SOCIAL ENGINEERING EXPLOITATION DETECTION (SEED) IN MALAYSIA'S SMES USING MACHINE LEARNING

BY

SIA KEN YEN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: SOCIAL ENGINEERING EXPLOITATION DETECTION (SEED) IN MALAYSIA'S SMES USING MACHINE LEARNING

Academic Session: __JAN2022____

I _____SIA KEN YEN_____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

_____

(Author's signature)

Address:
26, Jalan Perdana 2/22,
Taman Bukit Perdana 83000
Batu Pahat, Johor

Date: 21.04.2022

Verified by,

_____

(Supervisor's signature)

Vasaki Ponnusamy

Supervisor's name

Date: 22/04/22

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 21.04.2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ***SIA KEN YEN*** (ID No: ***18ACB04562***) has completed this final year project entitled "*SOCIAL ENGINEERING EXPLOITATION DETECTION (SEED) IN MALAYSIA'S SMES USING MACHINE LEARNING*" under the supervision of TS_DR VASAKI A/P PONNUSAMY (Supervisor) from the Department of Computer and Communication, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

(*SIA KEN YEN*)

DECLARATION OF ORIGINALITY

I declare that this report entitled "**SOCIAL ENGINEERING EXPLOITATION DETECTION (SEED) IN MALAYSIA'S SMES USING MACHINE LEARNING**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : SIA KEN YEN

Date : 21.04.2022

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Ts Dr Vasaki a/p Ponnusamy and my moderator Dr Ashvaany a/p Egambaram who has given me this bright opportunity to engage in this social engineering attack detection project. This project will be the foundation in establishing my career in the cybersecurity field.

Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course. I would also like to express my appreciation to my friends who had been giving me different suggestions in making this project.

ABSTRACT

Over the past years, computer security has been a field of study that assists in protecting one's information. It has matured over time in fighting against cybercrime in exploiting the technical vulnerabilities of hardware or software. However, there is a kind of attack that particularly exploits the human psychological weakness in acquiring confidential information is emerging which is called social engineering. It has lesser cost and branches to many variations of type of attacks than the traditional technical approach which challenges organization's protection such as SMEs. Most of the current detection models only provide a guideline framework in detecting such attacks which is not efficient or with low accuracy. This project aims at building a model that is based on another popular field which is machine learning in detecting attacks. This can be applied to flag a conversation as if it is a social engineering attack. The project will use natural language processing in extracting certain features as the input of an algorithm to generate a reputation score that will be trained using machine learning to build the detection model. The model will be evaluated and validated using datasets by generating the result scores.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number **Title** **Page**

LIST OF TABLES

Table Number **Title** **Page**

LIST OF ABBREVIATIONS

SE	Social Engineering
TBL	Topic Blacklist
SEDM	Social Engineering Detection Model
SEDA	Social Engineering Defense Architecture
NLP	Natural Language Processing
SME	Small Medium Enterprise
SVM	Support Machine Vector
VT	Virus Total
WOT	Web of Trust
ML	Machine Learning
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversampling Technique
AUC	Area Under Curve
SVM	Support Vector Machine
KNN	K Nearest Neighbours
SGD	Stochastic Gradient Descent

CHAPTER 1: INTRODUCTION

1.1 Problem Statement and Motivation

In recent years, SMEs are becoming more attractive and vulnerable to social engineering attacks due to the lack of awareness. Many SMEs have changed or set up their business on the internet because of the advancement in technology. This makes them the gate to sensitive online customer data which has been in the interest of cybercriminals for malicious purposes. Such cases have been proven true especially during the Covid-19 pandemic where it is reported that cybercriminals are exploiting the fear and uncertainty of the situation to deploy social engineering attacks on organizations including SMEs. A successful attack will cause a potential breach on the system which can be exploited in a bigger event, causing huge losses in SMEs. However, as most social engineering attacks differ from each other, it is difficult to be able to identify them. In addition, the problem arises as to successfully detect such attacks whilst working in a stressful environment where decisions are required to be made instantaneously stated in [1]. For example, employees may accidentally give out confidential information while handling a large scale of emails. The increasing number of social engineering attacks had actually raised attention in the cybersecurity field. Many researchers are beginning to explore and develop a framework or model for countering such attacks. The main prevention technique implemented in companies is to increase awareness of social engineering attacks by encouraging security education and training. Other methods such as verifying the source of calls to prevent vishing attacks, installing anti-phishing tools to block phishing websites and even a decision-making model based on human reasoning. Although these are the viable options on counter measuring attacks, it is still subjected to the human factor itself. Social engineering deals with human psychological exploitation where an experienced attacker will always take into account all plausible factors as long as the human element is present. Human error will still undoubtedly happen in a longer timescale thus increasing the risk of data leakage. Therefore, a new approach in analyzing the properties of the email content solely and its data text which is machine learning is proposed. This gives additional assistance and references for employees on social engineering attacks detection.

1.2 Project Objectives

Main objective:

- To determine whether dialogue between two ends is a social engineering attack.

Sub-objectives:

- To use natural language processing technique to perform data processing on the textual context in dialogues.
- To find the features of the dialogue itself that could lead to a potential attack and the extraction of it for further processing.
- To choose the best machine learning algorithm for the model.
- To validate the model by generating the result score of each technique.
- To evaluate awareness on social engineering among SME employees
- To educate SMEs employees on social engineering

1.3 Project Scope

The final product of this project will be a model of a system to detect exploitation of social engineering using machine learning and simulation results for validation. The machine learning techniques used are to resolve the analysis of the high volume of emails that needed to be processed. A trained model can be repeatedly and openly used for each individual as guidance in the detection of an attack. Multiple machine learning techniques are used to evaluate each of their performance. The natural processing language module is integrated into the model for a modern approach to processing text on a computer. The model also encompasses an algorithm in which a certain trust score based on various important features of an email produces a more extensive result to evaluate more accurately the requester intent.

1.4 Contribution

This project aims to develop a social engineering attack and exploitation detection model which serves as a basic framework that would help Malaysia's SMEs employees in identifying potential attacks. It is based on existing works with the integration of preliminary research and survey conducted among SME employees. The survey also evaluates their awareness of social engineering attack and educate them on this topic. The model will use a machine learning approach with another technique called Natural Language Processing (NLP) for processing the text content of emails. This model focuses on the analysis of specific words and generates a score based on an algorithm. Each score will then be trained and tested using machine learning. The model will then be evaluated using datasets to validate the approach.

1.6 Report organization

This paper consists of 7 chapters and is organized as follows: Chapter 1 is the introduction which covers the problem statement and motivation, project scope and project objective, contribution and background information. Chapter 2 is about the related studies of existing works with their respective advantages and disadvantages. Chapter 3 explains the project methodology, project flow diagram and specifications needed. Chapter 4 will discuss each part of the system design and encompasses a research from various sources. Chapter 5 would be the system experiment that includes the software setup with setting and configuration and implementation details of the model. It also covers the survey result and analysis as part of this research. Chapter 6 is about the evaluation of the model which result analysis of various performance metrics will be discussed. Challenges that were faced during the project, testing result and objectives evaluation are also mentioned in this chapter. Chapter 7 is the conclusion which summarizes the project and final result as well as recommendations for future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Previous works on social engineering detection

2.1.1 Automatic Detection of Social Engineering Attacks Using Dialog

Kale et al. [2] emphasize the research of dialogue-based attacks due to their effectiveness. It has given an approach to the detection of social engineering by the usage of semantic analysis of all the data text sent to the victims. In layman's terms, semantic analysis is the process of addressing and understanding the meaning of the text. In this case, the assumption is being carried out at the topics on each line. A statement will be considered inappropriate if it requests for secure information or unauthorized requests of secure operation. The appropriateness of each topic in the context is evaluated based on the presence of a Topic Blacklist (TBL). TBL is a vital component of the entire detection model in which it describes forbidden topics that are essentially labelled as blacklist words. It has derived a more dynamic nature in which new topics are added into the database automatically based on common security requirements or existing security policy documents in association with the system. Finally, a warning message will be prompt to the victim to alert about the detection of an attack.

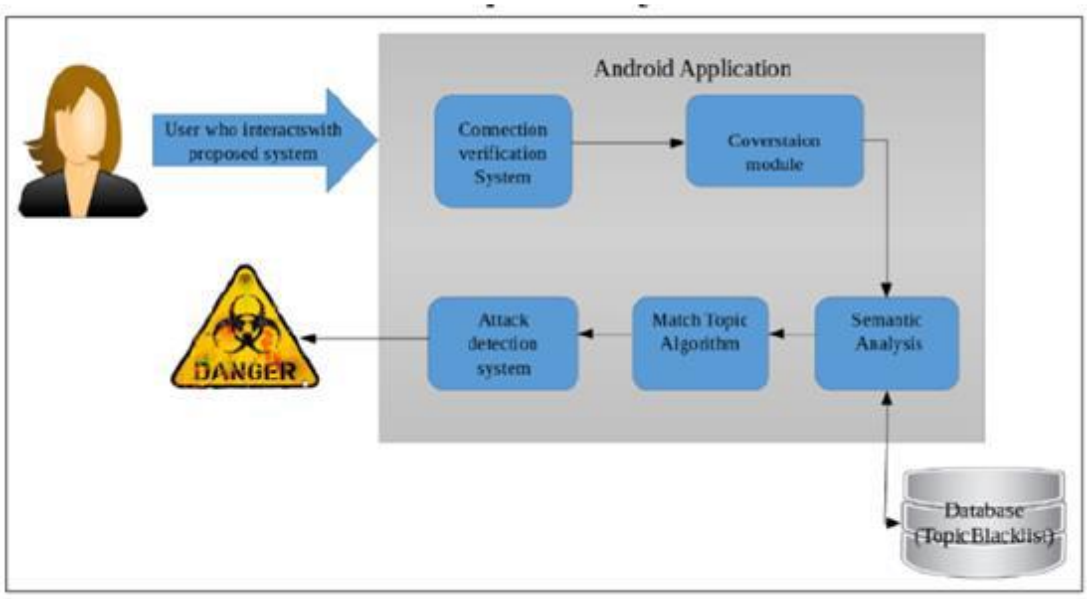


Figure 2.1 Proposed system using semantic analysis of dialogs

Figure 2.1 shows the proposed system using semantic analysis of dialogues in automatic detection of social engineering. The user will first interact with the proposed

system and establish a connection with the verification system. Then, the conversation module is used similar to normal dialogues. Each line will scan through the dialogue using semantic analysis with the TBL database which is done by a detection algorithm. In that detection algorithm, there exists a match topic algorithm function which checks each entry of every line of the dialogue to compare the topics with the blacklist of words in the database.

ACTION	RESOURCE
Send	Money
Send	Sensitive Data
Call	Number
Visit	Website

Figure 2.2 Topic blacklist (TBL)

If the word matches, the attack detection system will alert the user of the potential attack.

2.1.2 Social Engineering Attack Detection Model (*SEADM*)

In this paper, Bezuidenhout et al. [1] proposed a different model that takes on two main perspectives of social engineering which are the psychological perspective and the computer science perspective. The former focuses on the emotional state and cognitive abilities of the individual while the latter focuses on information sensitivity. The model also includes factors such as the urgency of the requested information and an individual's comprehension of the requested information. The model in this paper encompasses social engineering attacks with psychological aspects. It claims that psychological triggers play a vital role in a social engineering attack and therefore important to recognize them. Seven psychological vulnerabilities have been stated which are strong effect, overloading, reciprocation, deceptive relationship, diffusion of responsibility and moral duty, authority, integrity and consistency. Victims are to experience a sense of discomfort when encountering these triggers that set up a red flag of a potential attack. However, this will be an ideal case but not realistic on a day-to-day basis due to the complexity of human reasoning and the decision-making process. It is found that individuals find it difficult to make rational decisions in a limited time frame which is the reason why a predefined set of guidelines is needed.

The proposed practical application model on social engineering detection makes use of a decision tree in which breaks down the process into components for ease of management and aid to decision making. The SEADM model is shown in figure 2.3.

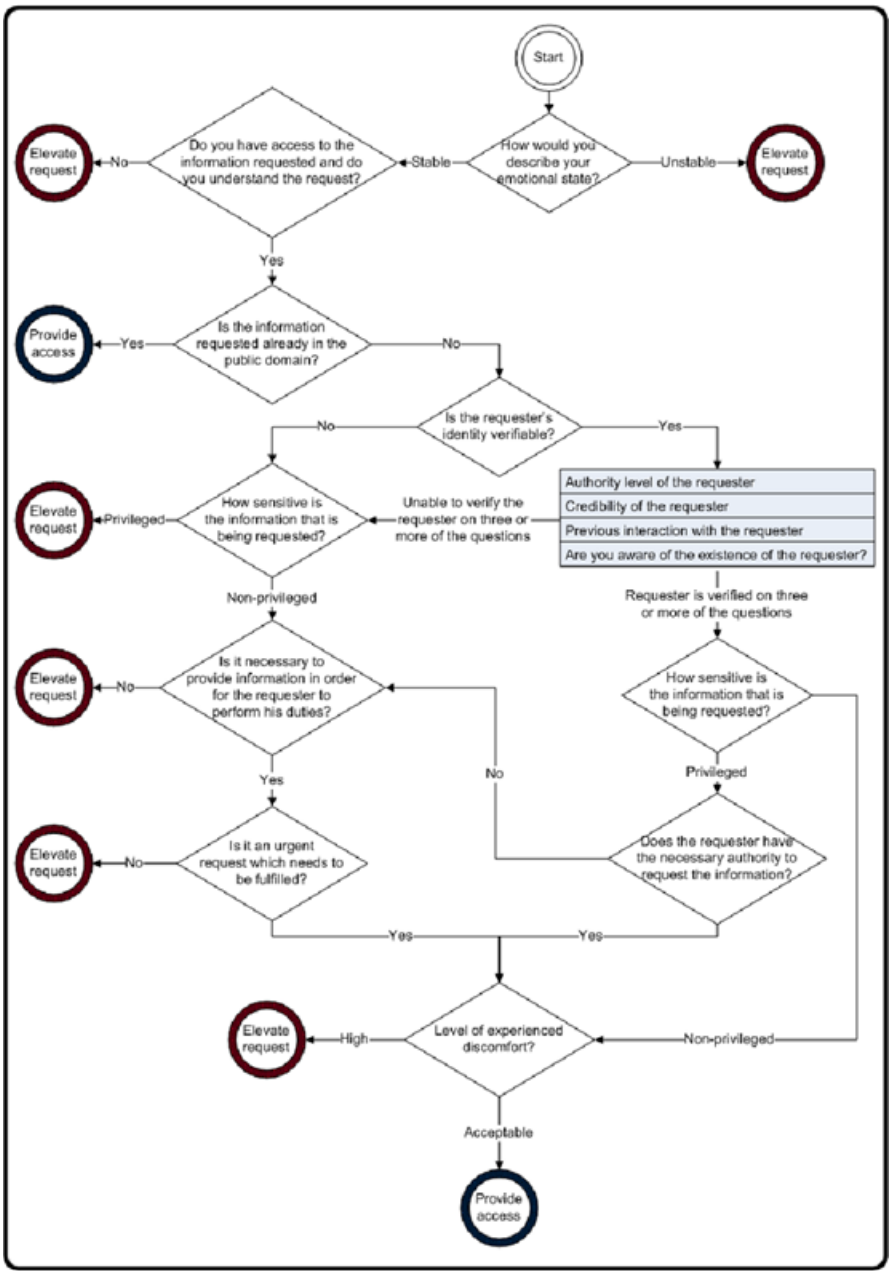


Figure 2.3 Social Engineering Detection Model

On a side note that the term individual refers to the person dealing with an incoming call and the term sequester refers to the person making the call and requesting information in the following discussion. In summary, the individual first needs to evaluate the emotional state of both themselves and the requester, on an ongoing basis. A negative emotional state will influence one's judgement on making decisions. The individual then assesses the accessibility and knowledge of the information requested. If not, they are to pass it to another individual who will also follow the same model. Individuals should understand if the information is already in the public domain. Some institutions have open files on their working hours, profit margin, key shareholders while others have strict registration. Next, individuals are to verify the requester's identity based on four criteria which are authority level of the requester, credibility of the requester, previous interaction with the requester and the awareness of the existence of the requester. Further on, individuals need to identify the sensitivity of the information requested. Information is divided into two categories in this model, privileged information that needs authorization and non-privileged information that does not. Then, the individual will need to determine if the requester possesses the necessary authority to request the information. If requesters do in fact have the same or higher authority level for the particular information, they will go straight to the final step, otherwise, an alternative route will be directed. The requester will be questioned on their necessity on the information requested in performing their duties to make sure that it would indeed be beneficial for both parties involved. Moreover, individuals will also need to consider the urgency of the request. In the presence of time leniency, a higher-up can be consulted but in contrast, which may lead to a life and death situation, the final step is considered. Finally, the evaluation of one's emotion is again tested with the level of experienced discomfort after all those cautious steps. If the accumulated sceptics are too much, it is suggested to reject the requester. If the level of discomfort is understandable and acceptable, the information can only then be provided.

The paper also included scenarios whereby proven the model is indeed feasible as a preventative measure against social engineering attacks.

