

MONEY LAUNDERING DETECTION OF SUSPICIOUS TRANSACTION
USING MACHINE LEARNING ALGORITHM

NUR ADRIANA BATRISYIA BINTI MOHD SUBRI

UNIVERSITI TEKNOLOGI MALAYSIA

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction to Methodology Framework in Money Laundering Detection

This chapter outlines the activities in lifecycle of a data science project to achieve the aim and objectives of this research by using supervised machine learning algorithm. It focuses on the use of Support Vector Machines and Decision Tree to enhance the prediction of suspicious transactions to predict suspicious transactions in money laundering. This project lifecycle revolves around six phases and illustrated as per Figure 3.1.

- i. **Phase 1: Problem Identification** to understand the current challenges in money laundering detection.
- ii. **Phase 2: Data Collection** which involves obtaining the synthetic transactions dataset that was developed by another research.
- iii. **Phase 3: Data Preparation** by cleaning the data, transforming the features, performing Exploratory Data Analysis (EDA) to identify the correlations and patterns, and handling imbalance dataset using under sampling techniques.
- iv. **Phase 4: Model Training** by using optimized Support Vector Machines and Decision Tree in which the models are tuned using the best kernels and hyperparameters.
- v. **Phase 5: Model Evaluation** to test the model performance on detecting suspicious transactions.
- vi. **Phase 6: Model Findings and Presentation** to highlight the key findings through clear visualizations dashboards.

