bhodia et al. (2019), their deep learning architecture which was also experimented on the Malimg malware dataset yielded a training accuracy of 98.39% for binary classification and 94.80% for multi-classification. A study by Lo et al. (2019) classified malware into families based on the integration of deep CNN with Xception model. Experimental results show that the training accuracy of the Xception model on the Malimg datasets is 99.37%. The study did not evaluate the classification accuracy and prediction time of the model. Hussain (2019) proposed a hybrid technique based on gradient boosting classifier. The method used information such as target of applications, privileges, static data and dynamic data to detect malicious application. The approach has a good detection accuracy of 96%. This result still leaves room for improvement. In another study, Nawaz et al. (2021) performed hybrid analysis using Android application features which include permissions, targets, and network features. The study extracts permissions and intent from a suspected file. It also obtains network related information from java files. The use of Info Gain as a feature selection method results in precision of 0.99. The study did not apply their model to Windows malware domain. An improved solution uses a small set of highly discriminant features for automated malware detection (Fang et al., 2019). The goal is to address the limitations of classical feature selection techniques. DQFSA interacts with the feature space and uses Q-learning to train an agent in order to achieve high accuracy. The proposed approach performs better than existing baseline feature selection methods for malware detection using small feature sets. The study did not apply the framework to other selection tasks. In an attempt to construct a detection framework, Chen et al. (2020) used the characteristics of data and features of the attention mechanism to construct a sliding local attention mechanism model (SLAM). The performance accuracy of the proposed model is 0.9723. The study did explore the used of the technique for malware detection. Bensaoud and Kalita (2022) proposed a novel deep learning method for classifying malware images for effective and efficient malware detection. Experimental results based on about 100,000 benign and malicious PE, APK, Mach-o, and ELF show that the method has the highest accuracy of more than 99.87%. The method is also effective at detecting different malware evasion techniques. The detection time of the model was not considered. Pan et al. (2020) used Logistic Regression, KNN and LightGBM to build models based on datasets of heartbeat and threat reports. The results obtained from the respective models show that LightGBM has the highest accuracy with AUC of 0.720687. The study attempted to enhance the training accuracy of the models, however, the detection time and detection accuracy were not considered. Using a custom model based on convolutional neural network with a benchmark dataset (Malimg dataset), Pant and Bista (2021) achieved 99.64% training accuracy while classifying malware into their respective families. The study did not consider the detection time of the model. The study also did not evaluate the model based on binary classification of the malware. A preliminary study conducted by the authors of this article reveal that the LightGBM achieves 0.08s prediction time for binary classification on Malimg dataset with 10,381 malware samples and a prediction time of 0.40s for multi classification on the same malware samples. The preliminary experiments reveal that the classification accuracy of the generic LightGBM machine learning algorithm is 99% TPR. From the literature review, no study has deemed it fit to improve the classification accuracy or reduce the prediction time of the LightGBM algorithm for windows malware detection.

3 Methodology

The research process flow in Figure 2 depicts the sequence of activities required to accomplish the overall objectives of the study.

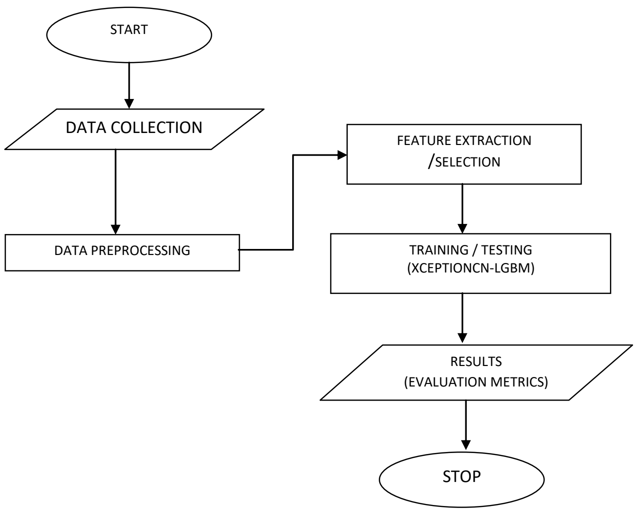
START
DATA COLLECTION
DATA PREPROCESSING
FEATURE EXTRACTION /SELECTION
TRAINING / TESTING (XCEPTIONCN-LGBM)
RESULTS (EVALUATION METRICS)
STOP

Figure 2: Research process flow

3.1 Data Collection

We used the Malimg malware dataset which contain 9,339 of Malware samples structured as grayscale images consisting of 25 malware families. Each of the malware families is made up of varying number of samples across the dataset. Malimg dataset is one of the most commonly used datasets for malware findings. The Malimg dataset was created by reading malware binaries into an 8-bit unsigned integer consisting of a matrix $M \in \mathrm{R}^{m \times n}$ (Nataraj et al., 2011). The matrix could be seen as image (grayscale) having values within the range of [0, 255], where 0 represents black, 1 represents white. In addition, benign dataset used are 1,042 windows benign executable files extracted from windows environment and further converted into images. Table 1 presents a breakdown of the Malimg dataset into families and their variants.

Table 1: The Malimg Dataset (Nataraj et al., 2011).