2.1.3 Social Engineering Defense Architecture (SEDA)

Social Engineering Defense Architecture (SEDA) is another proposed method in [3] which detects social engineering attacks perpetrated over telephones. It is designed based on the intent and deception of the attacker. The main purpose of SEDA is to increase the awareness of individuals within organizations on callers who are trying to make a social engineering attack by deception in order to obtain unauthorized information. The core idea of the system is a text-independent voice signature authentication system that is able to examine the unique speech of an individual.

The voice signatures collected by SEDA are linked to the organization's database which stores personal information such as employee name, corporate association, job title and phone numbers. The extent of data collected varies due to the policy of organizations.

The SEDA's strategy works in countering the impersonation of attackers by utilizing the voice signature authentication element. The requester will not be able to penetrate the first layer of an organization even if other credentials are matched in a phone conversation. The bypass solution would be changing the voice but it may be picked up by other detection systems of SEDA. Forensic investigation is also a component that falls under this architecture. Therefore, SEDA incorporates a voice-to-text engine that converts the recorded voice conversations into text. However, an opt-in process is required in order to be in line with the individual privacy policy and to follow the wiretap laws. Figure 2.4 shows the SEDA's decision tree structure.

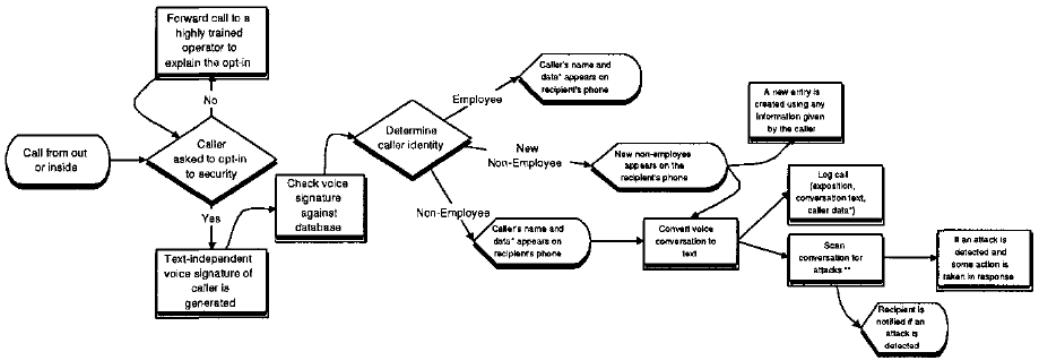


Figure 2.4 Social Engineering Defense Architecture decision tree

A content analysis tool will then be used for a simple analysis of the conversation text itself. This is to evaluate the malicious intent or violation of the security policy of the caller. It is done by scanning for attack phrases, NLP attack

detection and data-based attack detection. The final result of the analysis will determine if an attack is present. Figure 2.5 presents an expanded view of the attack detection process.

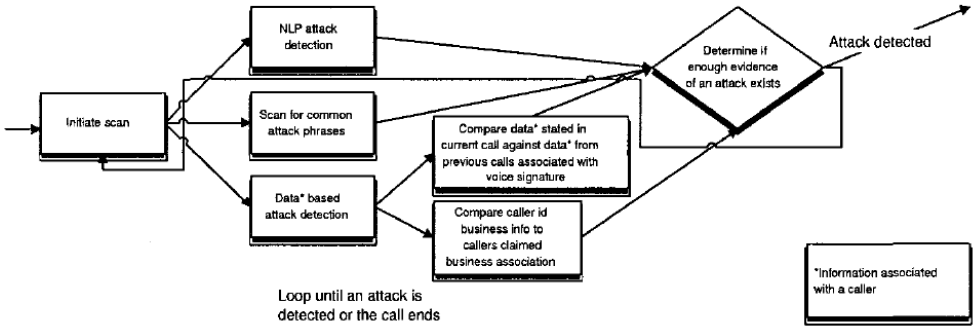


Figure 2.5 Attack detection process

2.1.4 SEADer: A Social Engineering Attack Detection method

Lansley et al. [4] introduce a distinct approach by introducing the integration of machine learning algorithms into the detection model. The core concept of this method is to parse the conversation text to check for grammatical errors with the usage of natural language processing technique follow by an example of a machine learning method called artificial neural network to perform classification on possible attacks.

This paper is a piece of additional evidence on the demonstration of the usage of a variety of different techniques such as Natural Language Processing (NLP) and Machine Learning (ML) methods in the research of an automated system for the recognition of social engineering attacks. It also takes in the consideration of the human psychology which is the fundamental principle in social engineering. The behavior and social aspects of people are the keys to determining the rate of success of an attack. By citing the work of Dr. Robert Cialdini, some key features are used to be the factors in building the foundation of the model in which are: authority, Reciprocation, Liking/Similarity, Scarcity, Commitment/Consistency and Social Proof. These traits have been agreed by psychologists to be the aspects in the definition of a persuasive person.

This paper proposed a three stage process on the research starting with data preprocessing, feature extraction and aggregation of results in sequence. Following the general goal of preparing the data for classification, most models capture the contextual or meta data for example time, date and IP addresses in relation to the original data to spot reoccurring adversaries possibly in different aliases. Erroneous content like HTML tags, corrupt texts or headings are removed to sanitize the dialogue. With the usage of the vast functionality of the NLP parsers library, programmers are able to use linguistic features extracted in classifying conversations. This paper focuses on three main criterions when comes to choosing features to determine whether social engineering attack has taken place in a certain dialog which are the principles of persuasion, the history of the attacker and usual phishing tactics.

After that, the outputs from the extraction process are aggregated for clear display. In order to increase the precision and leverage the output, the features are

weighted by the importance of the average overall result and generate a Fuzzy logic prediction. This can be further enhanced with other more advanced techniques are integrated into the equation. The proposed method in this paper undergoes a series of steps and algorithms to generate a dataset by preprocessing the dialogs.

Three machine learning algorithms are used, each executed 10 times after performing 5-fold cross validation on the datasets. Several tables are prepared for visualization with maximum value, minimum value and median value as labels of the 10 iteration results. There is also specification on those algorithms to fine-tune the best result. Decision Tree with the default setting from the SciKit is used which included unlimited leaf nodes. The parameter value on estimators of SciKit Random Forest classifier is set to 50. A parameter setting of MLP classifier with lbfgs solver, alpha = 1e-5 and hidden layer size of (15, 15, 15) are applied to the Neural Network.

The final experimental evaluation result on this method indicates to be successful as it could detect social engineering attacks with high accuracy. This is further supported with the comparison of alternative classification methods where its effectiveness is solidified.

2.1.5 Framework for Detection of Phishing Social Engineering Attacks

Balaji et al. [5] provide a similar approach with the integration of machine learning and SE detection by building a basic and simple framework. The paper revolved around the human psychology principle as it is viewed to be the foundation of social engineering attacks. According to [6], this paper concurred with him on the statement that the most common type of attack that leverages social engineering methods are phishing attacks. There is three main endeavors in most phishing scams which are obtaining personal information, the usage of shortened or misleading links that redirect end users to malicious websites and the manipulation of the receiver to panic and respond illogically with the pressure of incorporate threat, fear and a sense for urgency.

The framework proposed by this paper, the classifier approach is a classic machine learning model process that typically follows the same stages. In this case, the purpose is to differentiate spam or phishing emails from ham or legitimate emails. The first step is data acquisition or data collection which is the process of obtaining email datasets from various sources which consist of spam and ham email records. Next, these data are converted into electronic mail format. In order to filter the words in EML format of these emails, the process of tokenization is taken place to perform preprocessing.

After that, the subsequent stage will be feature extraction where some features on a specific email example are selected for processing. These features attributes are elements such as body of the email, header, URL, sender address field, receiver address field, cc and bcc fields. The resultant files of those features will be converted into ARFF format. The dataset is separated into training set and testing set with a ratio of 7:3.

Three classifier techniques are used in this instance to validate the performance of the dataset which are Naïve Bayes, J48 and Bayses Net ans J48. The final performance observation is presented with a table with several labels explaining the experimental results on the measurement of the effectiveness of the model. The label components are TPR, TNR, FPR, FNR, accuracy, precision, sensitivity and f-measure and respective values are shown in figure 2.6. The more important criteria of those labels are TPR, TNR and accuracy. TPR and TNR are calculated to evaluate the

robustness of the proposed model. Accuracy is the measurement of the model performance.

EXPERIMENT	BAYESNET (%)	J48 (%)	NAVIE-BAYES (%)
TPR	96.15	96.7	96.65
TNR	98.5	99.41	98.$65
FPR	0.4	0.53	0.25
FNR	2.75	3.25	1.35
ACCURACY	97.32	98.1	97.65
PRECISION	98.49	99.46	98.64
SENSITIVITY	96.15	96.75	96.65
F-MEASURE	97.31	98.08	97.64

Figure 2.6 Confusion matrix with different classification technique results

The model achieved very high accuracy on the detection of spam messages with 15 features on 1000 records. This is solid proof of the effectiveness of integrating machine learning to SE attacks detection. However, it is still a basic framework to be further optimized and fine-tuned by future scholars.

2.2 Summary and Critical Remarks of Previous Works

After reviewing the related works on social engineering detection, various approaches and methods have been studied thoroughly. Different detection models have been proposed and some of them provide new insight into the problem. Despite that, these models are limited in some particular way respectively where they could be optimized better. In the first paper, the model is too generic in a sense where it could not effectively identify an attack and could cause many false-negative cases for its list-based approach. Some legitimate actions might have been marked as malicious just due to the fact of certain blacklisted words. The second paper provides a more in-depth breakdown in increasing the accuracy by having a decision tree-based method rather than just semantic analysis. However, this SEDAM model discussed has a vulnerability in human error where it will be troublesome and ineffective for individuals to follow a long list of guidelines for each email they received. The third paper presented another detection architecture that focuses on voice authentication of receivers in conversation but it misses out on the textual conversion analysis on the content of each conversation. Moreover, since the detection module is based on certain phrases and does not take into account other criteria, it possesses the same problem in misjudging genuine requests. The fourth paper overcomes the limitation of the previous paper with the integration of machine learning and data normalization. This can be a useful guideline for the model implementation in this project. The downfall of it is the fact that there is not enough analysis on the result and some machine learning algorithm used is too complicated. It is also unclear on whether this model fits on different regions. The fifth paper is another paper that also includes the usage of ML but the performance is rather too high that leading to suspicion on possible overfitting data. It also did not have an in-depth analysis of the body of the email which is the main essence of social engineering.

This project aims to propose a social engineering detection model using machine learning algorithms to utilize past records of attack examples by analyzing their content. It will also be a basic framework for an automated detection system where it assists individual employees in SMEs where hundreds of emails are being processed. The proposed model will also solve the absence of a textual conversion tool by having an algorithm that would integrate each aspect in the email context that would decrease the false-negative rate of attack detection. Furthermore, the project will compact

analysis on the text body based on study and research of the Malaysia's SMEs sector while also comparing and analyzing on different machine learning techniques.

CHAPTER 3: SYSTEM MODEL

3.1 System model

3.1.1 Methodology and General Work Procedures

The processes of the project were categorized into different phases, which were problem analysis, preliminary research, survey development, data collection, data pre-processing, feature extraction and data normalization. The model training architecture building falls under model classification which will be discussed in depth in the next chapter. Preliminary research is about the facts finding from various sources and survey development include forming the questionnaires, survey distribution and result analysis that will be used in machine learning model with the integration of existing novelty ideas.

3.1.2 Project flow diagram

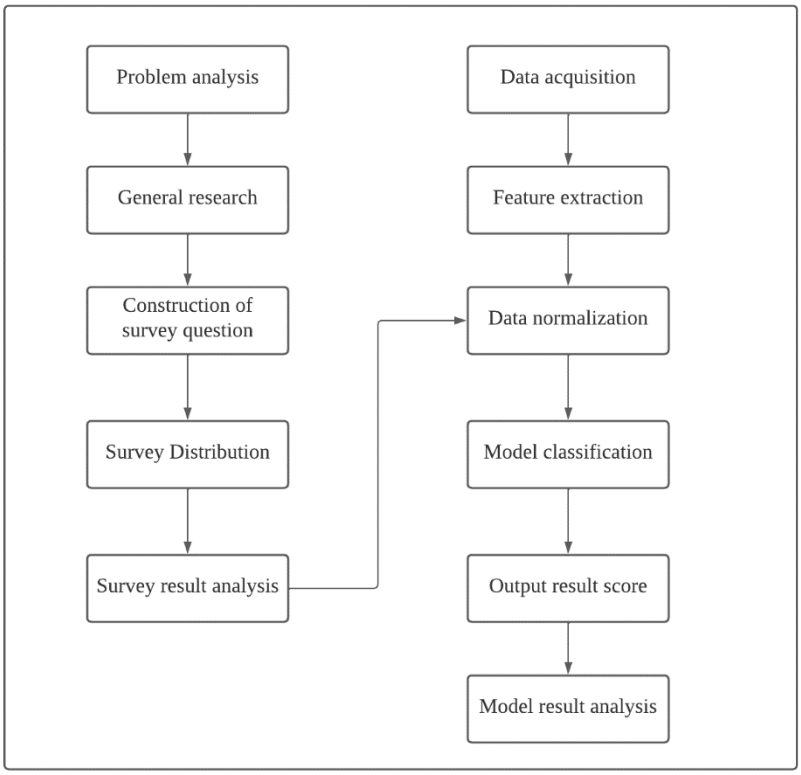


Figure 3.1 Project Flow Diagram

3.2 Tools to use

3.2.1 Hardware

The hardware involved in this project is a computer. The computer is issued for data preprocessing which is feature extraction and the generation of reputation score using the algorithm in the source code. It is also used in the validation of the final product model.

Table 3.1 Specifications of laptop

Description	Specifications
Model	TUF Gaming FX505GE_FX505GE
Processor	Intel Core i7-8750H
Operating System	Windows 10
Graphic	NVIDIA GeForce GTX 1050 Ti 4GB DDR5
Memory	16GB DDR4 RAM
Storage	931.5GB SATA HDD 512GB SATA SSD

3.2.2 Software

The software included in this project will be the Jupyter notebook. It is a free and open-source platform that supports the core programming language for this project which is python. Jupyter notebooks are able to illustrate the analysis process step by step which will be beneficial when training the model. Another software being used will be the python programming language which is a popular programming language, especially in the machine learning field due to its easy syntax and rich library.

3.3 Timeline

Task/Period	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Revise FYP1														
Revise codes of FYP1														
Research on additional machine learning model														
Finding dataset														
Implementation of new models														
Model training														
Model cross validation														
Model fine tuning														
Model testing														
Result Analysis														
Formatting report														
Finalize report														
Presentation														

Figure 3.2 FYP2 Timeline

CHAPTER 4: SYSTEM DESIGN

4.1 System block diagram

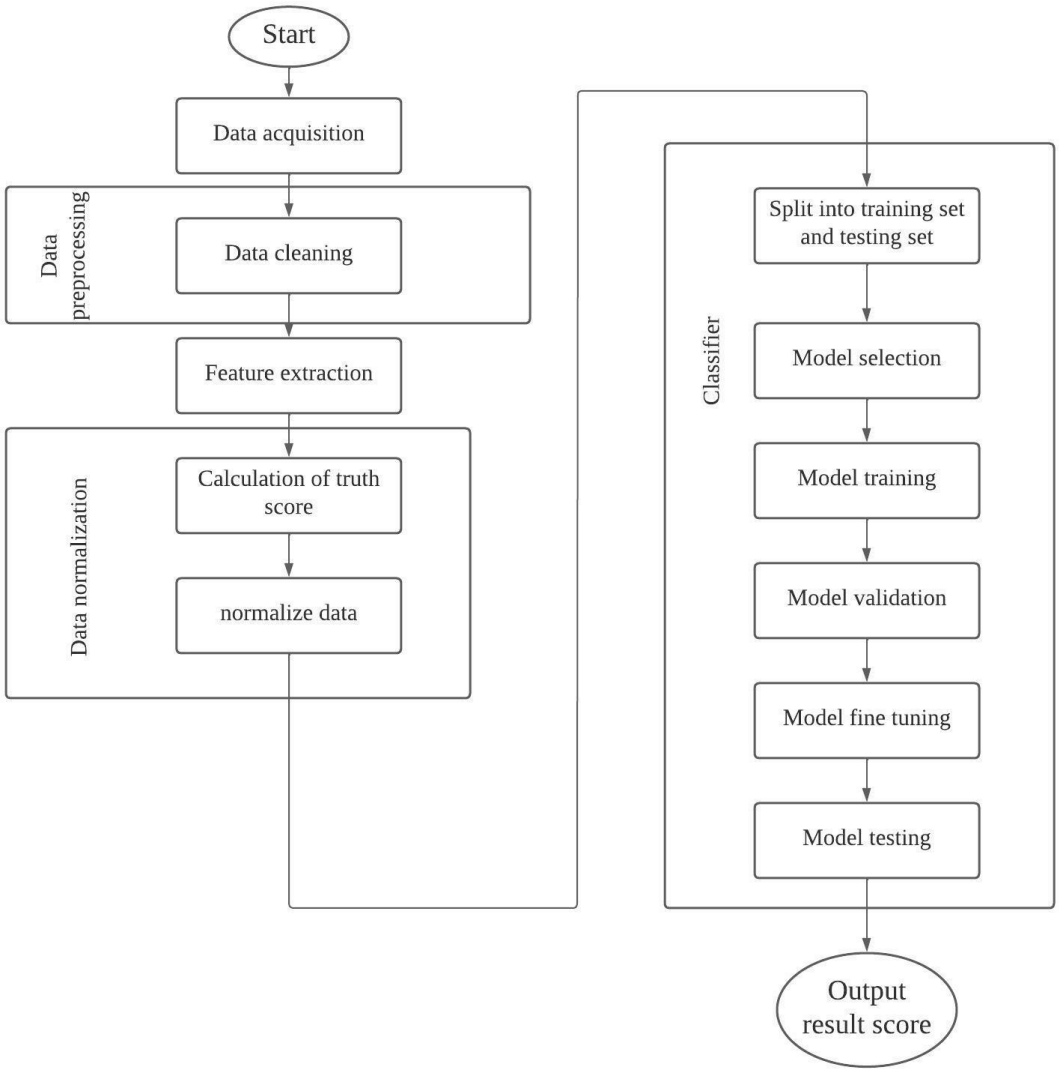


Figure 4.1 Block diagram of the model

4.1.1 Data collection

After deciding on the project direction, data collection is carried out as a fundamental part of the model. Due to company policy and employees' privacy considerations, actual real-life examples of dialogs on social engineering attacks are difficult to be acquired for the project. Therefore, an alternative way of searching datasets on the internet is carried out but even so, there is still very limited dataset online. However, an

excel .csv file is found on GitHub with over 150 records which will be used for this project.