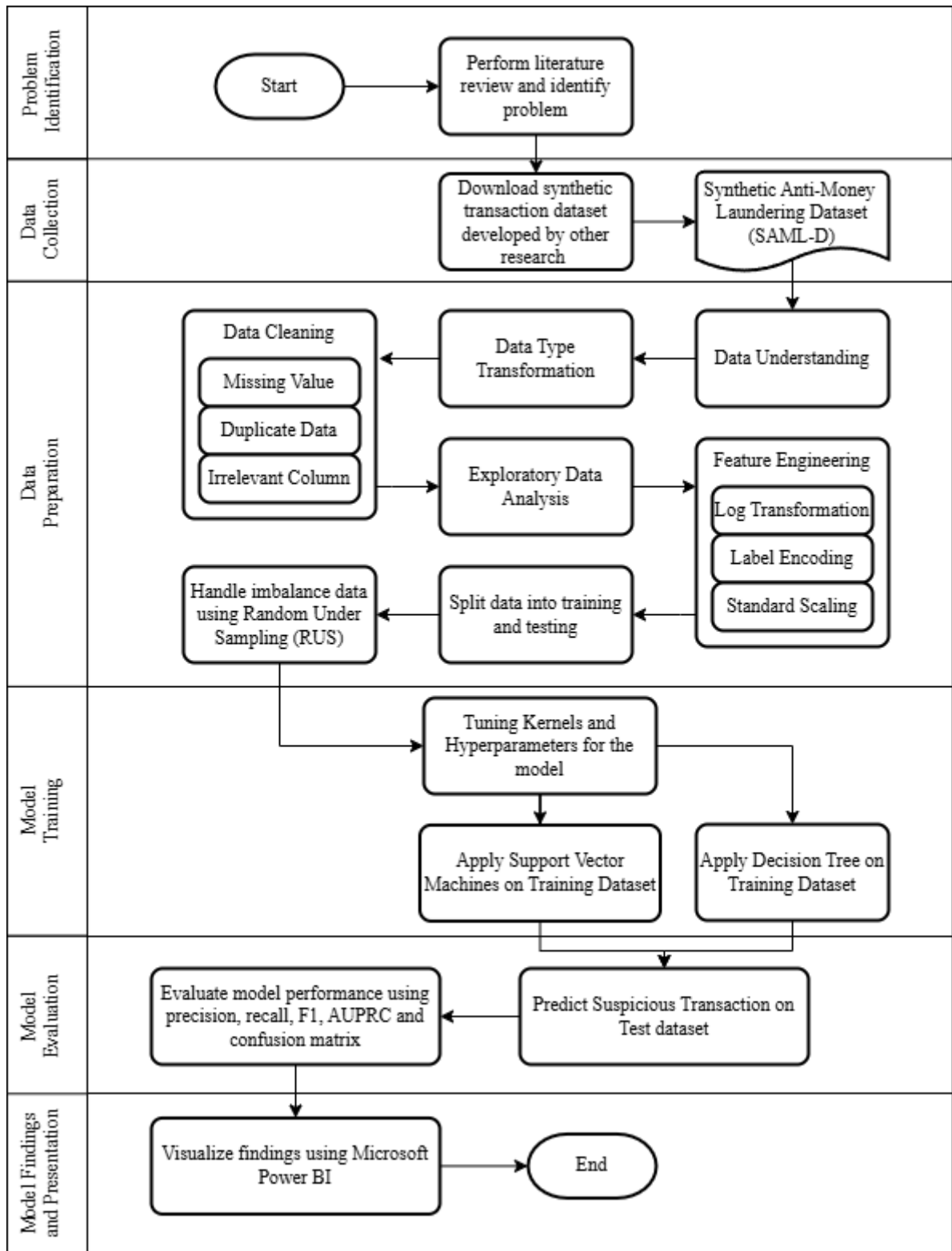


Figure 1.1: Flowchart to predict suspicious transactions for money laundering detection

3.2 Problem Identification

It is essential to have a good understanding of the problems to set a clear objectives and goals for the project. Based on literature review, the current and major challenge to combat money laundering crimes in Malaysia is that the current techniques to detect money laundering activities is not effective and powerful enough to identify the complex and hidden schemes used by criminals. Hence, there is a need to develop an effective machine learning approach to maintain financial integrity in Malaysia.

A thorough literature review also helps to grasp a deep understanding on money laundering concepts such as the indicators of suspicious transactions, existing patterns or schemes of money laundering, and regulatory framework on Anti-Money Laundering (AML). This activity also beneficial to identify the limitations on current approaches and provides insights on potential techniques to be experimented.

3.3 Data Collection

It is difficult to obtain real transaction dataset because it is not easily accessible due to legal and privacy reasons. Therefore, a synthetic transaction dataset known as SAML-D is used for this research. It was generated from research paper entitled “Enhancing Anti-Money Laundering: Development of a Synthetic Transaction Monitoring Dataset” by B. Oztas et al. (2023) and the authors made it available at Kaggle. The dataset was extracted in CSV format with size of almost 1GB. The key highlights of SAML-D are it incorporates geographic locations that involves high-risk countries, high-risk payment types, and wider range of typologies compared to other synthetic dataset which adds the complexity and brings greater realism to the dataset. In summary, this dataset has a total of 9,504,851 entries with 12 attributes as described in Table 3.1.

Table 3.1: Data Description for each Attributes

No.	Attributes	Description
1.	Time	The time of when the transaction occurred based on 24-hour system in format HH:MM:SS
2.	Date	The date of when the transaction occurred in format YYYY-MM-DD
3.	Sender_account	The bank account number of the sender
4.	Receiver_account	The bank account number of the receiver
5.	Amount	The amount of money that is being transferred in the transaction
6.	Payment_currency	The currency of where the money originated and related with the information on Sender_bank_location
7.	Received_currency	The currency of where the money is transferred to and related with the information on Receiver_bank_location
8.	Sender_bank_location	The country in which the money originated from
9.	Receiver_bank_location	The country in which the money is being transferred to
10.	Payment_type	The ways to transfer the money between sender and receiver
11.	Is_laundering	Indicates whether it is a laundering transaction or a normal transaction in a binary format
12.	Laundering_type	The patterns of cash flow involve in the transaction

3.4 Data Preparation

This phase involves the steps to prepare dataset before training with machine learning which includes:

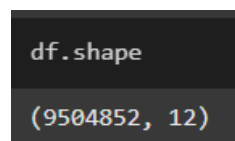
- i. Data understanding
- ii. Data type transformation
- iii. Data cleaning
- iv. Exploratory data analysis
- v. Feature Engineering
- vi. Split Train-Test
- vii. Handling class imbalance

In this section, it will focus on the first three subtopics which are preliminary analysis, data type transformation, and data cleaning. The rest of the subtopics will be covered in Chapter 4.

3.4.1 Data Understanding

Before going into a more in-depth analysis, it is important to explore the dataset to gain more understanding especially on the attributes and data types. The list of functions that are used to explore the data together with its applications are as below.