NO.	Family Name	Family	Samples
1	Adialer.C	dialer	122
2	Agent.FYI	Backdoor	116
3	Allaple.A	worm	2,949
4	Allaple.L	worm	1,591
5	Alueron.gen!J	Trojan	198
6	Autorun.K	worm	106
7	C2LOP.gen!g	Trojan	200
8	C2LOP.P	Trojan	146
9	Dialplatform.B	Dialer	177
10	Dontovo.A	Trojan Downloader	162
11	Fakerean	Rogue	381
12	Instantaccess	Dialer	431
13	Lolyda.AA1	PWS	213
14	Lolyda.AA2	PWS	184
15	Lolyda.AA3	PWS	123
16	Lolyda.AT	PWS	159
17	Malex.gen!J	Trojan	136
18	Obfuscator.AD	Trojan Downloader	142
19	Rbot!gen	Backdoor	158
20	Skintrim.N	Trojan	80

21	Swizzor.gen!E	Trojan Downloader	128
22	Swizzor.gen!I	Trojan Downloader	132
23	VB.AT	worm	408
24	Wintrim.BX	Trojan Downloader	97
25	Yuner.A	worm	800
	Total	—	**9,339**

The analysis of the families and variants of the Malimg Dataset with a total of 9,339 dataset (Nataraj et al., 2011).

3.2 Data Preprocessing

The Malimg dataset used consists of images, hence, there was no preprocessing done on the dataset. We directly passed the images that consist of the dataset into XceptionCNN for automatic feature extraction. Thereafter, we saved the extracted features as CSV and passed it for classification and training. We also separated the dataset into training and testing sets. Since our model require images as input, we further converted the windows benign executable file to images using 'exe2image' converter, a digital image converter software available on github (Malith, n.d.), operated on windows.

3.3 Feature Extraction

A total of 10,381 images of two (2) classes were used for the binary classification and the same number of images of twenty (26) classes were used for the multi-classification. We applied an Advanced Convolutional Neural Network model (Xception) which extracted the image features with distinctive pattern automatically from the dataset, and the extracted features were saved as CSV files. Malware that share the same family are very similar in layout and image (Nataraj et al., 2011).

3.4 Training and Testing

We trained the models on 100 epochs using the following hyperparameters : learning rate of 0.03 & Max-dept of 10. We used a total of 10,381 data samples. We randomly selected 80% and 20% of the samples in each of the families for training and testing respectively, resulting in 8,304 and 2,077samples used for training and testing respectively. The LightGBM was used for training and testing.

3.5 LightGBM Classifier

LightGBM is a new gradient boosting framework. It is a decision tree algorithm which supports many algorithms like Gradient Boosting Machine (GBM), Gradient Boosting Decision Tree (GBDT), Gradient Boosted Regression Tree (GBRT), and Multiple Additive Regression Tree (MART). It is has high level scalability, precision, and efficiency (Ke et al., 2017). It is suitable for classification and other machine learning activities (Abbadi et al., 2020). It applies a leaf-wise splitting of the tree based on the best fit, unlike other boosting algorithms which use depth-wise or level-wise splitting. The leaf-wise growth of LightGBM reduces the level of loss, which results in faster speed and higher accuracy than other boosting algorithms. Figure 3 shows the leaf-wise tree growth structure of the LightGBM algorithm.

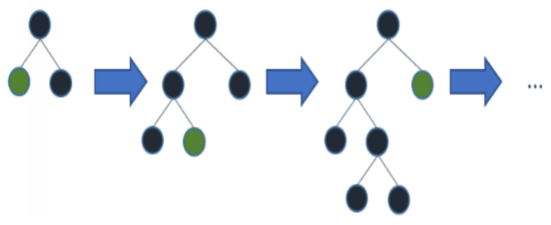

Figure 3: LightGBM Architecture (Khandelwal, 2017)

Figure 3 shows the leaf-wise tree growth (architecture) of the LightGBM algorithm as presented by Khandelwal (2017).

Leaf-wise splits results in high complexity and overfitting, which can be addressed by using a parameter known as max-depth to indicate how deep the splitting should be (Microsoft, 2021). LightGBM algorithm was proposed by Su et al. (2018) and has been applied in different studies such as Abbadi et al. (2020) and Fonseca et al. (2017). Abbadi et al. (2020) used LightGBM in IoT malware detection. LightGBM uses Gradient-based One-Side Sampling (GOSS) and the Exclusive Feature bundling (EFB) (Sharma, 2018) to minimize the complexity of histogram building (O(data*feature)). This is achieved by using *GOSS* and *EFB* to reduce the sampled data and feature size. Hence, the complexity becomes (O(data2 * bundles)) where data2 < data and bundles << feature (Sharma, 2018). LightGBM algorithm works on a supervised training set to compute an approximate function that minimizes the value of a specific loss function $L(y, f(x))$ as expressed:

$$\hat{f} = minEy, xL(y, f(x)) \tag{1}$$

where x represents a set of random input variable and y represents a random output or response variable. LightGBM computes an approximation of the final model by combining multiple T regression trees $\sum_{t=1}^{T} f_t(X)$ which is expressed as

$$f_T(x) = \sum_{t=1}^{r} f_t(x) \tag{2}$$

The regression trees could be expressed as $w_{q(x),} q \in \{1,2,\dots,j\}$ where j denotes the number of leaves, q represents the decision rules of the tree and w is a vector that denotes the sample weight of leaf nodes. Hence, LightGBM would be trained in an additive form at step t as follows:

$$\Gamma_t = \sum_{i=1}^{n} L(y_{i,} F_{t-1}\left(x_{i,}\right) + f_{t,}(x_{i,})) \tag{3}$$

In LightGBM, the objective function is rapidly approximated using Newton's method. After removing the constant term in the last equation for simplicity, the formulation can be represented as

$$\Gamma_t \cong \sum_{i=1}^{n} (g_i f_t\left(x_{i,}\right) + \frac{1}{2} h_i f_t^2\ (x_{i,})) \tag{4}$$

where g_i and h_i denote the first and second-order gradient statics of the loss function.