4.1.2 Data cleaning

The data preprocessing process is performed after acquiring the data needed. Some NULL data will be handled and unwanted or unnecessary tags and metadata will be sanitized to generate well-formatted data. This will ease the subsequent process by eliminating small data contributions associated with experimental errors.

4.1.3 Feature extraction

The features are the criterion of the dialogs in the datasets that hold a certain meaning or context to the intent of the sender. The stage is to extract relevant feature elements that are associated with the detection of a social engineering attack. Examples of these features are sender address field value, receiver address field value, date, time, dialog which is the body of the email and etc. This paper focuses on phishing email therefore the initial features will be concentrated on the text of the email body.

4.1.4 Calculation of truth score

Based on the novelty ideas in paper [4], the calculation of truth score involves a sequence of steps to be followed. The general purpose is to preprocess dialogs with the consideration of weighting components of the email and rigor mathematical expression to generate a comprehensive dataset.

The steps are as follows:

1. The extraction of all URLs from the dialog text by the usage of a regex pattern matcher.
2. The URL links, if exists, is sent to VT (Virus Total) API to evaluate if it is malicious.
3. According to the result given by the API, the number of risky site reports is determined, given by x. By implementing equation 1, the number is scaled between 0 and 1. The value of S_L is a measurement of the degree of legitimacy where a high value indicates less legitimacy on the context. The usage of an exponential function instead of a linear function punish more severely on a higher number of risky site. The constant a (alpha) can be adjusted on the harshness of the risky site report. For this instance, 0.6

is set to be the constant value on the equation based on the survey result in the following chapter.

$$S_L = 1 - e^{-ax}$$

Figure 4.2 Equation 1

4. Next, check for spelling by using the SpellChecker library.

5. This step is identical to step (3) where the risk site report is replaced by misspelled words with the exception that alpha value is set to a. S_{SP} is an indication of spelling quality where higher values represent poorer spelling.

$$S_{SP} = 1 - e^{-ax}$$

Figure 4.3 Equation 2

6. The subsequent step is to use the corrected spelling of the dialog and check it against the list of blacklist words. Examples of these words are password, credentials, database etc. M_B refers to the number of blacklist matched words.

7. In this step, the intent verbs and adjectives such as urgent, must, require etc are emphasized and checked. The value of the number of these words are indicated by M_I.

8. To prevent the result being ambiguous, results are tuned with multiplication of values M_B and M_I by the weights of W_B and W_I weighted at values 1 and 1.3 respectively with reference on survey result. This is an optional step as it only provides customization on the changes of weight if one considers any of the two component need to be considered more important. Hence the value x is given by equation 3.

$$x = (M_B \times W_B) + (M_I \times W_I)$$

Figure 4.4 Equation 3

9. At this step, with the same usage of the mathematical function in equation 2, the value of is normalized where a = 0.4 is the most appropriate constant. S_I refers to the context intent where a higher S_I value is an indication of a more concentration of blacklisted words.

$$S_I = 1 - e^{-ax}$$

Figure 4.5 Equation 4

4.1.5 Split into training set and testing set

In this stage, the datasets are split into 80% training set and 20% testing set. The datasets being used for the project include another .csv preprocessed dataset and a method called synthetic minority oversampling technique is carried out to tackle the presence of imbalanced dataset due to the nature of itself.

4.1.6 Model selection

New machine learning techniques different from the novelty will be used on this model. The project will only cover 4 different types of machine learning models which are support vector machine (SVM)**,** stochastic gradient descent (SGD), K nearest neighbours (KNN) and Neural Network.

4.1.7 Model training

Next, the models with a classifier will then be trained with the datasets scores input to classify the presence of a social engineering attack

4.1.8 Model validation

The model undergoes 5-fold cross validation in this process. This process is carried out to prevent any potential overfitting cases and check if the models are actually training in the previous training stage.

4.1.9 Model fine tuning

The process includes the defining sets of hyperparameter values for each model. Grid search tuning technique will be used to compute and find the most optimal value for the models.

4.1.10 Model testing

The models will then be tested using the testing dataset to evaluate how well they perform on unseen data.

4.1.11 Summarization of result score

Finally, the performance metrics result score of each model such as accuracy, precision, recall, F_1 scores and ROC AUC scores are generated and summarized. These performance metrics results of different machine learning techniques are compared and contrasted. It will also then be used in further analysis.

4.2 General research

Preliminary research on social engineering attacks among Malaysia SMEs is done prior to having a more in-depth understanding of the subject matter. Social engineering is a broad research direction that branches to many aspects. It is a non-technical method that emphasizes human interaction which exploits the low awareness of human on the information they possess by tricking them to break standard security protocol and procedures. Therefore, the human factor plays a critical role in SE and targets not only big enterprises but also small-medium enterprises (SMEs).

The centric groundwork of this paper is mainly related to Malaysia where fact findings are carried out from various sources. Since the outbreak of the COVID-19 pandemic in March 2020, the unemployment rate in the country has risen exponentially and caused an increase in social criminal activities in cyberspace. It has been revealed that 65% Malaysians report on the increase in the number of phishing emails directed at employees during the pandemic period. This indicates that phishing assaults against businesses increased significantly during the epidemic, as millions of home-based workers became a prime target for hackers. According to [7], phishing's success has been attributed to its capacity to grow and diversify over time, matching assaults to current events or worries, such as the pandemic, and preying on human emotions and trust. Phishing assaults are viewed as a low-level danger by many organizations, yet this underestimates their impact. Phishing is frequently the opening stage of a more complex, multi-stage attack.

In an interview with the Communications and Multimedia Minister of Malaysia, Datuk Saifuddin Abdullah, it reported that the Cyber999 Help Centre, a cybersecurity incident response centre which is managed by CyberSecurity Malaysia has received a total of 4615 cybersecurity incident reports. He also stated that internet usage has risen dramatically in the aftermath of the Covid-19 outbreak and the movement control order (MCO). As a result, Malaysians are now more vulnerable to cyber threats and attacks. Figure 4.6 shows the illustration of the incident statistic in 2020 from January to June.

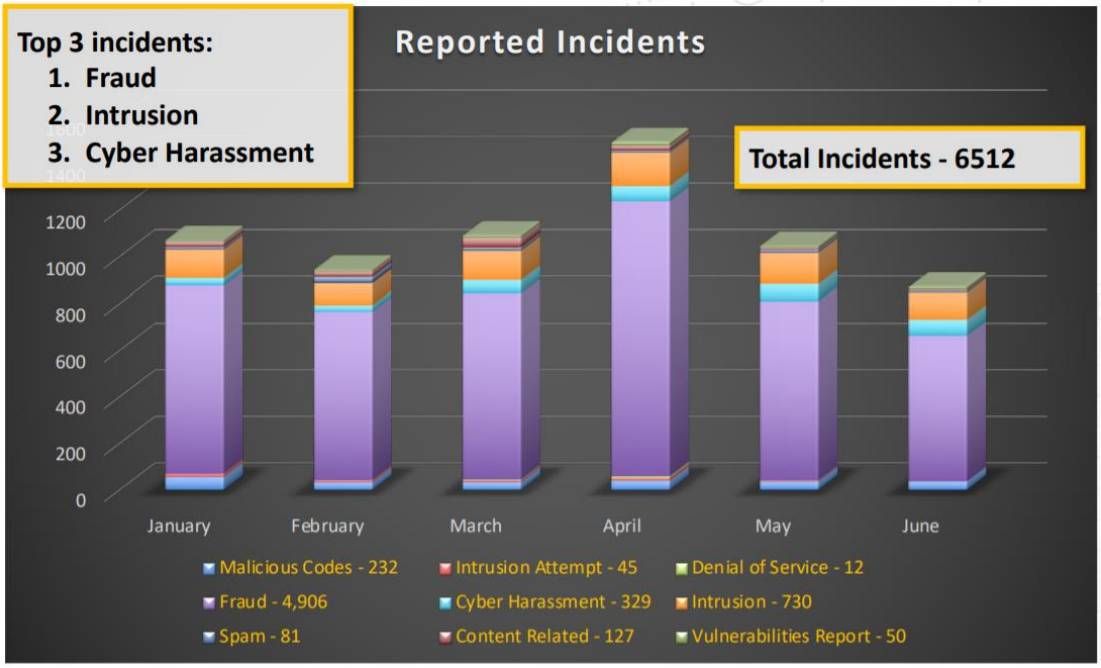


Figure 4.6 Statistic of security incidents from January to June 2020

It shows that the most reported cases are fraud which is strongly correlated to social engineering with its very nature in deceiving people. Other incidents are also possible products of SE attacks as phishing emails may be used to persuade users to install malware or share credentials that grant access to the business network, making a breach of intrusion in [7]. In addition, MyCERT (Malaysia Computer Emergency Response Team) noticed a rise in various cyber security assaults leveraging on the COVID 19 Pandemic during the Movement Restriction Order (MCO) with COVID-19 Phishing emails as one of the major threats.

According to [8], cyber security incidents have the potential to cost the country RM51 billion, or more than 4% of its entire GDP, and cybersecurity is the core important component in preventing cyber incidents and threats that undermine the country's sovereignty and economy. Wan Murdani Wan Mohamad, MDEC's director of digital infrastructure and services, indicated that 84 percent of Malaysia's SMEs had been harmed in some way by cyber threats, with 76 percent having been victims of several attacks. The Covid-19 epidemic spawned a distant work and quarantine culture, which drives people to go virtual and digital, opening up new doors for cybercrime and cyberattacks. Social engineering works best when targeting human emotions like fear, curiosity, greed, helpfulness, and urgency, according to savvy cybercriminals. It's expected that phishing assaults in Malaysia have escalated since the epidemic began,

as evidenced by the newest survey from security firm Sophos, titled Phishing Insights, 2021 as seen in [8].

As shown in [8], based on the findings of Kapersky Security Network for Malaysia, from 18.53 million in the first quarter (Q1'21), the number of web threats increased by 56 percent to 28.93 million in the second quarter (Q2'21). Remote working cybersecurity hazards, social engineering attacks, and ransomware, according to Kaspersky Southeast Asia general manager Yeo Siang Tiong, are three trends in cyberattacks and security concerns identified this year. Working from home introduces new cybersecurity threats because home offices are frequently less secure than central offices. Yeo [8] has stated that traditional security vetting may not have been as thorough as usual in the effort to keep things running, and cybercriminals have adapted their strategies to take advantage. Many staff uses personal devices for two-factor authentication and to communicate with clients via mobile messaging apps. Due to the fact that barriers between personal and professional life are blurred, there is a greater risk of sensitive material slipping into the wrong hands which is shown in [8]. Moreover, Yeo [8] also stated that social engineering attacks like phishing have been targeting remote workforces, with attackers focusing on workers that connect to their employer's network from home because they are vulnerable targets. Employees are also subjected to regular phishing assaults, as well as an increase in whaling attacks aimed at senior executives. Organizations are improving their anti-phishing defenses, but cybercriminals are also continuously seeking new ways to stay ahead of the game. This includes advanced phishing kits that target victims differently based on where they are which may increase the rate of success of an attack which is seen in [8].

In summary, these are the proofs and analyses of the rising cases in social engineering attacks. It is clear to have research on the countermeasure of these attacks. An ideal solution stated in [7] is to keep phishing emails from ever reaching the target with the senior and prepared staff who can notice and report questionable messages before they get any further. This is somewhat similar to the proposition from [3] in their paper but unfortunately, shares the same limitation which is the persistence of the human factor. The official Malaysian Communication and Multimedia Commission has provided 4 criteria in identifying phishing emails that exploit the fear emotion in people. These 4 aspects are a theme of threat in the context, a claim on a compromised account

or fraudulent activity, unauthorized changes and a claim in needing information for verification purposes.

CHAPTER 5: EXPERIMENT/SIMULATION

5.1 Software setup

For the prototype implementation of the model, Jupyter notebook will be used as it is a well-known open-source software that is free for download. It is a notebook kernel that executes the codes in a notebook document. The advantage of this software is it is able to run a section of code individually in a cell which is optimal for machine learning model. The programming language that will be used is Python which consists of many library and modules for machine learning such as numpy, pandas, nltk, spellchecker and many more for this project. An application called Anaconda need to be installed which is used for package management and deployment that includes jupyter notebook by default. In figure 5.1, the notebook can be launched by inputting 'jupyter notebook' in the anaconda CLI prompt.

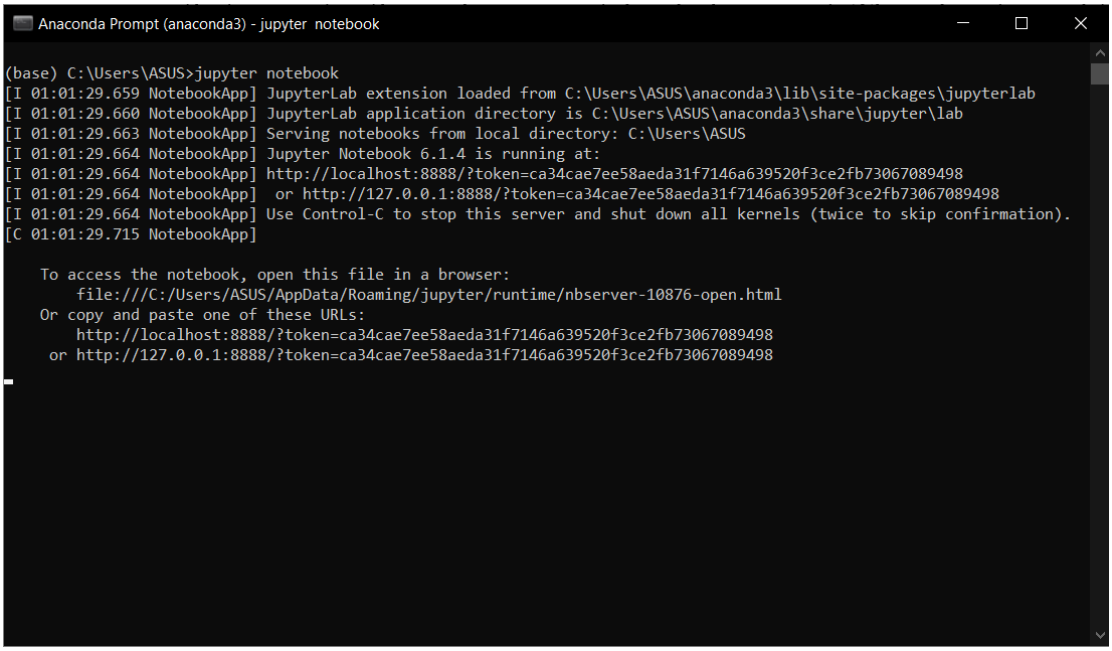


Figure 5.1 Booting jupyter notebook

Figure 5.2 Jupyter notebook interface

Figure 5.2 shows the user interface when jupyter notebook is launched successfully on the browser.

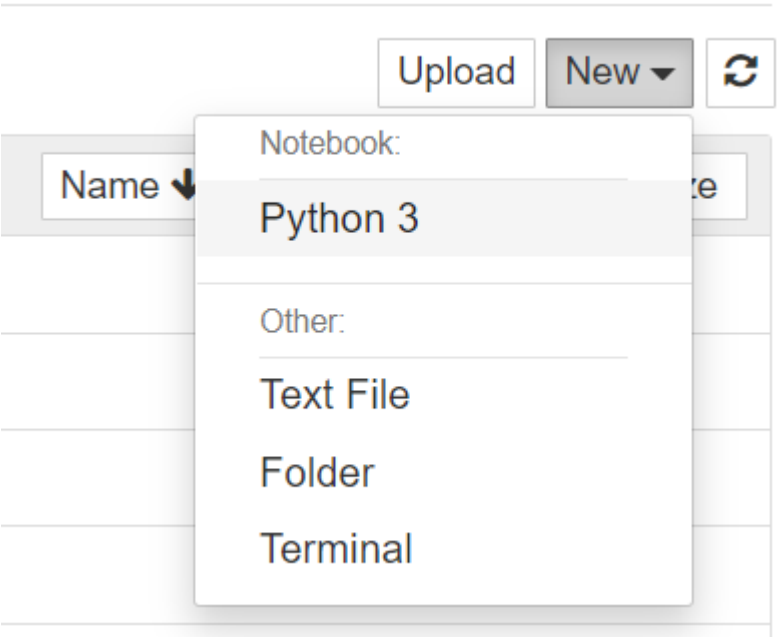


Figure 5.3 Creation of empty python file

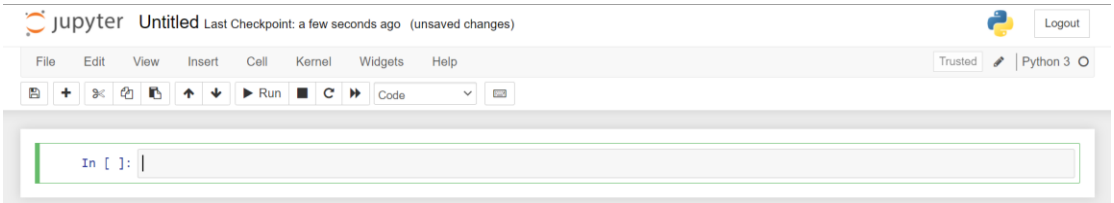


Figure 5.4 Interface of python file in jupyter notebook

After directing to the directory workspace path of the project, a new notebook with Python 3 can be created shown in figure 5.3. The figure 5.4 is the interface of a new notebook with an empty cell. Something worth highlighted is the fact that as jupyter notebook is a notebook kernel, it will lose the connection and be unable to use the python file once the command prompt is closed.

5.2 Setting and Configuration

Based on the previous research and facts gathering, the detection model will focus on the contextual dialog of the dataset as phishing is the main social engineering vector. This project will be derived from the novelty idea from paper [4]. The concept is applied to current Malaysia SMEs sector with modification and tinkering according to the data collected in the survey. After researching on the internet, the dataset used for the project is a dialogue dataset with 148 records and can be found in https://github.com/npolatidis/seader . The dataset is located to be in the same directory as the python project file and loaded shown in figure 5.6 after importing required libraries in figure 5.5.

```
# Import Library pandas
import pandas as pd
# Import the package re which is meant for Regular Expressions
import re
# Import library to manage environment variables
from dotenv import load_dotenv
load_dotenv()
# Import library for various time-related function
import time
```

Figure 5.5 Code snippets of importing library

```
# Load dataset in a dataframe using Panda
df = pd.read_csv('./dialogues_dataset_formatted.csv')
print("Dataset shape:", df.shape)
```

```
Dataset shape: (149, 7)
```

Figure 5.6 Code snippets of loading dataset

5.3 System Experiment

The exploratory data analysis is carried out prior to understanding the structure and types of the data being processed. This will ease the process in further stages. The column labels are shown in figure 5.7 and their respective data type in figure 5.8. Figure 5.9 shows the first five records in the dataset. Figure 5.11 indicates how many records are labelled as an attack and figure 5.10 shows the opposite. Figure 5.12 is to check for any NULL value present in the dataset.

```
df.columns
```

```
Index(['voice_id', 'num_caller', 'num_receiver', 'date', 'time', 'is_attack',
       'dialog'],
      dtype='object')
```

Figure 5.7 Code snippets and output of displaying dataset column

```
df.dtypes
```

```
voice_id        int64
num_caller      int64
num_receiver    int64
date            int64
time            int64
is_attack        bool
dialog         object
dtype: object
```

Figure 5.8 Code snippets and output of displaying datatype

```
print(df.head())
```

```
     voice_id  num_caller  num_receiver      date  time  is_attack  \
0  9999999999  6094660035    9082456893  20050215  1010       True
1  4444444444  6094660035    9082456893  20050215  2050      False
2  5555555555  6094660036    9082456847  20050221  1254      False
3  3333333333  6094660038    9082456829  20050422  1315       True
4  2222222222  6094660035    9082456896  20050515  1150       True

                                              dialog
0  Hi I'm Carl from orange. I am a sales rep. wou...
1  This is mike smith from orange. I am a sales r...
2  Hi this is mike jones from orange. I am the sa...
3  I am john doe from apple computers where I am ...
4  This is john candy from apple. I am a computer...
```

Figure 5.9 Code snippets and output of displaying first five records of dataset

```
total_false = df[df.is_attack == False]

total_false.count()
```

```
voice_id        71
num_caller      71
num_receiver    71
date            71
time            71
is_attack       71
dialog          71
dtype: int64
```

Figure 5.10 Code snippets and output of displaying number of legitimate records

```
total_true = df[df.is_attack == True]

total_true.count()
```

```
voice_id        78
num_caller      78
num_receiver    78
date            78
time            78
is_attack       78
dialog          78
dtype: int64
```

Figure 5.11 Code snippets and output of displaying number of malicious records

```
# Check for NULL values
df.isna().sum()
```

```
voice_id        0
num_caller      0
num_receiver    0
date            0
time            0
is_attack       0
dialog          0
dtype: int64
```

Figure 5.12 Code snippets and output of number of NULL value record

In this stage, all the significant criteria in the dialogue are extracted. First, an empty dictionary is initialized in figure 5.13 and the regular expression string of the URL seen in figure 5.14. The extraction of the urls from the dialog is performed by using a for loop to run through every dialog record in figure 5.15. It is stored in a variable which in then be appended to the array of dictionary in figure 5.16. Additional attribute labels are added for the subsequent process.

```
# create dictionary of dialogs
dialogs_dictionary = []
```

Figure 5.13 Code snippets of the creation of an empty dictionary

```
url_regex = "(?:https?://)?(?:(?i:[a-z]+\.)+)[^\s,]+"
```

Figure 5.14 Code snippets on assigning regular expression pattern

```
for index, row in df.iterrows():

    # The extraction of the urls from the dialog
    urls = re.findall(url_regex, row['dialog'])
    # The extraction of the dialogs as a whole
    dialog = row['dialog']
```

Figure 5.15 Code snippets on the extraction of urls and dialogues

```
# Append the dictionary with additional labels and intialize them
dialogs_dictionary.append({
    'misspelled_count': 0,
    'misspelled_words': [],
    'dialog': dialog,
    'urls': urls,
    'class': row['is_attack']
```

Figure 5.16 Code snippets on the appending of additional labels

To check if the url is malicious, an application programming interface (API) is integrated into the project. Virus Total (VT) is used for this purpose and the substitution of Web of Trust (WOT) proposed in the API. An account needs to be registered in order to obtain the API key to use it. The following process is shown in figure 5.17. A library called requests is imported to process http request in python. The data parameters are the key and url instances that are sent to the API to check for malicious website with a post request. This request is converted to JSON format and stored in the dictionary array with a new label ‘url_post’. The process is looped until all instances have been sent with a 31 seconds delay due to the API policy.

```
# The usage of the API requires the requests module in order to send http request
import requests



VT_API = 'https://www.virustotal.com/vtapi/v2/url/scan'

# The submission of the URL to the API to check for malicious website
for value in dialogs_dictionary:
    if value.get('urls'):
        parameter = {
            'apikey': 'b7a56ce80f82e5780cb12a420d79105154be9e4dc6fd98d0ac3f57ba69eafaf3',
            'url': value['urls'][0]
        }

        res_sub = requests.post(VT_API, data=parameter)
        # conversion of results to json format
        res_sub = res_sub.json()
        value['url_post'] = res_sub
        # delay for the API policies
        time.sleep(31)
```

Figure 5.17 Code snippets on the usage of API

Figure 5.18 shows the retrieval of the response report of the scanning result. The report parameter is set to get the resource and scan id which is iterated in a loop for each record. It is then converted to JSON format and stored in a new label in the dictionary called ‘url_report’

```
# get the scan report
VT_REPORT = 'https://www.virustotal.com/vtapi/v2/url/report'

for value in dialogs_dictionary:
    if value.get('urls'):
        parameter = {
            'apikey': 'b7a56ce80f82e5780cb12a420d79105154be9e4dc6fd98d0ac3f57ba69eafaf3',
            'resource': value['url_post']['resource'],
            'scan_id': value['url_post']['scan_id']
        }

        res_report = requests.get(VT_REPORT, params=parameter)
        # Covert the report to json format
        res_report = response_report.json()
        value['url_report'] = res_report
```

Figure 5.18 Code snippets on getting results from API

Figure 5.19 shows the summary of the scanning result of each record in the dictionary. This can be summarized to the number of VirusTotal partners who consider this url harmful, the verbose message of the scanning result and total number of partners who reviewed the link. These are all also embedded in the dictionary.

```
for scan in dialogs_dictionary:
    if scan.get('urls'):
        scan['url_scan_positives'] = scan['url_report']['positives']
        scan['url_scan_verbose_message'] = scan['url_report']['verbose_msg']
        scan['url_scan_total'] = scan['url_report']['total']
```

Figure 5.19 Code snippets of appending API results to dictionary

Next would be the process of checking the spelling errors of the sentences. The spellchecker library is imported for this purpose. In figure 5.21, the special characters such as ",", "\", "." and others are removed. The filtered dialogue is then assigned to a variable and appended to the dictionary. Figure 5.22 shows the code snippets in finding possible misspelled words in the syntax format of "{word} : {spell.correction(word)} -> {spell.candidates(word)}". Correction on the misspelled words is made in the dictionary and the misspelled word is appended as a new label in the dictionary. A counter labeled to indicate the number of misspelled words for each dialogue is recorded and also added to the dictionary.

```
from spellchecker import SpellChecker
import re
```

Figure 5.20 Code snippets of importing library

```
spell = SpellChecker()
for value in dialogs_dictionary:
    # First, the removal of special characters such as , / ? ;
    special_chars = [';', ':', '!', "*", ".", ",", "+", "-", "/", "?", "  "]
    text = ''.join(i for i in value['dialog'] if not i in special_chars)
    # The filtered dialog is assigned to the array
    value['dialog_clean'] = text
```

Figure 5.21 Code snippets on the removal of special characters

```
# find possible misspelled words
check_spelling = spell.unknown(text.split(" "))

for word in check_spelling  :
    if len(word) > 1:
        print(f'{word} : {spell.correction(word)} -> {spell.candidates(word)}')
        if word not in spell.candidates(word):
            # replace the misspelled words on the clean dialog
            value['dialog_clean'] = value['dialog_clean'].replace(word, spell.correction(word))
            value['misspelled_count'] += 1
            value['misspelled_words'].append(word)
```

Figure 5.22 Code snippets on checking misspelled words

Further on is the inspection of words in the dialogue against the blacklisted words. Figure 5.23 is the code snippets for the list of blacklisted words and the list of adjectives

that have an element of intent and urgency. As shown in figure 5.24, a for loop is used to go through all records in the dictionary and two counters which are the number of blacklisted words and the number of adjectives that show an intent are initialized. Then, a nested for loop is applied to scan through if the listed words exist in the dialogue. Finally, the total number of both number is being recorded in the dictionary.

```
# blacklisted words
blacklist = [
    'password', 'credentials', 'database', 'confidential', 'attack', 'access', 'passcode', 'pass'
]

# search for intent verbs and adjectives such as urgent, must and etc
intent_adjectives = [
    'need', 'must', 'urgent', 'warning', 'have to'
]
```

Figure 5.23 Code snippets on defining blacklisted words and intent adjectives

```
for value in dialogs_dictionary:
    blacklist_count = 0
    intent_adj_count = 0
    for word in value['dialog_clean'].split(" "):
        if word in blacklist:
            blacklist_count += 1
    for word in value['dialog_clean'].split(" "):
        if word in intent_adjectives:
            intent_adj_count += 1

    value['bl_count'] = blacklist_count
    value['ia_count'] = intent_adj_count
```

Figure 5.24 Code snippets on checking for blacklisted words and intent adjectives

Figure 5.25 shows the code snippets to covert the dictionary to a csv file for organization purposes.

```
import csv

csv_file = "dataset.csv"

keys = dialogs_dictionary[0].keys()
print(keys)
with open(csv_file, 'w', newline='')  as output_file:
    dict_writer = csv.DictWriter(output_file, keys)
    dict_writer.writeheader()
    dict_writer.writerows(dialogs_dictionary)
```

Figure 5.25 Code snippets on the conversion of the dictionary to csv file

The normalization of the data begins with the substitution of NULL values with 0. By using panda, an empty dataframe is created to store the calculated score for each feature. Figure 5.26 demonstrates the codes snippets of the mathematical expression in calculating those scores for the features which are url score, misspelled words score and blacklisted and intent adjectives scores.

```
import pandas as pd
import numpy as np
from math import e

csv_file = "dataset.csv"

# Substitution of NULL value with 0
data = pd.read_csv(csv_file).fillna(0)

df = pd.DataFrame()

# urls score
url_alpha = 0.6
df[0] = 1 - (e** -(url_alpha*data['url_scan_positives'].values))

# misspelled words score
miss_alpha = 0.5
df[1] = 1 - (e** -(miss_alpha*data['misspelled_count'].values))

# blacklisted and intent verb words score
blacklist_alpha = 0.4
blacklist_weight = 1
intent_adj_weight = 1.3
x = (blacklist_weight * data['bl_count'].values) + (intent_adj_weight * data['ia_count'].values)
df[2] = 1 - (e** -(blacklist_alpha*x))
```

Figure 5.26 Code snippets on calculating scores for each feature

For the purpose of increasing the dataset, another .csv file which has the same attribute features and preprocessed sample data is used. Figure 5.27 shows the implementation

of reading the dataset that had been preprocessed and figure 5.28 shows the implementation of concatenating this dataset with previous dataset together

```
df2 = pd.read_csv('compound_dataset.csv')
```

Figure 5.27 Code snippets of loading additional preprocessed dataset

```
df_renamed = df.rename(columns={0:'link', 1: 'spelling', 2: 'intent'})
```

```
df_concat = pd.concat([df_renamed, df2.drop('is_attack', axis=1)])
```

Figure 5.28 Code snippets on concatenation of dataset

The features score in the data frame are assigned the X variable and the binary class on whether the recorded is an attack is mapped as integer to Y variable which is shown in figure 5.29.

```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics

# Assign scores as features
X = df_concat
# Mapping binary class of true / false as 1 / 0
y = pd.Series(list(data['class'].values.astype(int)) + list(df2['is_attack']))
```

Figure 5.29 Code snippets on assigning X and Y value

The data is then separated to 80% training set and 20% testing set in figure 5.30.

```
# split train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 5.30 Code snippets on splitting of training and testing dataset

Due to the fact that there are many dialogs which are marked as not an attack, this causes a potential imbalanced dataset. The way to tackle this is by introducing a technique called Synthetic Minority Oversampling Technique (SMOTE) which is an oversampling technique where it will generate synthetic samples for the minority class.

The implementation is shown in figure 5.31.

```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_train, y_train = oversample.fit_resample(X_train, y_train)
```

Figure 5.31 Code snippets on implementation of SMOTE

Figure 5.32 shows the code snippets of defining the training result function of each model later.

```
# training result funciton
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
def training_result(train_acc, y_train, y_pred):
    print("Training accuracy: {:.4f}".format(train_acc))
    print('precision = {:.4f}'.format(precision_score(y_train, y_pred)))
    print('recall = {:.4f}'.format(recall_score (y_train, y_pred)))
    print('f1 score = {:.4f}'.format(f1_score(y_train, y_pred)))

    # results
    accuracy = metrics.accuracy_score(y_train, y_pred)
    print("Accuracy:", accuracy)
    print('----------------------------------------------------')
    print(metrics.classification_report(y_train, y_pred))
    print('----------------------------------------------------')
    print("Confusion matrix")
    confusion_matrix = metrics.confusion_matrix(y_train,y_pred)
    print(confusion_matrix)
```

Figure 5.32 Code snippets of defining the training result function

For this project, 3 classical machine learning model and 1 deep learning model were selected to carry out the training process. These model includes the Support Vector Machine (SVM), K-nearest neighbors algorithm (KNN), Stochastic Gradient Descent (SGD) and Neural Network model. Figure 5.33 shows the code snippets on the implementation of the SVM model and its training result on figure 5.34.

```
# Training accuracy
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
svmmodel = svm.SVC(random_state=42)
svmmodel.fit(X_train, y_train.ravel())

# predict
y_pred = svmmodel.predict(X_train)
train_acc = accuracy_score(y_train, y_pred)

training_result(train_acc, y_train, y_pred)
```

Figure 5.33 Code snippets on the implementation of SVM model

```
Training accuracy: 0.8392
precision = 0.8114
recall = 0.8838
f1 score = 0.8461
Accuracy: 0.8392255892255892
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      0.79      0.83       594
           1       0.81      0.88      0.85       594

    accuracy                           0.84      1188
   macro avg       0.84      0.84      0.84      1188
weighted avg       0.84      0.84      0.84      1188

-----------------------------------------------------
Confusion matrix
[[472 122]
 [ 69 525]]
```

Figure 5.34 Training result of SVM model

Figure 5.35 shows the code snippets on the implementation of the KNN model and its training result on figure 5.36.