Let I_j denote the sample set of leaf j, (4) could be expressed as

$$\Gamma_t = \sum_{j=1}^{j} ((\sum_{i \in I_j} g_i) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2)) \tag{5}$$

For a certain tree structure$q(x)$, the optimal leaf weight scores of each leaf node w_j^* and the extreme value of Γ_k can be expressed as:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{6}$$

$$\Gamma_T^* = -\frac{1}{2} \sum_{j=1}^{j} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} \tag{7}$$

where
Γ_T^* is the scoring function that measures the quality of the tree structure q. Finally, the objective function after adding the split is:

$$G = \frac{1}{2}\left(\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda}\right) \tag{8}$$

where I_L and I_R are the sample sets of the left and right branches respectively. Unlike traditional GBDT based techniques such as XGboost and GBDT which grow trees horizontally, LightGBM adopts a vertical approach to grow the tree, which makes the algorithm effective for handling large datasets and features. LightGBM increases training performance and minimizes memory requirements by using algorithms based on the histogram. The advantages of LightGBM as presented by Khandelwal (2017) are as follows:

i. Higher efficiency: It uses histogram-based algorithm to convert continuous feature values into discrete bins which results increases the speed of training a dataset.
ii. Reduced memory requirements - the replacement of continuous values with discrete bins results in low memory utilization.
iii. Higher accuracy than similar techniques – by using leaf-wise instead of level-wise splitting to compute more complex trees.
iv. Suitable for large datasets – performs well on large datasets and minimizes the training time considerably when compared to XGBOOST.
v. It supports parallel learning.

3.6 Architecture of XceptionCNN

Xception model was developed using an 'extreme' interpretation of Google's Inception model (Chollet, 2017). Its structure is a linear stack of 36 independent convolution layers which use depth-wise splitting method and are linked together by residual connections. The layered stack is responsible for feature extraction on the network. The 36 independent convolution layers are grouped into 14 modules, which are joined by linear residual connections, with the exception of the first and last modules (Chollet, 2017). XceptionCNN diagrammatic representation is shown in figure 4.

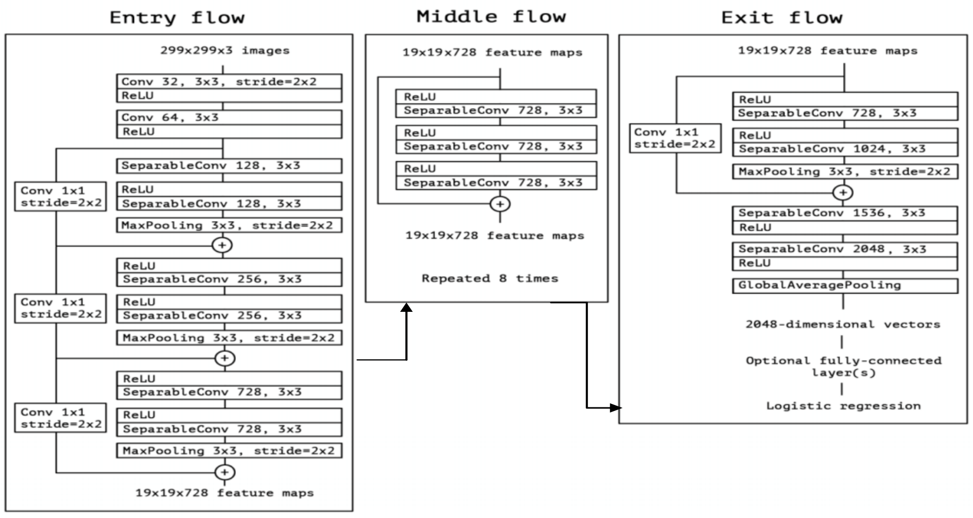

Figure 4: XceptionCNN Architecture (Chollet, 2017)

Figure 4 shows the Xception architecture. It is divided into 3 components (entry flow, middle flow and exit flow), 14 modules and 36 convolutional layers. It uses the layers with a depth of 126 to perform feature extraction. Its input format is a 299x299 RGB image. A global average pooling layer is substituted for the fully-connected layer to reduce the parameter size, while the softmax function is used to predict the output. The flow of data from the entry flow to the middle flow is repeated eight times before it finally passes through the exit flow. The number of convolutional layers in the entry flow, middle flow and exit flow are 8, 8*3= 24 and 4 respectively. The model uses depth-wise separable convolution to reduce the operational cost of the convolution process.

3.7 Design of the XceptionCNN - LightGBM Experiments

To reduce the prediction time and take advantage of the efficiency of XceptionCNN, we combine XceptionCNN with LightGBM. We use the XceptionCNN to extract features automatically; it uses less resource and time as compared to the already available methods. The XceptionCNN is hybridized with the LightGBM model. We directly passed the images that comprise the dataset into XceptionCNN model for automatic feature extraction, thereafter, saved the extracted features as CSV and passed it to the LightGBM model for classification and training.

In order to demonstrate that the XceptionCNN-LightGBM can reduce prediction time and training time, improve the performance and improve the training accuracy, experiments with XceptionCNN-LightGBM were conducted. We compared XceptionCNN-LightGBM models with the generic LightGBM models (preliminary experiment) to verify that using XceptionCNN-LightGBM can reduce prediction time and training time, improve classification accuracy and training accuracy. We also compared our results with similar studies to show that our model performs better on the Malimg dataset than the previous ones.