```python
from sklearn import neighbors

# create the model
knnmodel = neighbors.KNeighborsClassifier(n_neighbors=25, weights='uniform')

# train the model
knnmodel.fit(X_train, y_train.ravel())

# predict
y_pred = knnmodel.predict(X_train)
train_acc = accuracy_score(y_train, y_pred)

training_result(train_acc, y_train, y_pred)
```

Figure 5.35 Code snippets on the implementation of KNN model

```
Training accuracy: 0.7020
precision = 0.8797
recall = 0.4680
f1 score = 0.6110
Accuracy: 0.702020202020202
------------------------------------------------------
              precision    recall  f1-score   support

           0       0.64      0.94      0.76       594
           1       0.88      0.47      0.61       594

    accuracy                           0.70      1188
   macro avg       0.76      0.70      0.68      1188
weighted avg       0.76      0.70      0.68      1188

------------------------------------------------------
Confusion matrix
[[556  38]
 [316 278]]
```

Figure 5.36 Training result of KNN model

Figure 5.37 shows the code snippets on the implementation of the SGD model and its result on figure 5.38.

```
from sklearn.linear_model import SGDClassifier

sgdmodel = SGDClassifier(random_state = 42)
#Train the model
sgdmodel.fit(X_train,y_train)
y_pred = sgdmodel.predict(X_train)
train_acc = accuracy_score(y_train, y_pred)

training_result(train_acc, y_train, y_pred)
```

Figure 5.37 Code snippets on the implementation of SGD model

```
Training accuracy: 0.8359
precision = 0.8018
recall = 0.8923
f1 score = 0.8446
Accuracy: 0.8358585858585859
---------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.78      0.83       594
           1       0.80      0.89      0.84       594

    accuracy                           0.84      1188
   macro avg       0.84      0.84      0.84      1188
weighted avg       0.84      0.84      0.84      1188

---------------------------------------------------------
Confusion matrix
[[463 131]
 [ 64 530]]
```

Figure 5.38 Training result of SGD model

Figure 5.39 shows the code snippets on the implementation of the Neural Network model and its result on figure 5.40.

```python
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix

# revisar funcion de activacion
nnmodel = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,15,15))
nnmodel.fit(X_train, y_train.ravel())

# predict
y_pred = nnmodel.predict(X_train)
train_acc = accuracy_score(y_train, y_pred)

training_result(train_acc, y_train, y_pred)
```

Figure 5.39 Code snippets on the implementation of Neural Network model

```
Training accuracy: 0.8460
precision = 0.8309
recall = 0.8687
f1 score = 0.8494
Accuracy: 0.8459595959595959
------------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      0.82      0.84       594
           1       0.83      0.87      0.85       594

    accuracy                           0.85      1188
   macro avg       0.85      0.85      0.85      1188
weighted avg       0.85      0.85      0.85      1188

------------------------------------------------------
Confusion matrix
[[489 105]
 [ 78 516]]
```

Figure 5.40 Training result of Neural Network model

Figure 5.41 shows the code snippets for the visualization of the ROC curve for four of the model being trained and the curve diagram that consists of each model is on figure 5.42.

```
#ROC Before CV
from sklearn import metrics
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import plot_roc_curve
y_scores_svm = svmmodel.decision_function(X_train)
y_scores_sgd = sgdmodel.decision_function(X_train)
y_scores_knn = knnmodel.predict_proba(X_train)
y_scores_knn = y_scores_knn[:,1]
y_scores_nn = nnmodel.predict_proba(X_train)
y_scores_nn = y_scores_nn[:,1]


# Compute the FPR, TPR and the thresholds of the classifiers using roc_curve

fpr_svm, tpr_svm, thresholds_svm = metrics.roc_curve(y_train, y_scores_svm)
fpr_sgd, tpr_sgd, thresholds_sgd = metrics.roc_curve (y_train, y_scores_sgd)
fpr_knn, tpr_knn, thresholds_knn = metrics.roc_curve (y_train, y_scores_knn)
fpr_nn, tpr_nn, thresholds_nn = metrics.roc_curve (y_train, y_scores_nn)

roc_auc_svm = metrics.auc(fpr_svm, tpr_svm)
#Plotting ROC curve for all 4 models

disp = RocCurveDisplay.from_predictions(y_true = y_train, y_pred = y_scores_svm, name = "SVM")
RocCurveDisplay.from_predictions(y_train, y_scores_sgd, name = "SGD", ax=disp.ax_)
RocCurveDisplay.from_predictions(y_train, y_scores_knn, name = "KNN", ax=disp.ax_)
RocCurveDisplay.from_predictions(y_train, y_scores_nn, name = "Neural Network", ax=disp.ax_)

plt.show()
```

Figure 5.41 Code snippets for visualization of ROC curve

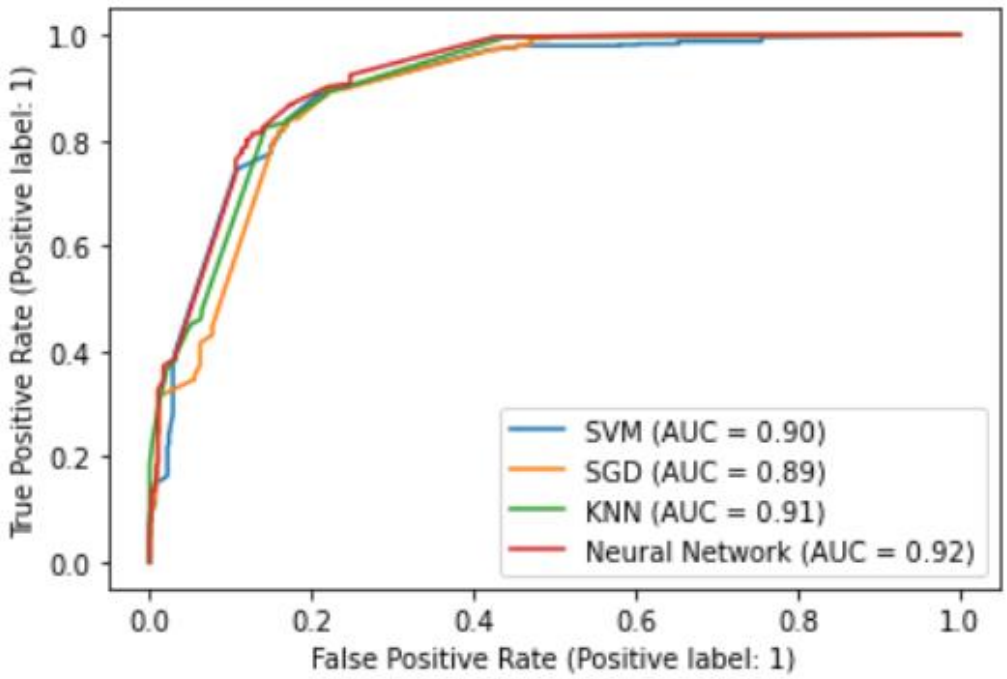


Figure 5.42 ROC curve diagram for each model

Figure 5.43 shows the code snippets and result of a more precise figure on the area under curve for four of the models.

```
#Area under curve for all 4 models
from sklearn.metrics import roc_auc_score
auc_svm = roc_auc_score(y_train, y_scores_svm)
print('AUC for Support Vector Machine = {:.4f}'.format(auc_svm))

auc_sgd = roc_auc_score(y_train, y_scores_sgd)
print('AUC for SGD = {:.4f}'.format(auc_sgd))

auc_knn = roc_auc_score(y_train, y_scores_knn)
print('AUC for KNN = {:.4f}'.format(auc_knn))

auc_nn = roc_auc_score(y_train, y_scores_nn)
print('AUC for Neural Network = {:.4f}'.format(auc_nn))
```

```
AUC for Support Vector Machine = 0.9017
AUC for SGD = 0.8932
AUC for KNN = 0.9077
AUC for Neural Network = 0.9178
```

Figure 5.43 Code snippets and result of ROC score for each model in 4 decimal places

In order to avoid overfitting of the data, the project uses a 5-fold cross validation to estimate the performance of the model on unseen data. Figure 5.44 shows the implementation of cross validation and its result on SVM.

```
svm_result = cross_validate (svmmodel, X_train, y_train, cv=5, scoring=['accuracy', 'precision', 'recall', 'f1'])
print('Support Vector Machine: validation accuracy = {:.4f}'.format(svm_result['test_accuracy'].mean()))
print('Support Vector Machine: validation recall = {:.4f}'.format(svm_result['test_recall'].mean()))
print('Support Vector Machine: validation precision = {:.4f}'.format(svm_result['test_precision'].mean()))
print('Support Vector Machine: validation f1 = {:.4f}'.format(svm_result['test_f1'].mean()))
print('-----------------')
y_pred_cv = cross_val_predict(svmmodel, X_train, y_train, cv = 5)
confusion_matrix = metrics.confusion_matrix(y_train,y_pred_cv)
print(confusion_matrix)
```

```
Support Vector Machine: validation accuracy = 0.8292
Support Vector Machine: validation recall = 0.8518
Support Vector Machine: validation precision = 0.8150
Support Vector Machine: validation f1 = 0.8321
-----------------
[[479 115]
 [ 88 506]]
```

Figure 5.44 Code snippets and performance result of SVM after cross validation

Figure 5.45 shows the implementation of cross validation and its result on SGD.

```
sgd_result = cross_validate (sgdmodel, X_train, y_train, cv=5, scoring=['accuracy', 'precision', 'recall', 'f1'])
print('SGDClassifier: validation accuracy = {:.4f}'.format(sgd_result['test_accuracy'].mean()))
print('SGDClassifier: validation recall = {:.4f}'.format(sgd_result['test_recall'].mean()))
print('SGDClassifier: validation precision = {:.4f}'.format(sgd_result['test_precision'].mean()))
print('SGDClassifier: validation f1 = {:.4f}'.format(sgd_result['test_f1'].mean()))
print('-----------------')
y_pred_cv = cross_val_predict(sgdmodel, X_train, y_train, cv = 5)
confusion_matrix = metrics.confusion_matrix(y_train,y_pred_cv)
print(confusion_matrix)
```

```
SGDClassifier: validation accuracy = 0.8249
SGDClassifier: validation recall = 0.8299
SGDClassifier: validation precision = 0.8226
SGDClassifier: validation f1 = 0.8251
-----------------
[[487 107]
 [101 493]]
```

Figure 5.45 Code snippets and performance result of SGD after cross validation

Figure 5.46 shows the implementation of cross validation and its result on KNN.

KNN

```
knn_result = cross_validate (knnmodel, X_train, y_train, cv=5, scoring=['accuracy', 'precision', 'recall', 'f1'])
print('KNN: validation accuracy = {:.4f}'.format(knn_result['test_accuracy'].mean()))
print('KNN: validation recall = {:.4f}'.format(knn_result['test_recall'].mean()))
print('KNN: validation precision = {:.4f}'.format(knn_result['test_precision'].mean()))
print('KNN: validation f1 = {:.4f}'.format(knn_result['test_f1'].mean()))
print('-----------------')
y_pred_cv = cross_val_predict(knnmodel, X_train, y_train, cv = 5)
confusion_matrix = metrics.confusion_matrix(y_train,y_pred_cv)
print(confusion_matrix)
```

```
KNN: validation accuracy = 0.7600
KNN: validation recall = 0.6058
KNN: validation precision = 0.8826
KNN: validation f1 = 0.7025
-----------------
[[543  51]
 [234 360]]
```

Figure 5.46 Code snippets and performance result of KNN after cross validation

Figure 5.47 shows the implementation of cross validation and its result on Neural Network.

```
nn_result = cross_validate (nnmodel, X_train, y_train, cv=5, scoring=['accuracy', 'precision', 'recall', 'f1'])
print('NN: validation accuracy = {:.4f}'.format(nn_result['test_accuracy'].mean()))
print('NN: validation recall = {:.4f}'.format(nn_result['test_recall'].mean()))
print('NN: validation precision = {:.4f}'.format(nn_result['test_precision'].mean()))
print('NN: validation f1 = {:.4f}'.format(nn_result['test_f1'].mean()))
print('-----------------')
y_pred_cv = cross_val_predict(nnmodel, X_train, y_train, cv = 5)
confusion_matrix = metrics.confusion_matrix(y_train,y_pred_cv)
print(confusion_matrix)
```

```
NN: validation accuracy = 0.8291
NN: validation recall = 0.8586
NN: validation precision = 0.8107
NN: validation f1 = 0.8335
-----------------

[[484 110]
 [ 81 513]]
```

Figure 5.47 Code snippets and performance result of Neural Network after cross validation

Figure 5.48 shows the code snippets and curve diagram for the visualization of the ROC curve for four of the model being trained after cross validation.

```
disp = RocCurveDisplay.from_predictions(y_true = y_train, y_pred = y_scores_svm, name = "SVM")
RocCurveDisplay.from_predictions(y_train, y_scores_sgd, name = "SGD", ax=disp.ax_)
RocCurveDisplay.from_predictions(y_train, y_scores_knn, name = "KNN", ax=disp.ax_)
RocCurveDisplay.from_predictions(y_train, y_scores_nn, name = "Neural Network", ax=disp.ax_)
plt.show()
```

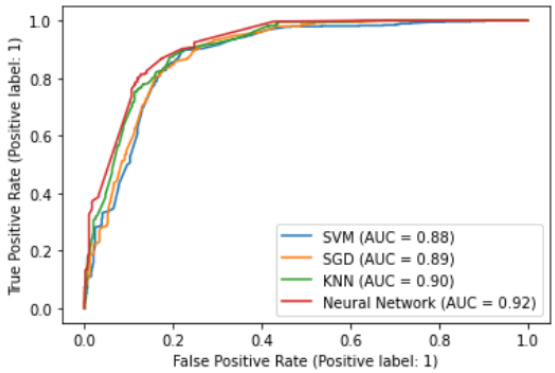


Figure 5.48 Code snippets and ROC curve diagram on each model after cross validation

Figure 5.49 shows the code snippets and result of a more precise figure on the area under curve for four of the models after cross validation.

```
#Area under curve for all 4 models
from sklearn.metrics import roc_auc_score
auc_svm = roc_auc_score(y_train, y_scores_svm)
print('AUC for Support Vector Machine = {:.4f}'.format(auc_svm))

auc_sgd = roc_auc_score(y_train, y_scores_sgd)
print('AUC for SGD = {:.4f}'.format(auc_sgd))

auc_knn = roc_auc_score(y_train, y_scores_knn)
print('AUC for KNN = {:.4f}'.format(auc_knn))

auc_nn = roc_auc_score(y_train, y_scores_nn)
print('AUC for Neural Network = {:.4f}'.format(auc_nn))
```

```
AUC for Support Vector Machine = 0.8808
AUC for SGD = 0.8885
AUC for KNN = 0.9047
AUC for Neural Network = 0.9178
```

Figure 5.49 Code snippets and result of ROC score for each model in 4 decimal places after cross validation

After that, the experiment is continued by fine tuning the hyperparameters of each model to find the most optimal hyperparamtersfor their respective algorithms. This is carried out using the grid search approach which is an exhaustive search to compute optimum values of hyperparameters. Figure 5.50 shows the set of hyperparamter values for the grid search technique to take place for SVM mdoel.

```
from sklearn.model_selection import GridSearchCV
tuned_parameters = {
 'C': [1, 10, 100,500, 1000], 'kernel': ['linear','rbf'],
 'C': [1, 10, 100,500, 1000], 'gamma': [1,0.1,0.01,0.001, 0.0001], 'kernel': ['rbf'],

    }
```

Figure 5.50 Code snippets on hyperparameter values of SVM model

Figure 5.51 shows the code implementation for grid search tuning and figure 5.52 is the code snippets on training the tuned model.

```
svm_model_tuned = GridSearchCV(svmmodel, tuned_parameters,cv=5,scoring='accuracy')
```

Figure 5.51 Code snippets on grid search tuning for SVM model

```
svm_model_tuned.fit(X_train, y_train)
```

Figure 5.52 Code snippets on training tuned SVM model

Figure 5.53 shows the result of the best parameters of the tuned model using grid search. Figure 5.54 shows the implementation of the prediction value for the tuned SVM model and performance metrics on training set in figure 5.55

```
print(svm_model_tuned.best_params_)
```

```
{'C': 500, 'gamma': 0.0001, 'kernel': 'rbf'}
```

Figure 5.53 Result of best hyperparameter values of SVM model

```
y_pred_cv = cross_val_predict (svm_model_tuned, X_train, y_train, cv=3)
```

Figure 5.54 Code snippets on predicting value with tuned SVM model

```
train_acc = accuracy_score(y_train, y_pred_cv)

print("Training accuracy: {:.4f}".format(train_acc))
print('precision = {:.4f}'.format(precision_score(y_train, y_pred_cv)))
print('recall = {:.4f}'.format(recall_score (y_train, y_pred_cv)))
print('f1 score = {:.4f}'.format(f1_score(y_train, y_pred_cv)))
```

```
Training accuracy: 0.8308
precision = 0.8134
recall = 0.8586
f1 score = 0.8354
```

Figure 5.55 Performance metrics result of tuned SVM model

Figure 5.56 shows the set of hyperparamter values for the grid search technique to take place for KNN mdoel.

```
#List Hyperparameters that we want to tune.
leaf_size = list(range(1,50))
n_neighbors = list(range(1,30))
p=[1,2]
#Convert to dictionary
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
```

Figure 5.56 Code snippets on hyperparameter values of KNN model

Figure 5.57 shows the code implementation for grid search tuning and figure 5.58 is the code snippets on training the tuned model.