In our experiments, we chose LightGBM because of its better accuracy than any other boosting algorithm (Khandelwal, 2017). And also because of, its performance and effectiveness in Robust Intelligent Malware Detection as studied by (Abbadi et al., 2020). Similarly, we chose XceptionCNN because of its efficiency in extracting image features automatically, it is less time consuming and effective (Lo et al., 2019). Figure 5 shows our approach in accomplishing our enhanced LightGBM model.

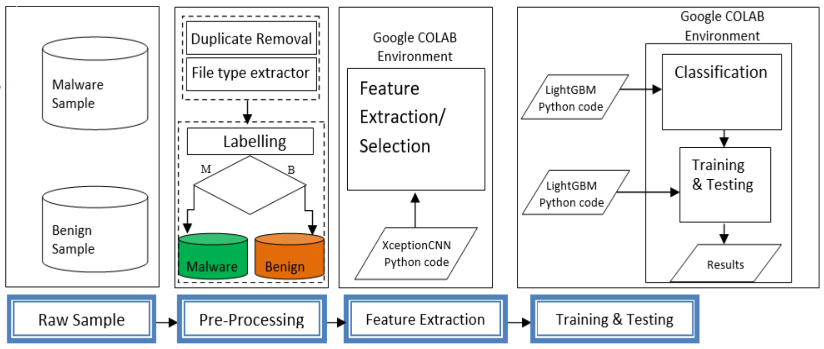

Figure 5: Design of the Proposed XceptionCNN-LightGBM model

3.8 Performance Evaluation

A Confusion matrix N x N (where N is the number of target classes) is used to evaluate the performance of the proposed model. A 2 x 2 matrix is required to perform binary classification. The confusion matrix is used to evaluate the ability of our model to classify the data based on its assigned labels. Each C_{mn} is the data instances which belong to group m (true label) and predicted belonging to group n (predicted label) (Harikrishnan, 2019). C_{00}, C_{01} and C_{10} denote the number of true negatives, false positives, and false negatives respectively.

The basic terminologies used for defining Confusion Matrix include (1) True Positive (TP), which refers to the point at which the predicted positive value matches the real value; (2) True Negative (TN), when the predicted negative value matches the real value; (3) False Positive (FP) - where the real value is negative but the model predicted positive (also referred to as Type 1 error); and (4) False Negative (FN) – a situation where the real value is positive but the model predicted negative (commonly referred to as Type 2 error).

3.8.1 Evaluation Metrics

Evaluation metrics assess the quality of machine learning model in order to obtain necessary feedback and determine its effectiveness and efficiency. We explain the metrics used in evaluating our models below :

i. **Accuracy:** The machine learning model accuracy for a given classification task is given as $\frac{NumberofCorrectPrediction}{TotalNumberofPredictionMade}$

$$\text{Accuracy} = \frac{TN+TP}{TN+FP+TP+FN} \tag{9}$$

Where, *TN, TP, FN*, and *FP* represent True Negative, True Positive, False Negative and False Positive data points respectively.

ii. **True Positive Rate**: True Positive Rate corresponds to the proportion of positive data points that are correctly predicted as positive, with respect to all positive data points.

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FN} \tag{10}$$

TP and *FN* are as described in (9).

iii. **False Positive Rate (FPR):** False Positive Rate corresponds to the proportion of negative data points that are mistakenly predicted as positive, with respect to all negative data points.

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN+FP} \tag{11}$$

TN and *FP* are as described in (9).

iv. **Precision:** used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class (Hossin, 2015). It is calculated as the ratio of the correct positive results to the number of positive results predicted by the classifier.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{12}$$

TP and *FP* are as described in (9).

v. **Recall -** used to measure the fraction of positive patterns that are correctly classified (Hossin, 2015). It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$\text{Recall} = \frac{TP}{TP+FN} \tag{13}$$

TP and *FN* are as described in (9)

vi. **F1 Score:** measures the accuracy and effectiveness of a classifier (Mishra, 2018). The value ranges between 0 and 1. A high F1 score indicates that the model has good performance. It is calculated as the Harmonic Mean of precision and recall. it can be expressed mathematically as:

$$\text{F1 Score} = 2 * \frac{\boldsymbol{Precision*Recall}}{\boldsymbol{Precision + Recall}} \tag{14}$$

vii. **Training Time (TT):** The training time of a model is the total amount of time required for a model to be completely trained. It is the difference between the end time and the start time of training the model.

$$\text{TT} = \text{EndTime(ET)} - \text{StartTime(ST)} \tag{15}$$

3.9 Implementation Environment

We conducted experiments on Google Co-laboratory (COLAB) environment with TPU v3, 32GB HMB. COLAB is a machine learning education and research platform based on Jupyter Notebook (Carneiro et al., 2018). It is pre-configured with necessary machine learning and artificial intelligence libraries, such as TensorFlow, Matplotlib, and Keras. It provides a CPU, GPU and TPU accelerated Python 2 and 3 runtime.

4 Results and Discussion

We performed two experiments involving the Malimg dataset, with binary/multi classification level, using hybrid (XceptionCNN - LightGBM) learning technique. In this section, each experiment will be discussed in detail and results will be presented for the two separate experiments. Each experiment represents the Malimg dataset, binary/multi classification level. The results from our preliminary LightGBM experiments on the Malimg malware dataset are presented in Table 2.

Table 2 presents the results (binary & multi-class) obtained from our preliminary experiment using LightGBM Algorithm**.**

Table 2: Results of LightGBM (Preliminary) Experiments

Classification / Metric	Binary	Multi-Class
Recall	99.80%	96.87%
Precision	99.80%	96.75%
F1_Score	99.80%	96.51%
Training Accuracy	99.80%	96.87%
Training Time	179.51s	2224.77s
Prediction time	0.08s	0.40s

For the binary classification results in Table 1, training time of 179.51s means the model spent a total of 179.51s for training. It obtained a training accuracy of 99.80%, a precision of 99.80% which is the positive patterns correctly predicted from the total predicted patterns in a positive class, and a recall of 99.80%, which means 99.80% fraction of positive patterns, were correctly classified. The Harmonic Mean between this precision and recall which is the F1_score, is 99.80%. The greater the F1 Score, the better the performance of the model. The corresponding multiclass model spent a total of 2224.77s of training time, with training accuracy of 96.87% and a precision of 96.75%, which is the positive patterns correctly predicted from the total predicted patterns in a positive class, with a recall of 96.87%, which implies 96.87% fraction of positive patterns were correctly classified. The Harmonic Mean between precision and recall which is the F1 Score is 96.51%. Similarly, the greater the F1 Score the better the performance of the model. The prediction time of 0.08s and 0.40s for binary and multi classification respectively show the time taken for predictions to occur.

4.1 Experiments on XceptionCNN – LightGBM

In these experiments, we improved the LightGBM model by hybridizing it with XceptionCNN. We installed LightGBM as an independent model on the colab notebook for implementation.

4.1.1 Binary Classification

Binary classification is used to distinguish malware from benign samples. We created the malware class by placing all Malimg families into one malware set. There are 1042 benign samples which are converted to images.

4.1.2 Multi-Classification

We also used the XceptionCNN – LightGBM algorithm to classify malware samples into distinct families. It is a multi-classification problem consisting of 26 classes, The actual Malimg dataset consist of 25 malware families, while the benign set is considered an additional "family" resulting in a total of 26 classes. We present the results obtained from our hybrid experiments in the Table 3.

Table 3: Results of XceptionCNN – LightGBM Experiments

Classification / Metric	Binary	Multi-Class
Recall	99.85%	97.40%
Precision	99.85%	97.29%
F1_Score	99.85%	97.29%
Training Accuracy	99.85%	97.40%
Training Time	29.97s	447.75s
Prediction time	0.08s	0.37s

Table 3 presents the results (binary & multi-class) obtained from our experiments using our hybrid model.

For the binary classification results in Table 3, training time of 29.97s means the model spent a total of 29.97s for training. It obtained a training accuracy of 99.85%, a precision of 99.85% which is the positive patterns correctly predicted from the total predicted pattern in a positive class and a recall of 99.85%, which means 99.85% fraction of positive pattern were correctly classified. The Harmonic Mean between this precision and recall, which is the

F1_score is 99.85%. The greater the F1 Score the better the performance of the model. The corresponding multi-class model spent a total of 447.75s training time, with training accuracy of 97.40% and a precision of 97.29% which is the positive patterns correctly predicted from the total predicted patterns in a positive class. A recall of 97.40% was obtained, which means 97.40% fraction of positive patterns were correctly classified. The Harmonic Mean between precision and recall which is the F1 Score is 97.29%. Similarly, the greater the F1 Score the better the performance of the model. The prediction time of 0.08s and 0.37s for binary and multi classification respectively show the time and how fast predictions occur.

4.2 Comparison of Results

The results in Table 3 are comparable to those obtained in our generic LightGBM preliminary experiment (Table 2) and serves to confirm our hybrid model implementation. Table 4 shows the results comparison.

Table 4: Results Comparison

	LightGBM (Preliminary Experiment)		Xception – LightGBM	
Classification / Metric	Binary	Multi-Class	Binary	Multi-Class
Recall	99.80%	96.87%	99.85%	97.40%
Precision	99.80%	96.75%	99.85%	97.29%
F1_Score	99.80%	96.51%	99.85%	97.29%
Training Accuracy	99.80%	96.87%	99.85%	97.40%
Training Time	179.51s	2224.77s	29.97s	447.75s
Prediction time	0.08s	0.40s	0.08s	0.37s

Table 4 presents the results obtained from the various training techniques used. This study shows that it is effective to combine pre-trained XceptionCNN model with LightGBM algorithm to improve detection and classification of windows malware. It leverages on the strengths and benefits of XceptionCNN (Shaheed & Zhang, 2022) and the LightGBM algorithm (Abbadi et al., 2020). Comparing the experimental results obtained in this study with the preliminary experiments, Table 4 shows that combining the pre-trained XceptionCNN model with LightGBM obtains better classification accuracy than applying the generic LightGBM algorithm. Results clearly indicate that extracting image features using XceptionCNN and performing classification using LightGBM provides the best performance for malware detection. In order to evaluate our model, we chose Accuracy, Precision, F1-score, Recall, Training time and Detection time as evaluation criteria. From the results of these experiments in Table 4, we can see that our model achieves a good performance result compared to the preliminary experiments based on the generic LightGBM algorithm. Although the detection time of the binary class looks the same with the LightGBM, figure 7 shows that our hybrid approach outperforms it in terms of detection accuracy of 100%. This is due to the use of pre-trained XceptionCNN model. XceptionCNN extracted fewer misleading features than the generic LightGBM. Hence, fewer misleading data improves modeling accuracy.

4.3 Confusion Matrix

We also present the confusion matrix for our experiments in Figure 6 and Figure 7, to show the improved classification accuracy of our XceptionCNN – LightGBM algorithm.