```
#Use GridSearch
clf = GridSearchCV(knnmodel, hyperparameters, cv=5, verbose =1)
```

Figure 5.57 Code snippets on grid search tuning for KNN model

```
#Fit the model
best_model = clf.fit(X_train, y_train)
```

Figure 5.58 Code snippets on training tuned KNN model

Figure 5.59 shows the result of the best parameters of the tuned model using grid search. Figure 5.60 shows the implementation of the prediction value for the tuned KNN model and performance metrics on training set in figure 5.61.

```
print(best_model.best_params_)
```

```
{'leaf_size': 30, 'n_neighbors': 22, 'p': 1}
```

Figure 5.59 Result of best hyperparameter values of KNN model

```
y_pred_cv = cross_val_predict (best_model, X_train, y_train, cv=3, verbose =1)
```

Figure 5.60 Code snippets on predicting value with tuned KNN model

```
train_acc = accuracy_score(y_train, y_pred_cv)

print("Training accuracy: {:.4f}".format(train_acc))
print('precision = {:.4f}'.format(precision_score(y_train, y_pred_cv)))
print('recall = {:.4f}'.format(recall_score (y_train, y_pred_cv)))
print('f1 score = {:.4f}'.format(f1_score(y_train, y_pred_cv)))
```

```
Training accuracy: 0.6953
precision = 0.8372
recall = 0.4848
f1 score = 0.6141
```

Figure 5.61 Performance metrics result of tuned KNN model

Figure 5.62 shows the set of hyperparameter values for the grid search technique to take place for SGD mdoel.

```
loss = ['hinge', 'log', 'squared_hinge']
penalty = ['l2', 'elasticnet']
alpha = [0.0001, 0.001, 0.01, 0.1, 1]
learning_rate = ['constant', 'optimal']
eta0 = [1, 10, 100]
param_distributions = dict(loss=loss,
penalty=penalty,
alpha=alpha,
learning_rate=learning_rate,
eta0=eta0)
```

Figure 5.62 Code snippets on hyperparameter values of SGD model

Figure 5.63 shows the code implementation for grid search tuning and figure 5.64 is the code snippets on training the tuned model.

```
sgd_model_tuned = GridSearchCV(sgdmodel, param_distributions,cv = 5, scoring='roc_auc', verbose =1)
```

Figure 5.63 Code snippets on grid search tuning for SGD model

```
sgd_model_tuned.fit(X_train, y_train)
```

Figure 5.64 Code snippets on training tuned SGD model

Figure 5.65 shows the result of the best parameters of the tuned model using grid search. Figure 5.66 shows the implementation of the prediction value for the tuned SGD model and performance metrics on training set in figure 5.67

```
print(sgd_model_tuned.best_params_)
```

```
{'alpha': 0.0001, 'class_weight': {1: 0.7, 0: 0.3}, 'eta0': 100, 'learning_rate': 'constant', 'loss': 'hinge', 'penalty': 'l1'}
```

Figure 5.65 Result of best hyperparameter values of SGD model

```
y_pred_cv = cross_val_predict (sgd_model_tuned, X_train, y_train, cv=3)
```

Figure 5.66 Code snippets on predicting value with tuned SGD model

```
train_acc = accuracy_score(y_train, y_pred_cv)

print("Training accuracy: {:.4f}".format(train_acc))
print('precision = {:.4f}'.format(precision_score(y_train, y_pred_cv)))
print('recall = {:.4f}'.format(recall_score (y_train, y_pred_cv)))
print('f1 score = {:.4f}'.format(f1_score(y_train, y_pred_cv)))
```

```
Training accuracy: 0.7769
precision = 0.8029
recall = 0.7340
f1 score = 0.7669
```

Figure 5.67 Performance metrics result of tuned SGD model

Figure 5.68 shows the set of hyperparamter values for the grid search technique to take place for Neural Network model.

```
param_grid = {'hidden_layer_sizes': [(10,10,10), (50,100,50), (100,1), (15,15,15)],
          'activation': ['relu','tanh','logistic'],
          'alpha': [0.0001, 0.05],
          'learning_rate': ['constant','adaptive'],
          'solver': ['adam']}
```

Figure 5.68 Code snippets on hyperparameter values of Neural Network model

Figure 5.69 shows the code implementation for grid search tuning and figure 5.70 is the code snippets on training the tuned model.

```
nn_model_tuned = GridSearchCV(nnmodel, param_grid,cv=5, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
```

Figure 5.69 Code snippets on grid search tuning for Neural Network model

```
nn_model_tuned.fit(X_train, y_train)
```

Figure 5.70 Code snippets on training tuned Neural Network model

Figure 5.71 shows the result of the best parameters of the tuned model using grid search. Figure 5.72 shows the implementation of the prediction value for the tuned Neural Network model and performance metrics on training set in figure 5.73.

```
print(nn_model_tuned.best_params_)
```

```
{'activation': 'tanh', 'alpha': 0.05, 'hidden_layer_sizes': (100, 1), 'learning_rate': 'adaptive', 'solver': 'adam'}
```

Figure 5.71 Result of best hyperparameter values of Neural Network model

```
y_pred_cv = cross_val_predict (nn_model_tuned, X_train, y_train, cv=3)
```

Figure 5.72 Code snippets on predicting value with tuned Neural Network model

```
train_acc = accuracy_score(y_train, y_pred_cv)

print("Training accuracy: {:.4f}".format(train_acc))
print('precision = {:.4f}'.format(precision_score(y_train, y_pred_cv)))
print('recall = {:.4f}'.format(recall_score (y_train, y_pred_cv)))
print('f1 score = {:.4f}'.format(f1_score(y_train, y_pred_cv)))
```

```
Training accuracy: 0.8342
precision = 0.8049
recall = 0.8822
f1 score = 0.8418
```

Figure 5.73 Performance metrics result of tuned Neural Network model

After that, each of the tuned models will be used on testing set where the performance metrics result is shown. Figure 5.74 shows the code snippets on defining the testing result function.

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
def testing_result(train_acc, y_test, y_pred):
    print("Testing accuracy: {:.4f}".format(train_acc))
    print('precision = {:.4f}'.format(precision_score(y_test, y_pred)))
    print('recall = {:.4f}'.format(recall_score (y_test, y_pred)))
    print('f1 score = {:.4f}'.format(f1_score(y_test, y_pred)))

    # results
    accuracy = metrics.accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)
    print('----------------------------------------------------')
    print(metrics.classification_report(y_test, y_pred))
    print('----------------------------------------------------')
    print("Confusion matrix")
    confusion_matrix = metrics.confusion_matrix(y_test,y_pred)
    print(confusion_matrix)
```

Figure 5.74 Code snippets on defining the test result function

Figure 5.75 is the implementation of the tuned SVM model on testing set. The performance metrics result is shown at figure 5.76.

```
#Use the trained classifier to predict on test set
y_pred_test = svm_model_tuned.predict(X_test) # fit on test set
train_acc = accuracy_score(y_test, y_pred_test)

#Compute and print the precision, recall, f1 score of the prediction
print('SVM')
print('-------------------------')
testing_result(train_acc, y_test, y_pred_test)
```

Figure 5.75 Code snippets of testing the tuned SVM model

```
SVM
-------------------------
Testing accuracy: 0.8167
precision = 0.5000
recall = 0.9091
f1 score = 0.6452
Accuracy: 0.8166666666666667
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.97      0.80      0.88       147
           1       0.50      0.91      0.65        33

    accuracy                           0.82       180
   macro avg       0.74      0.85      0.76       180
weighted avg       0.89      0.82      0.83       180

-----------------------------------------------------
Confusion matrix
[[117  30]
 [  3  30]]
```

Figure 5.76 Result of testing performance metrics on tuned SVM model

Figure 5.77 is the implementation of the tuned KNN model on testing set. The performance metrics result is shown at figure 5.78.

```
#Use the trained classifier to predict on test set
y_pred_test = best_model.predict(X_test) # fit on test set
train_acc = accuracy_score(y_test, y_pred_test)

#Compute and print the precision, recall, f1 score of the prediction
print('KNN')
print('-------------------------')
testing_result(train_acc, y_test, y_pred_test)
```

Figure 5.77 Code snippets of testing the tuned KNN model

```
KNN
-------------------------
Testing accuracy: 0.8833
precision = 0.7308
recall = 0.5758
f1 score = 0.6441
Accuracy: 0.8833333333333333
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       147
           1       0.73      0.58      0.64        33

    accuracy                           0.88       180
   macro avg       0.82      0.76      0.79       180
weighted avg       0.88      0.88      0.88       180

-----------------------------------------------------
Confusion matrix
[[140   7]
 [ 14  19]]
```

Figure 5.78 Result of testing performance metrics on tuned KNN model

Figure 5.79 is the implementation of the tuned SGD model on testing set. The performance metrics result is shown at figure 5.80.

```
#Use the trained classifier to predict on test set
y_pred_test = sgd_model_tuned.predict(X_test) # fit on test set
train_acc = accuracy_score(y_test, y_pred_test)

#Compute and print the precision, recall, f1 score of the prediction
print('SGD')
print('-------------------------')
testing_result(train_acc, y_test, y_pred_test)
```

Figure 5.79 Code snippets of testing the tuned SGD model

```
SGD
-------------------------
Testing accuracy: 0.8167
precision = 0.5000
recall = 0.9091
f1 score = 0.6452
Accuracy: 0.8166666666666667
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.97      0.80      0.88       147
           1       0.50      0.91      0.65        33

    accuracy                           0.82       180
   macro avg       0.74      0.85      0.76       180
weighted avg       0.89      0.82      0.83       180

-----------------------------------------------------
Confusion matrix
[[117  30]
 [  3  30]]
```

Figure 5.80 Result of testing performance metrics on tuned SGD model

Figure 5.81 is the implementation of the tuned Neural Network model on testing set. The performance metrics result is shown at figure 5.82.

```
#Use the trained classifier to predict on test set
y_pred_test = nn_model_tuned.predict(X_test) # fit on test set
train_acc = accuracy_score(y_test, y_pred_test)

#Compute and print the precision, recall, f1 score of the prediction
print('Neural Network')
print('-------------------------')
testing_result(train_acc, y_test, y_pred_test)
```

Figure 5.81 Code snippets of testing the tuned Neural Network model

```
Neural Network
-------------------------
Testing accuracy: 0.8167
precision = 0.5000
recall = 0.9091
f1 score = 0.6452
Accuracy: 0.8166666666666667
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.97      0.80      0.88       147
           1       0.50      0.91      0.65        33

    accuracy                           0.82       180
   macro avg       0.74      0.85      0.76       180
weighted avg       0.89      0.82      0.83       180

-----------------------------------------------------
Confusion matrix
[[117  30]
 [  3  30]]
```

Figure 5.82 Result of testing performance metrics on tuned Neural Network model

5.4 Survey result and analysis

In order to educate and evaluate the awareness of the SMEs and collect first-hand data on the subject matter, this project included a survey where the target audience being SMEs employees. This is to recognize the public opinion and their knowledge on social engineering to be a reference while building the detection model. To signify varying levels of awareness, the items were separated into three categories which are knowledge, practices and solutions. The questionnaire consists of 23 questions and is organized into four parts. The first part acts as an invitation and consent form for surveyees in the acknowledgement in giving information. The second section collects the demographical information about the surveyees such as age range, gender, educational background and their position in the company. The third section contains comments intended to assess the level of awareness of social engineering attacks among Malaysia's SMEs. The final part requests a voluntary response on the records of the phishing emails dialogs to be used in the modelling process if possible.

A total of 50 respondents all of whom are employees in the SME industry completed this survey from 6 November 2021 to 22 November 2021. The demographic distribution is shown in table 5.1. It reflects that most of the respondent is around in their youth and hold a tertiary education. A big portion of surveyees are general staff as they are the ones who are most likely to receive harmful emails but there is also senior executive member that participated in the evaluation. In some cases, a leak from administrative members can be more fatal therefore it is important to educate on the matter equally.

Characteristic	**Total Respondents**	
	Frequency	Percentage
Gender		
Male	14	28%
Female	39	72%
Total	50	100%
Age		
18-25	33	66%
26-35	13	26%
36-45	3	6%
46 or older	1	2%
Total	50	100%
Educational background		
No schooling	0	0%
Primary	0	0%
Secondary	0	0%
Matriculation	1	2%
Certificate	0	0%
Diploma	3	6%
Bachelor’s degree	41	82%
Master’s degree	5	10%

Doctorate's degree	0	0%
Total	50	100%
Company position		
Employee	38	76%
Manager	8	16%
Director	2	4%
Owner	2	4%
Total	50	100%

Table 5.1 Demographic of respondents

The participants were questioned on whether or not they understood the term "social engineering." This is to have a general view of the respondents of their prior knowledge of social engineering. Based on table 5.2 and figure 5.83, it is seen that 70% of respondents do in fact have considerable understanding of it while 30% of them are oblivious to it. This result can be said that social engineering has begun raising attention on the employees but there is still a substantial group of people who have no psychological defense on this kind of attack.

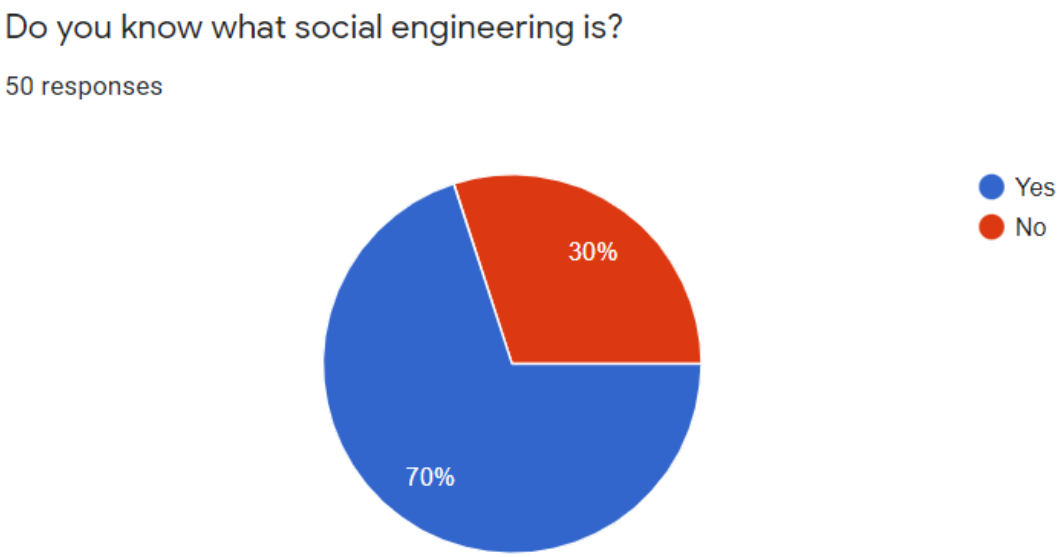


Figure 5.83 Prior knowledge of respondents on social engineering

Characteristic	Total Respondents	
	Frequency	Percentage
Social engineering knowledge		
Yes	35	70%
No	15	30%
Total	**50**	**100%**

Table 5.2 Prior knowledge of respondents on social engineering

Appendix B-1 presents all of the responses to the questions. The following analysis will only cover the important outcome from the survey that is required in later implementation. The next section evaluates the social engineering and information security knowledge of surveyees. They are educated on the meaning of SE and the various type of attacks.

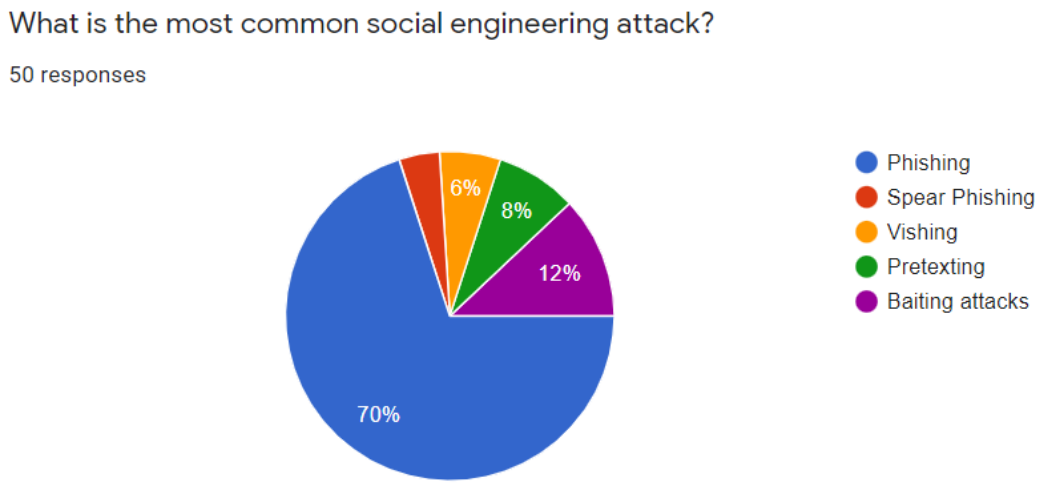


Figure 5.84 Response on the most common social engineering attack

From figure 5.84, when asked about the most common social engineering attack, 35 out of 50 respondents have chosen phishing email which partition 70% of the answer. This aligned with the preliminary research previously in which phishing attack cases have been reported to be the highest during these years. Therefore, phishing dialogue will be the cornerstone in the implementation of the detection model.