Figure 6 shows the classification accuracy of the generic LightGBM model. It shows a total data point of 2,077, which corresponds to the 20% (testing data) of the total dataset. The generic LightGBM model performed reasonably well in this malware classification, correctly identifying 99% of the malware samples (True Positive Rate of 99%).

The classification accuracy of the hybrid model is presented in Figure 7. It shows a total of 2,077 data points, which correspond to the 20% (testing data) of the total dataset. Our hybrid model shows an improved classification accuracy of 100% True Positive Rate. This means our model correctly identifies 100% samples that are truly malware.

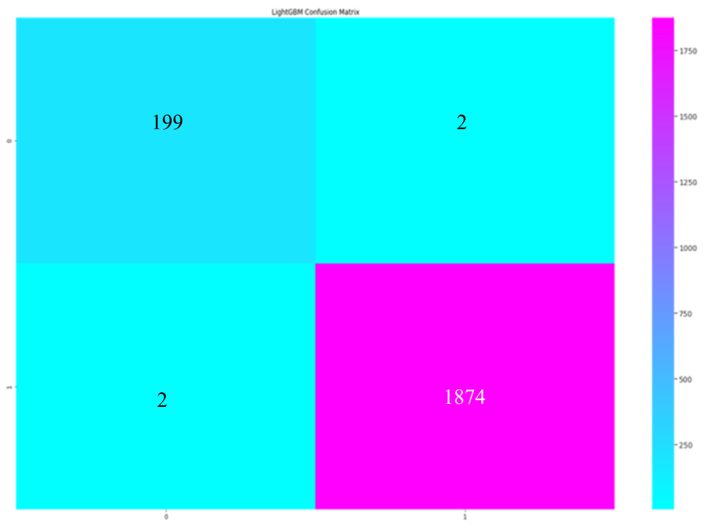

Figure 6: Generic LightGBM Confusion matrix

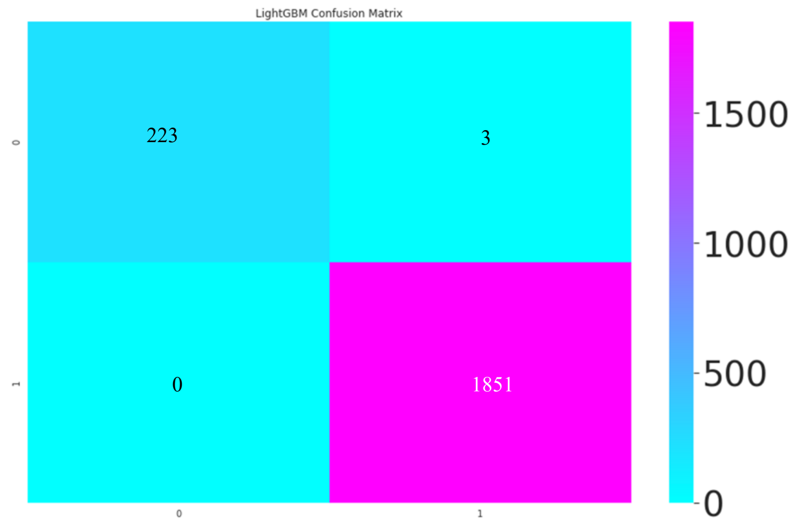

Figure 7: XceptionCNN –LightGBM Confusion Matrix

4.4 Comparison with Related Works

In Table 5, we compare our method with similar studies using the Malimg dataset for Windows Malware detection. Our results maintain higher accuracy than all the approaches in the related work. Our hybrid approach further maintains 100% TPR (see Figure 7).

Table 5: Comparison with Related Works.

Training Techniques	Classification	Training Accuracy
LightGBM (Preliminary Experiment)	Binary	99.80%
	Multi-Class	96.87%
XceptionCNN -LightGBM (Ours)	Binary	99.85%
	Multi-Class	97.40%
M-CNN (Kalash et al., 2018)	Binary	98.25%
DL (Bhodia et al., 2019)	Binary	98.39%
XceptionCNN (Lo et al., 2019)	Binary	99.37%
	Multi-Class	94.80%

The results as presented in Table 5 show that the XceptionCNN – LightGBM model is more effective and robust than previous solutions.

The XceptionCNN – LightGBM model accepts image data as input. The Malimg dataset is available publicly as benchmark Windows malware image dataset used in many studies for image based malware classification. Many machine learning and deep learning algorithms have been presented to develop models for effective malware detection. In this study, we combined XceptionCNN and LightGBM algorithms for binary and multi classification approach for effective and efficient malware detection. The proposed approach was compared with current methods in the literature. Table 5 summarizes the comparison of our performance results with current methods in the literature in terms of Training accuracy. Our model outperforms the state–of–the–art approaches with the best training accuracy. These significant improvements are attributed to the usage of the pre-trained XceptionCNN model and the LightGBM algorithm which aided in producing an excellent outcome. This is because XceptionCNN which is a pre-trained model for image based feature extraction, extracts the best image features needed for training the model by the LightGBM algorithm. The XceptionCNN model extracted adequate and less redundant image features from the dataset which were further trained by LightGBM. Less redundant data means fewer tendencies to make decision based on noise, reduced overfitting and improved robustness. Similarly, Quality training using the LightGBM algorithm produces a highly reliable model, resulting in fast and accurate classification.

5 Conclusion

This study proposed XceptionCNN – LightGBM model for Windows malware detection. The proposed hybrid technique addresses the limitations of the generic LightGBM algorithm in terms of classification accuracy, prediction time, training accuracy and training time. The model was tested on 9,339 malware samples across 25 malware families and 1,042 benign samples. Preliminary experiments based on the generic LightGBM algorithm show a classification accuracy of 99% TPR with prediction time of 0.08s and 0.40s for binary and multi classification respectively. Experimental results show that the hybrid technique provides improved classification accuracy of 100% and reduced prediction time of 0.08s and 0.37s for binary and multi classification respectively. The training accuracy improved by 0.5% and achieved a reduced training time of 29.97s from 179.51s for binary classification and 447.75s from 2224.77s for multi classification. The practical implication of this study is that the hybrid approach provides accurate and reliable detection of malicious software that attack computer systems and compromise the confidentiality, integrity and availability of information stored in them. The reduction in detection time provides early detection of a malware before it causes significant damage to files stored in computer systems. This minimizes the losses an organization will suffer in case of malware attack. The reduction in the training time enables the model to converge quickly and train a large amount of data in a relatively short period of time. A future study will consider using two or more malware datasets to conduct the experiments. This will further enhance the validity and reliability of the proposed model.

References

Abbadi, M. A., Al-Bustanji, A. M., & Al-kasassbeh, M. (2020, April 30). Robust Intelligent Malware Detection using LightGBM Algorithm. *International Journal of Innovative Technology and Exploring Engineering*, *9*(6), 1253–1260. https://doi.org/10.35940/ijitee.f4043.049620

Abusitta, A., Li, M. Q., & Fung, B. C. (2021). Malware Classification and Composition Analysis: A Survey of Recent Developments. *Journal of Information Security and Applications*, *59*, 102828. https://doi.org/10.1016/j.jisa.2021.102828

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013). A Survey on Heuristic Malware Detection Techniques. *The 5th Conference on Information and Knowledge Technology (*pp. 113-120*).* https://doi.org/10.1109/ikt.2013.6620049

Bensaoud, A., & Kalita, J. (2022). Deep Multi-task Learning for Malware Image Classification. *Journal of Information Security and Applications*, *64*, 103057. https://doi.org/10.1016/j.jisa.2021.103057

Bhodia, N., Prajapati, P., Di Troia, F., & Stamp, M. (2019). Transfer Learning for Image-based Malware Classification. *Proceedings of the 5th International Conference on Information Systems Security and Privacy* (pp 719-726). https://doi.org/10.5220/0007701407190726

Carneiro, T., Nobrega, R. V., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685. https://doi.org/10.1109/access.2018.2874767

Chang, J., Venkatasubramanian, K. K., West, A. G., & Lee, I. (2013). Analyzing and Defending against Web-based Malware. *ACM Computing Surveys*, *45*(4), 1–35. https://doi.org/10.1145/2501654.2501663

Chen, J., Guo, S., Ma, X., Li, H., Guo, J., Chen, M., & Pan, Z. (2020). SLAM: A Malware Detection Method Based on Sliding Local Attention Mechanism. *Security and Communication Networks*, 1–11. https://doi.org/10.1155/2020/6724513

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* (pp.1251-1258). https://doi.org/10.1109/cvpr.2017.195

Damodaran, A., Troia, F. D., Visaggio, C. A., Austin, T. H., & Stamp, M. (2015). A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. *Journal of Computer Virology and Hacking Techniques*, *13*(1), 1–12. https://doi.org/10.1007/s11416-015-0261-z

Fang, Z., Wang, J., Geng, J., & Kan, X. (2019). Feature Selection for Malware Detection Based on Reinforcement Learning. *IEEE Access*, *7*, 176177–176187. https://doi.org/10.1109/access.2019.2957429

Fonseca, E., Gong, R., Bogdanov, D., Slizovskaia, O., Gomez, E., & Serra, X. (2017). Acoustic Scene Classification by Ensembling Gradient Boosting Machine and Convolutional Neural Networks. In Virtanen,

T., Mesaros, A., Heittola, T., Diment, A., Vincent, E., Benetos, E., Martinez B. (Eds). Detection and Classification of Acoustic Scenes and Events 2017 Workshop: Tampere University of Technology (pp.37-41). http://hdl.handle.net/10230/33454

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of Machine Learning for Detection and Classification of Malware: Research developments, Trends and Challenges. *Journal of Network and Computer Applications*, *153*, 102526. https://doi.org/10.1016/j.jnca.2019.102526

Harikrishnan, B. (2019, December 10). *Confusion Matrix, Accuracy, Precision, Recall, F1 Score Binary Classification Metric*. National Institute of Advanced Studies, Bengaluru, India. https://medium.com/@harikrishnannb

Hossin, M., & Sulaiman, N. (2015). A Review on Evaluation Metrics for Data Classification. *International Journal of Data Mining & Knowledge Management Process*, *5(2),* 01-11. https://dio.org/10.5121/ijdkp.2015.5201

Huang, K. (2020). An Optimized LightGBM Model for Fraud Detection. *Journal of Physics: Conference Series*, *1651*(1), 012111. https://doi.org/10.1088/1742-6596/1651/1/012111

Hussain, S. J., Ahmed, U., Liaquat, H., Mir, S., Jhanjhi, N., & Humayun, M. (2019). IMIAD: Intelligent Malware Identification for Android Platform. *2019 International Conference on Computer and Information Sciences (ICCIS)*. (pp. 1- 6). https://doi.org/10.1109/iccisci.2019.8716471