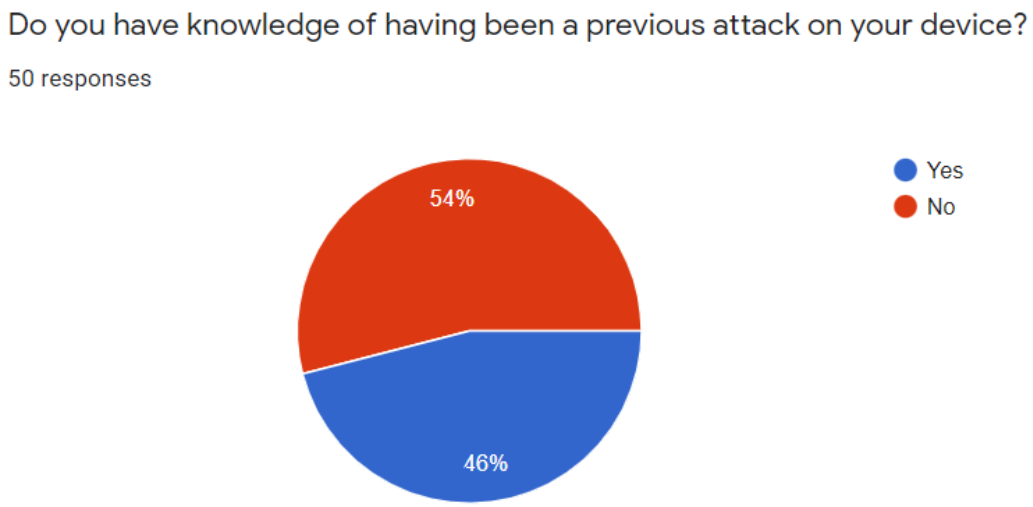


Figure 5.85 Respondents awareness of being attacked without consent

Based on figure 5.85, the chart shows that there are 27 out of 50 respondents stated that they have not been in a social engineering attack. This is an interesting result where despite the arising of phishing attack cases, over half of the response contradicts the research finding. A reasonable deduction would be these respondents had actually been fallen to the attack by cybercriminals without any consent. This would have explained the contradictory facts from both sources. This result will also be in consideration on the parameter adjustment in choosing misspelled words as feature and the evaluation of the penalty on misspelled words. Misspelled words are used to increase the chances of penetrating through the filters. Furthermore, they act as a filter itself to find the appropriate target. The people who reply to those misspelled dialogues are statistically less knowledgeable and more likely to follow the steps of the attackers.

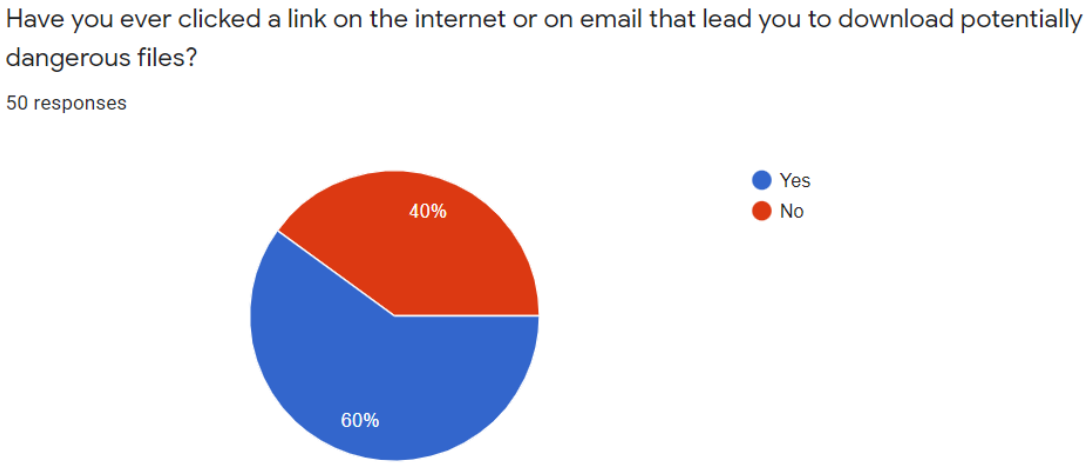


Figure 5.86 Response on clicking unverified link in the past

In the subsequent section on information security practices, the respondents are given some questions on their usual approach when handling certain security procedures to evaluate their awareness as the parameter in the implementation stage. One of them is about the history of whether they had clicked on a link that direct download suspicious files unknowingly or other shady websites shown in figure 5.86. About 60%, which is 30 out of 50 respondents claimed they had indeed done it before. This may be due to the lack of knowledge and alertness on checking those links beforehand in a stressful environment. This response is noted and integrated as a feature and also assessing the penalty value during model building.

Figure 5.87 Respondents' password sharing behavior

Figure 5.87 is another question about the awareness of the respondents on sharing sensitive information such as passwords to other people in a different scenario. A total of 88% which rounds to 44 gave a strong stand in denying to share passwords to others. It can be seen that even without undergoing certain security practices incentive training programs, most employees have the basic understanding and sensitivity when requested for confidential information regardless of the scenario they are in. This is an important outcome and deciding factor on the another feature and the weighting of blacklisted words during the implementation of the model.

From: Greg
To: Jussi
Subject: need to ssh into rootkit
im in europe and need to ssh into the server. can you drop open up
firewall and allow ssh through port 59022 or something vague?
and is our root password still 88j4bb3rw0cky88
or did we change to
88Scr3am3r88 ?
thanks

Figure 5.88 Example of social engineering attack dialogue

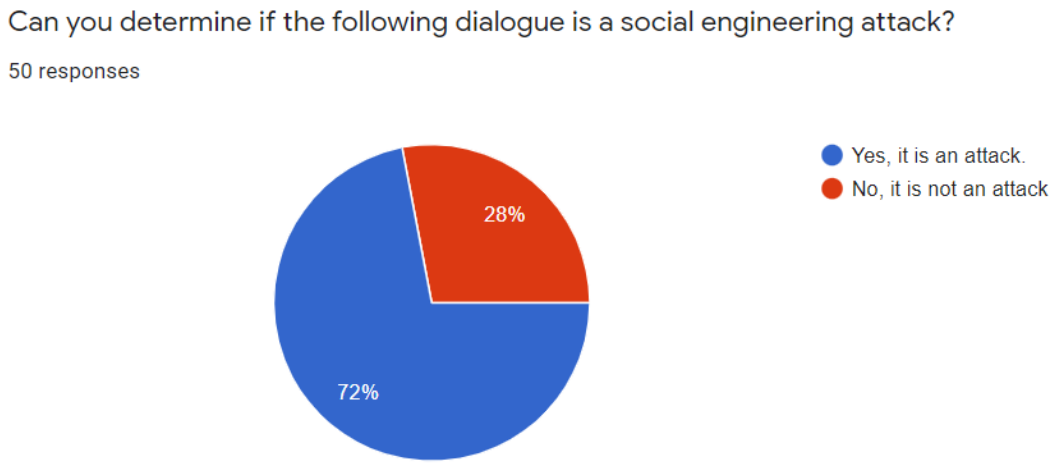


Figure 5.89 Respondent answer on differentiating social engineering dialogue

The final question shown in figure 5.89 quizzed the respondent on whether the dialogue in figure 5.88 is an example of a social engineering attack. This dialogue is retrieved from [9] in the analysis of a real life attack on a well-known technology security company. There are 72% of respondents that gave the correct answer in which they agree that it is indeed an attack. The other 18 respondents answered wrong stating that it is not. They failed to realize the urgency, unavailability and strong intent elements in the context which are tactics commonly used by social engineering attackers. The results of this response will be used as additional feature and in the weighting of the intent adjectives in dialogue in the model implementation stage.

5.5 Concluding remark

The chapter has covers the implementation details on the experiments of models from setting up the environment by step by step configuration to the generation of the final result. It also included a survey to educate SMEs employees and evaluate their awareness on social engineering. The survey result then undergoes analysis as reference on setting up certain parameters during model building. Each of the model's performance result is shown and summarized. The main setback is the limited dataset used in the model where it constraints the model in finding more correlation within data thus unable to predict more accurately. Another challenge would be the occurrence of imbalanced dataset. However, these problems are tackled by the addition of another set of preprocessed data and SMOTE technique.

CHAPTER 6: SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

6.1.1 Result analysis on confusion matrix

Confusion matrix is defined as a matrix that is normally used to evaluate the performance of a classification model. It shows of the summary of number of correct and incorrect prediction results on a classification problem with count values and categorized by each class. The matrix has four components: number of positive samples that are correctly predicted as positive is labeled as true positive (TP), number of positive samples that are falsely predicted as negative is labeled as false negative (FN), number of negative samples that are correctly predicted as negative is labeled as true negative (TN) and number of negative samples that are falsely predicted as positive is labeled as false positive (FP). This will give a better representation of result on the model. Table 6.1 shows the confusion matrix result for each machine learning model.

	SVM	KNN	SGD	Neural Network
True Positive	125	146	125	117
False Positive	22	1	22	30
False Negative	6	17	6	3
True Negative	27	16	27	30

Table 6.1 Confusion matrix result for each machine learning model

From table 6.1, the SVM model has 125 samples marked as TP, 22 samples as FP, 6 samples as FN and 27 samples as TN. The KNN model identify the most number of positive samples correctly which is 146 and having the least FP and TN value which is 1 and 16 respectively. However, it has the highest number of FN which is 17 samples among all the models. The SGD model shows the same result as the SVM model due the similarity of their properties with the different that it treats data in batches and perform gradient descent training. The neural network shows the lowest number value on TP which is 117 but the highest number of samples on TN which is 30. It also has the highest number of FP samples which is 30 and lowest FN value which is 3 samples. In hindsight, the KNN model seems to be the one performing the best. Further analysis is carried out in the next section.

6.1.2 Result analysis of F1 score, Recall and precision

In order to evaluate the quality of the algorithm on this classification problem, several performance metrics are used in the analysis. These metrics are precision score, recall score and f1 score. Precision score is defined in equation 1. Recall is defined in equation 2 and f1 score is defined in equation 3. Table 6.2 shows the result of the aforementioned metrics on each model.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$F1 = \frac{TP}{TP+\frac{FN+FP}{2}} \quad (3)$$

	SVM	KNN	SGD	Neural Network
Precision	0.5510	0.9412	0.5510	0.500
Recall	0.8182	0.4848	0.8182	0.9091
F_1 Score	0.6585	0.6400	0.6585	0.6452

Table 6.2 Result of precision, recall and F_1 scores on each model.

Precision also known as positive predictive value (PPV) measures the accuracy of the positive prediction. It is used to evaluate the reliability on samples that are predicted as true. The SVM model has a precision of 0.5510 which is same as SGD as both model has the same confusion matrix distribution. The KNN model has a very high precision score of 0.9412 which lead to believe it is very reliable in predicting positive samples. The neural network has the lowest precision score of 0.500.

Recall or sensitivity measures the ratio of positive samples that are correctly labeled. It is used to evaluate on how many positive samples in the dataset are correctly detected. Due to the same reason as before, the recall result is the same for SVM model and SGD model which is 0.8182. KNN model has the lowest recall result with a value of 0.4848 while neural network model has compensated the precision result before by having the highest recall result which is 0.9091. This is a very normal as there exist a precision/recall trade off where the increase precision usually will lower the recall rate and vice versa.

F_1 score combines the precision and recall into a single score for a clearly and more unbiased representation on the model performance. The formula is derived from the harmonic mean of precision and recall. The metric favors classifiers that have similar precision and recall and it is used when both metrics are equally important. As usual, the SVM model and SGD model have the same F_1 score of 0.6585 which is also the highest compared to others. The KNN model has the lowest F_1 result of 0.6400 while neural network has the F_1 score of 0.6452. Overall, the F_1 score results are similar across all model with only slight differences.

6.1.3 Result analysis of best hyperparameter

In the process of the experiment, there is a stage of fine tuning the model after training and validating them. Fine tuning hyperparameter is process that takes the trained model and tweak it further in order to obtain the most optimal hyperparameter values that will lead to a better performance. The following tables show the summary of the hyperparameter value sets for each model. Table 6.3 shows the set of hyperparamter values for SVM. Table 6.4 shows the set of hyperparamter values for KNN. Table 6.5 shows the set of hyperparamter values for SGD. Table 6.6 shows the set of hyperparamter values for neural network.

SVM	
Parameter	Value
c	1, 10, 100, 500, 1000
kernel	Linear, rbf
gamma	1, 0.1, 0.01, 0.001, 0.0001

Table 6.3 Hyperparameter values set for SVM model

KNN	
Parameter	Value
leaf size	Range between 1 and 50
n_neighbors	Range between 1 and 30
p	1, 2

Table 6.4 Hyperparameter values set for KNN model

SGD	
Parameter	Value
loss	hinge, log, squared_hinge
penalty	l2, elasticnet
alpha	0.0001, 0.001, 0.01, 0.1
learning_rate	constant, optimal
eta0	1, 10, 100

Table 6.5 Hyperparameter values set for SGD model

Neural Network	
Parameter	Value
hidden layer size	(10, 10, 10), (50, 100, 50), (100, 1), (15, 15, 15)
activation	relu, tanh, logistic
alpha	0.0001, 0.05
learning rate	constant, adaptive
solver	adam

Table 6.6 Hyperparameter values set for Neural Network model

After that, the experiment uses GridSearchCV which attempts to loop through all the combinations of the predefined values passes in the dictionary and perform evaluation for each combination using cross-validation on the model. In the end, the best parameters can be selected from the listed hyperparameters. The best hyperparameter value is shown in the following tables.

SVM	
Parameter	Best Value
c	500
kernel	rbf
gamma	0.0001

Table 6.7 Best hyperparameter value for SVM model

KNN	
Parameter	Best Value
leaf size	30
n_neighbors	22
p	1

Table 6.8 Best hyperparameter value for KNN model

SGD	
Parameter	Best Value
loss	log
penalty	l2
alpha	0.0001
learning_rate	optimal
eta0	1

Table 6.9 Best hyperparameter value for SGD model

Neural Network	
Parameter	Best Value
hidden layer size	(100, 1)
activation	tanh
alpha	0.05
learning rate	adaptive
solver	adam

Table 6.10 Best hyperparameter value for Neural Network model

6.1.4 Result analysis of ROC after cross validation

Further analysis is carried out to validate if the model is actually training. One of the methods is the receiver operating characteristic (ROC) curve where it illustrates a graph that shows the performance of classifiers on all classification threshold. The curve consists of two parameters which is true positive rate and false positive rate.
The formula of false positive rate is defined in equation 4.

$$FPR = \frac{FP}{FP + TN} \quad (4)$$

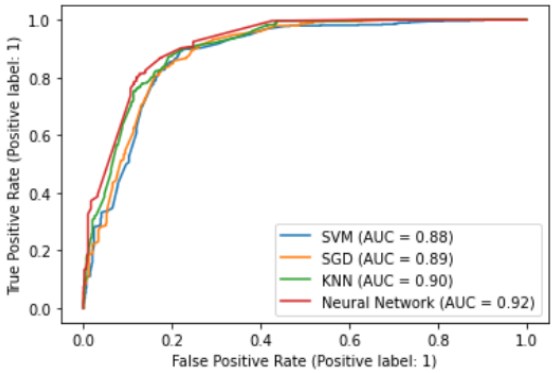


Figure 6.1 ROC curve for each model after cross validation

Figure 6.1 shows the plot diagram of the ROC curve on four models. Noted that this process is performed using the training set. It can be seen that all of the model are performing the training process without any abnormally.

6.2 Testing Setup and result

The testing result for four of the machine learning models are shown in table 6.11.

	SVM	KNN	SGD	Neural Network
Testing Accuracy	0.8444	0.9000	0.8444	0.8167

Table 6.11 Result of accuracy score for each model

The accuracy score is defined in equation 5. Accuracy is typically used to describe the number of correctly predicted data and how the model perform across all classes. Testing accuracy represents the result of the trained model on unseen data.

$$Accuracy = \frac{TP + TN}{TP+TN + FP + FN} \quad (5)$$

From table 6.11, it is inferenced that KNN model achieve the highest testing accuracy score which is 0.90. This is followed by the SVM model and SGD model whereby both of them achieved the same score of 0.8444. The model having the lowest accuracy is neural network which is 0.8167.

However, due to the nature of the dataset, there exists an imbalanced condition on the data distribution which lead the result on test accuracy less significant. An additional ROC AUC score metric is needed to better evaluation. Table 6.12 shows the ROC AUC score on all model.

	SVM	KNN	SGD	Neural Network
Average AUC	0.8343	0.7390	0.8343	0.8525

Table 6.12 ROC AUC scores for each model

From table 6.12, the KNN model which obtained the highest test accuracy encounter a poor performance on this metric where it only had a score of 0.8343. This lead to believe that overfitting of data may have occurred where a imbalanced data could have affect the classification threshold on KNN algorithm. In contrast, the neural network which has the lowest test accuracy before achieve a ROC AUC score of 0.8525 which implied that the model is performing just fine. Both the SVM model and SGD model have the same score of 0.8343.

6.3 Project Challenges

There are several implementation challenges throughout the research project. One of them is the substitution of API in checking whether the URL is malicious. The original novelty uses Web of Trust (WOT) in their paper for evaluation. However, the service required a subscription of $39 per month which is too costly. The alternative option will be the Virus Total (VT) service that is free of charge but some manipulation need to be done on the function due to different output. Next would be the issues faced during the stage of data acquisition. Due to the covid-19 pandemic, there has been a rise in business sales where all online SMEs are on a busy schedule and unfortunately cannot find time to arrange possible social engineering attacks' dialogs. Besides that, there is also much confidential information which is sensitive to the public hence no company is willing to give out the dialogs. Moreover, most of the research and data on the internet

focus on phishing websites instead of phishing dialogs. This challenges the research in finding related information on the topic.

6.4 Objectives Evaluation

After analyzing thoroughly on each metric of the models, the most optimal model is chosen for the one of the project objective. On hindsight, the model with the best testing accuracy is the KNN model with 90%. However, there is an important criterion that should not be ignored which is the imbalance nature of dataset whereby causes the accuracy performance metric to carry less weight when selecting the best model.
An imbalance of dataset is commonly occurred such as to detecting a spam email from ham email as spam email usually has much lower samples. This example shares similar properties to this paper in detecting social engineering attacks. The statement is also validated with the low ROC AUC score of the KNN model.

Therefore, another measure which is suitable for such cases would be the recall metric. A higher recall score means a higher ratio on predicting correctly on positive samples. SME companies should avoid false-negatives result at all cost since a leak on the security boundary could possibly lead to a series of lethal consequences. The neural network deep learning model in this case would be ideal model to be selected for its high recall score of 0.9091.

6.5 Concluding Remarks

In brief, this chapter has summarized the performance metrics result of each model. Every component is discussed and analyzed deeply on how it measured the performance of the model. The best hyperparameter value result is also recorded for future references in the fine tuning process. A best model was then being selected with solidified reason.

CHAPTER 7: CONCLUSION AND RECOMMENDATION

7.1 Conclusion

In conclusion, this paper presented a social engineering exploitation detection method among Malaysia's SMEs based on previous novelty idea. It also includes a survey to educate and evaluate the social engineering attack awareness of the employees of Malaysia's SMEs. The parameter and model building will be based on their information security knowledge and remarks on the historical attack. The approach first processes a conversation before generating a dataset that can be used for categorization. Four machine learning techniques are used for this project which are SVM, SGD, KNN and Neural Network. The KNN model achieved the highest accuracy among the models which is 90%. However, after deeper analysis, it is found that the neural network architecture is more suitable candidate as the best model due to its high recall score and the nature of the dataset itself. This research can be used as a baseline for existing models to determine whether they could be automated by the utilization of natural language processing and artificial neural network will help to eliminate human errors in the detection of social engineering attacks.

7.2 Recommendation

Although the final result is not exceptionally good, it is believed that improvements can be made in increasing the performance of the model. The advancement of technologies especially among the deep learning field which certain breakthroughs has shown new possibilities on its implementation. The standard neural network model is only the foundation basis in the world of deep learning where it is also able to generate a better performance by having a large dataset which is true for the increasing cases of social engineering attack in recent years. Therefore, exploration and implementation of deep learning algorithms and the finding of new datasets will be the main aim in future work. Moreover, there are plans that can be made in to have additional features to the dataset by expanding the number of attributes necessary. It is also targeted to develop a more balanced dataset and the application of algorithms in real time environment.

REFERENCES

[1] M. Bezuidenhout, F. Mouton and H. Venter, "Social engineering attack detection model: SEADM," in *Information Security for South Africa (ISSA)*, 2010.

[2] P. Kale, S. Kashiwant, N. Kamble, S. Awate and S. Tidke, "Automatic Detection of Social Engineering Attacks Using Dialo," *Journal of Computer Engineering*, vol. 17, no. 6, pp. 76-78, 2015.

[3] M. Hoeschele and M. Rogers, "Detecting Social Engineering," in *IFIP International Conference on Digital Forensics*, 2005.

[4] M. Lansley, N. Polatidis and S. Kapetanakis, "SEADer: A Social Engineering Attack Detection method based on Natural Language Processing and Artificial Neural Networks," in *Computational Collective Intelligence*, Springer, 2019, pp. 686-696.

[5] S. Balaji and K. Punitha, "Framework for Detection of Phishing Social Engineering Attacks," *The International journal of analytical and experimental modal analysis*, vol. 12, no. 10, pp. 732-737, 2020.

[6] J. Pettit, "The State of Security," Tripwire, Mar 22, 2022. Accessed on Nov. 8, 2021. [Online]. Available: https://www.tripwire.com/state-of-security/security-awareness/5-social-engineering-attacks-to-watch-out-for/

[7] L. Nathan, "Phishing attacks rising since pandemic struck," The Malaysian Reserve, Sep. 2, 2021. Accessed on Nov. 11, 2021. [Online]. Available: https://themalaysianreserve.com/2021/09/02/phishing-attacks-rising-since-pandemic-struck/

[8] J. Moh, "Ignore cyberthreats at your peril, Malaysian businesses told," theSundaily, Oct. 10, 2021. Accessed on Nov. 9, 2021. [Online]. Available: https://www.thesundaily.my/business/ignore-cyberthreats-at-your-peril-malaysian-businesses-told-EG8448475

[9] B. Gyunka and O. C. Abikoye, "Analysis of Human Factors in Cyber Security: A Case Study of Anonymous Attack on Hbgary," January 2017. [Online].

[10] "Identify Phishing E-Mail," Malaysian Communications And Multimedia Commission. Accessed on Nov. 24, 2021. [Online]. Available: https://www.mcmc.gov.my/en/faqs/phishing-attack/2-identify-phishing-e-mail

[11] F. Mouton, M. M. Malan, L. Leenen and H. Venter, "Social Engineering Attack Framework," in *Information Security for South Africa*, Johannesburg, 2014.

[12] F. Salahdine and N. Kaabouch, "Social Engineering Attacks: A Survey," April 2, 2019. Accessed on Aug. 20, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/11/4/89

[13] D. Shetty, "Social Engineering: The Human Factor,". Accessed on Aug. 20, 2021. [Online]. Available: http://index-of.es/Failed-attack-techniques/Social%20Engineering.pdf.

[14] Q. Tariq, "Saifuddin: More cybercrime reported during pandemic," The Star, June 3, 2021. [Online]. Available: https://www.thestar.com.my/tech/tech-news/2021/06/03/saifuddin-more-cybercrime-reported-during-pandemic.

APPENDIX A

A.1 Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 2**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Revise on FYP1 report
- Reading FYP2 guideline
- Revise on FYP1 code snippets
- Research of additional machine learning model

2. WORK TO BE DONE

- Finding additional machine learning model
- Finding dataset

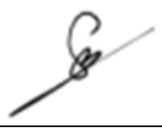
3. PROBLEMS ENCOUNTERED

- Hard to find additional dataset

4. SELF EVALUATION OF THE PROGRESS

- Time management

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Selecting on machine learning model
- Finding dataset

2. WORK TO BE DONE

- Implementing machine learning model found
- Model cross validation

3. PROBLEMS ENCOUNTERED

- Not much dataset on phishing dialogue

4. SELF EVALUATION OF THE PROGRESS

- Time management

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Additional model implementation
- Implementation of additional dataset

2. WORK TO BE DONE

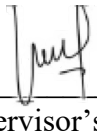
- Model training
- Model cross validation
- Model fine-tuning

3. PROBLEMS ENCOUNTERED

- Imbalance dataset
- Debugging on implementation error

4. SELF EVALUATION OF THE PROGRESS

- Time management



Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 8**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Model training
- Model cross validation
- Check if the models are actually training

2. WORK TO BE DONE

- Model fine-tuning
- Model testing

3. PROBLEMS ENCOUNTERED

- Debugging on implementation error

4. SELF EVALUATION OF THE PROGRESS

- Time management

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 10**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Continuation on report writing
- Model fine-tuning on hyperparameters
- Model testing

2. WORK TO BE DONE

- Analysis on the result of performance metric
- Formatting report

3. PROBLEMS ENCOUNTERED

- Debugging code of the model

4. SELF EVALUATION OF THE PROGRESS

- Time management

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: SIA KEN YEN 18ACB04562	
Supervisor: Ts Dr Vasaki a/p Ponnusamy	
Project Title: Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning	

1. WORK DONE

- Formatting report
- Update latest references formatting
- Send report draft for checking
- Analysis on result and discussion of the performance of various model

2. WORK TO BE DONE

- Completed FYP report
- Presentation

3. PROBLEMS ENCOUNTERED

- Urgency in finishing report due to deadline

4. SELF EVALUATION OF THE PROGRESS

- Time management

Supervisor's signature

Student's signature

A.2 Poster

Social Engineering Exploitation Detection

SEED

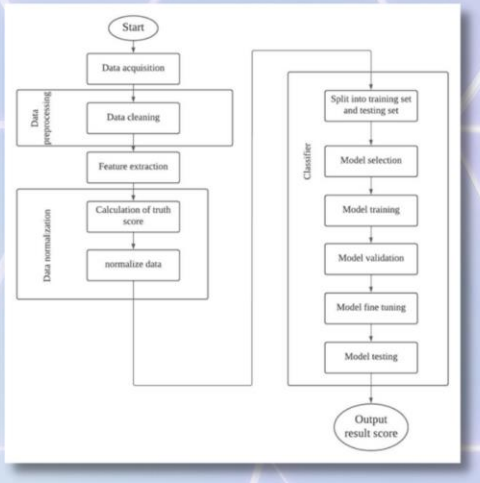
in Malaysia's SMEs using Machine Learning

INTRODUCTION

This project will be building a model to detect exploitation of social engineering using machine learning

METHODOLOGY

The detction model will integrate with machine learning technique and focus on phishing attacks based on spelling checking



	SVM	KNN	SGD	Neural Network
Testing Accuracy	0.8444	0.9000	0.8444	0.8167
Precision	0.5510	0.9412	0.5510	0.5000
Recall	0.8182	0.4848	0.8182	0.9091
F1 Score	0.6585	0.6400	0.6585	0.6452

RESULTS

DISCUSSION

Four machine learning methods had been implemented for the detection model. Although KNN achieved the highest test accuracy, Neural Network is a more optimal model due to high recall score.

CONCLUSION

The best model is neural network. This research provides a baseline framework to eliminate human error on detecting SE attacks. Future work will focus on expanding dataset and execution in real time environment.

SIA KEN YEN | 18ACB04562
Supervised by Ts Dr Vasaki a/p Ponnusamy

APPENDIX B

B.1 Survey Questions and Answers

Category	Characteristic	Total Respondents	
		Frequency	Percentage
Social engineering and information security knowledge	What is the most common social engineering attack?		
	Phishing	35	70%
	Spear Phishing	2	4%
	Vishing	3	6%
	Pretexting	4	8%
	Baiting attacks	6	12%
	Total	50	100%
	Attackers cannot target me; my computer has no value to them.		
	Yes	19	38%
	No	31	62%
	Total	50	100%
	Would you recognize if your work computer is being hacked?		
	Yes	21	42%
	No	29	58%
	Total	50	100%
	Do you know how to tell if your computer has been hacked or infected?		
	Yes	21	42%
	No	29	68%
	Total	50	100%

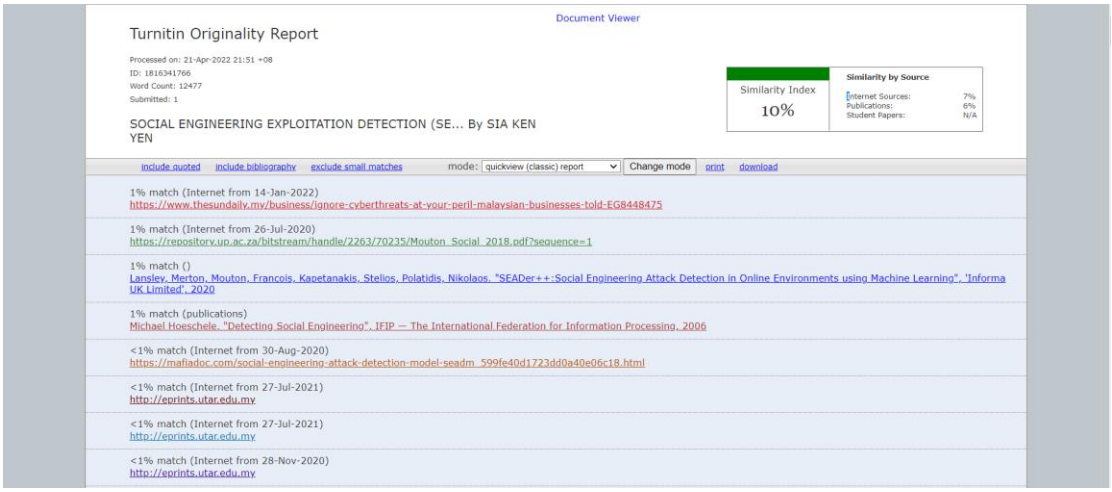
	Do you have knowledge of having been a previous attack on your device?		
	Yes	23	46%
	No	27	54%
	Total	50	100%
Information security practices	Have you logged private accounts using public computers such as a library or hotel?		
	Yes	34	68%
	No	18	32%
	Total	50	100%
	Have you ever found a virus or Trojan on your computer at work?		
	Yes	17	34%
	No	33	66%
	Total	50	100%
	How careful are you when you open an attachment in email?		
	I always ensure it is from someone I know or someone I am expecting an email from	32	64%
	I open the attachment as long as the sender is familiar to me	7	14%
	I open attachments regardless of whether I know the sender or not	11	22%

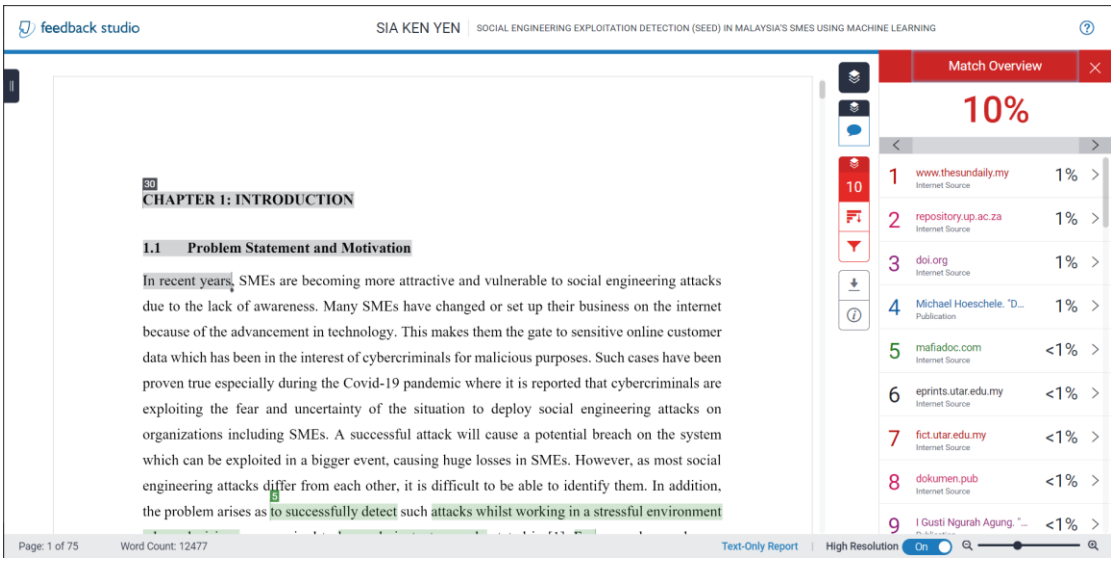
	Total	50	100%
	Have you ever clicked a link on the internet or on email that lead you to download potentially dangerous files?		
	Yes	30	60%
	No	20	40%
	Total	50	100%
	Have you ever had you email account hacked or stolen?		
	Yes	29	58%
	No	21	42%
	Total	50	100%
	Do you usually share your passwords with anyone?		
	Yes, with everybody	2	4%
	Yes, but only with members in the company organization	4	8%
	No, I do not share my passwords with anyone	44	88%
	Total	50	100%
	How do you usually form your passwords?		

	I usually form my passwords using a combination of letters, numbers, and special characters.	29	58%
	I usually form my passwords using my personal information such as name and date of birth	21	42%
	Total	50	100%
	Is the USB considered a transferor of viruses?		
	Yes	43	86%
	No	7	14%
	Total	50	100%
Technical security solutions	Is the firewall on your computer enabled?		
	Yes	46	92%
	No	4	8%
	Total	50	100%
	Is there an anti-virus software on your device?		
	Yes	43	86%
	No	7	14%
	Total	50	100%
	Are you updating your anti-virus software regularly?		
	Yes	32	64%
	No	18	36%
	Total	50	100%
	How often do you scan your device?		

	Weekly	10	20%
	Monthly	20	40%
	Once every 6 months	9	18%
	Yearly	4	8%
	I do not scan my device	7	14%
	Total	50	100%
	Can you determine if the following dialogue is a social engineering attack?		
	Yes, it is an attack.	36	72%
	No, it is not an attack	14	28%
	Total	50	100%

PLAGIARISM CHECK RESULT





Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date:	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	SIA KEN YEN
ID Number(s)	18ACB04562
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	Social Engineering Exploitation Detection (SEED) in Malaysia's SMEs using Machine Learning

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 10 % **Similarity by source** Internet Sources: 7 % Publications: 6 % Student Papers: N/A %	Checked and verified
Number of individual sources listed of more than 3% similarity: 0	Checked and verified
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Vasaki Ponnusamy

Date: 22/04/22

Signature of Co-Supervisor

Name:

Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB04562
Student Name	SIA KEN YEN
Supervisor Name	TS DR VASAKI A/P PONNUSAMY

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

_____

(Signature of Student)
Date: 21.04.2022