Ju, Y., Sun, G., Chen, Q., Zhang, M., Zhu, H., & Rehman, M. U. (2019). A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting. *IEEE Access*, *7*, 28309–28318. https://doi.org/10.1109/access.2019.2901920

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., & Iqbal, F. (2018). Malware Classification with Deep Convolutional Neural Networks. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. https://doi.org/10.1109/ntms.2018.8328749

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st International Conference On Neural Information Processing Systems*, (pp. 3149–3157). https://dl.acm.org/doi/10.5555/3294996.3295074

Khandelwal, P. (2017, June 12). *Which Algorithm takes the crown: LightGBM vs XGBOOST?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-

Kumar, A. (2017). *A Frame work for Malware Detection with Static Features using Machine Learning Algorithms*. [Doctoral thesis] Soongsil University. https://doi.org/10.13140/RG.2.2.35593.90723

Landage, J., & Wankhade, P. (2013). Malware and Malware Detection Techniques: A Survey. *International Journal of Engineering Research & Technology*, *2(12),* 61 - 68. https://doi.org/ 10.17577/IJERTV2IS120163

Lo, W. W., Yang, X., & Wang, Y. (2019). An Xception Convolutional Neural Network for Malware Classification with Transfer Learning. *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (pp.1-5). https://doi.org/10.1109/ntms.2019.8763852

Machado, M. R., Karray, S., & Sousa, I. T. (2019). LightGBM: an Effective Decision Tree Gradient Boosting Method to Predict Customer Loyalty in the Finance Industry. *2019 14th International Conference on Computer Science & Amp; Education (ICCSE)* (pp. 1111-1116). https://doi.org/10.1109/iccse.2019.8845529

Malith, O. (n.d.). *A Simple Utility to Convert EXE Files to PNG Images and Vice Versa*. Github. Retrieved from http://github.com/OsandaMalith/Exe2Image

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing, 17(3), 12–22. https://doi.org/10.1109/mprv.2018.03367731

Microsoft Cooperation. (2021). Read the Docs, *LightGBM Release 3.2.1.99*. Github. Retrieved from https://lightgbm.readthedocs.io/

Minastireanu, E. A., & Mesnita, G. (2019). LightGBM Machine Learning Algorithm to Online Click Fraud Detection. *Journal of Information Assurance &Amp; Cybersecurity*, *12,* 1–12. https://doi.org/10.5171/2019.263928

Mishra, A. (2018, February 24). *Metrics to Evaluate your Machine Learning Algorithm.* Towards Data Science. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. (2011). Malware Images: Visualization and Automatic Classification. *8th* International *Symposium on Visualization for Cyber Security* 2011 (pp.1–7). https://doi.org/10.1145/2016904.2016908

Nawaz, A. (2021). Feature Engineering based on Hybrid Features for Malware Detection over Android Framework. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(10), 2856–2864. https://doi.org/10.17762/turcomat.v12i10.4931

Pan, Q., Tang, W., & Yao, S. (2020). The Application of LightGBM in Microsoft Malware Detection. *Journal of Physics: Conference Series*, *1684*(1), 012041. https://doi.org/10.1088/1742-6596/1684/1/012041

Pant, D., & Bista, R. (2021b). Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning. *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*. https://doi.org/10.1145/3503047.3503081

Şahin, D. Z., Kural, O. E., Akleylek, S., & Kılıç, E. (2021). A Novel Permission-based Android Malware Detection System using Feature Selection based on Linear Regression. *Neural Computing and Applications*, 33, 1 – 16. https://doi.org/10.1007/s00521-021-05875-1

Shaheed, K., Mao, A., Qureshi, I., Kumar, M., Hussain, S., Ullah, I., & Zhang, X. (2022). DS-CNN: A pre-trained Xception Model based on Depth-Wise Separable Convolutional Neural Network for Finger Vein Recognition. *Expert Systems With Applications*, *191*, 116288. https://doi.org/10.1016/j.eswa.2021.116288

Sharma, A. (2018, October 15). *Understanding GOSS and EFB: The core Pillars of LightGBM.* Towards Data Science. https://towardsdatascience.com/what-makes-lightgbm-lightning-fast-a27cf0d9785e

Singh, J., & Singh, J. (2021). A Survey on Machine Learning-based Malware Detection in Executable Files. *Journal of Systems Architecture*, *112*, 101861. https://doi.org/10.1016/j.sysarc.2020.101861

Su, J., Vargas, V. D., Prasad, S., Daniele, S., Feng, Y., & Sakurai, K. (2018). Lightweight Classification of IoT Malware Based on Image Recognition. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (pp. 664 - 669). https://doi.org/10.1109/compsac.2018.10315

Sun, X., Liu, M., & Sima, Z. (2020). A Novel Cryptocurrency Price Trend Forecasting Model Based on LightGBM. *Finance Research Letters*, *32*, 101084. https://doi.org/10.1016/j.frl.2018.12.032

Venkat, T., Rao, N., Unnisa, A., & Sreni, K. (2020). Medicine Recommendation System based on Patient Reviews. *International journal of Scientific & Technology research, 9(2)*, 3308 - 3312.

Wang, J. (2018). *Detection and Analysis of Web-based Malware and Vulnerability* [Doctoral thesis]. Nanyang Technological University, Singapore. https://doi.org/10.32657/10220/47659

Wong, M. Y., Landen, M., Antonakakis, M., Blough, M. D., Redmiles, M. E., & Ahamad, M. (2021). An inside look into the practice of Malware Analysis**.** *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 3053–3069). ACM SIGSAC. https://doi.org/10.1145/3460120.3484759