- i. **.shape**
This function provides the dimensionality of the dataset. Based on Figure 3.2, this dataset has 9504852 rows and 12 columns.



```
df.shape
(9504852, 12)
```

Figure 3.2: Shape of the dataset

ii. **.info()**

This function gives general overview on the dataset. Based on the output in Figure 3.3, the dataset has a total of 9504852 entries with 12 columns. Four columns are numerical variables which indicated by data type 'int64' and 'float64'. Meanwhile, the rest are categorical variables which indicated by data type 'object'.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9504852 entries, 0 to 9504851
Data columns (total 12 columns):
#   Column                Dtype
---  -
0   Time                  object
1   Date                  object
2   Sender_account        int64
3   Receiver_account      int64
4   Amount                float64
5   Payment_currency      object
6   Received_currency     object
7   Sender_bank_location  object
8   Receiver_bank_location object
9   Payment_type          object
10  Is_laundering          int64
11  Laundering_type        object
dtypes: float64(1), int64(3), object(8)
memory usage: 870.2+ MB
```

Figure 3.3: General overview of the dataset

iii. **.unique()**

This function list out unique values of the attributes. Output in Figure 3.4 shows that 'Payment Type' consist of seven different transaction methods which are credit card, debit card, cheque, automated clearing house (ACH) transfers, cross-border, cash withdrawal, and cash deposit.

```
print(df['Payment_type'].unique())

['Cash Deposit' 'Cross-border' 'Cheque' 'ACH' 'Credit card' 'Debit card'
 'Cash Withdrawal']
```

Figure 3.4: List of payment types

As per Figure 3.5, 'Laundering_type' has a total of 28 different typologies. It represents the patterns of transactions which split between 11 normal transactions and 17 suspicious transactions.

```
print(df['Laundering_type'].unique())

['Normal_Cash_Deposits' 'Normal_Fan_Out' 'Normal_Small_Fan_Out'
 'Normal_Fan_In' 'Normal_Group' 'Normal_Cash-Withdrawal'
 'Normal_Periodical' 'Normal_Foward' 'Normal_Mutual' 'Smurfing'
 'Normal_Plus_Mutual' 'Normal_single_large' 'Cash-Withdrawal'
 'Behavioural_Change_2' 'Structuring' 'Behavioural_Change_1'
 'Layered_Fan_In' 'Layered_Fan_Out' 'Scatter-Gather' 'Cycle' 'Fan_In'
 'Stacked Bipartite' 'Over-Invoicing' 'Deposit-Send' 'Single_large'
 'Bipartite' 'Gather-Scatter' 'Fan_Out']
```

Figure 3.5: List of laundering types

'Sender_bank_location' and 'Receiver_bank_location' represent the country where the bank located at. There are a total of 18 countries involves in this dataset including UK, UAE, Spain, France, USA, Mexico, Albania, Turkey, Nigeria, Switzerland, Italy, Germany, Japan, Austria, Netherlands, India, Pakistan, and Morocco as per Figure 3.6.

```
#Identify bank locations
#Combined unique location across both columns
combined_unique_location = pd.unique(df[['Sender_bank_location', 'Receiver_bank_location']].values.ravel())
print(combined_unique_location)

['UK' 'UAE' 'Spain' 'France' 'USA' 'Mexico' 'Albania' 'Turkey' 'Nigeria'
 'Switzerland' 'Italy' 'Germany' 'Japan' 'Austria' 'Netherlands' 'India'
 'Pakistan' 'Morocco']
```

Figure 3.6: List of countries of bank location

As per Figure 3.7, there are 13 types of different currencies exist in this dataset which are UK pounds, Dirham, Indian rupee, Pakistani rupee, Euro, US dollar, Mexican Peso, Albanian lek, Turkish lira, Naira, Swiss franc, Yen, and Moroccan dirham.


```
#Identify currency involved
#Combined unique currency across both columns
combined_unique_currency = pd.unique(df[['Payment_currency', 'Received_currency']].values.ravel())
print(combined_unique_currency)

['UK pounds' 'Dirham' 'Indian rupee' 'Pakistani rupee' 'Euro' 'US dollar'
'Mexican Peso' 'Albanian lek' 'Turkish lira' 'Naira' 'Swiss franc' 'Yen'
'Moroccan dirham']
```

Figure 3.7: List of currencies

iv. .nunique()

This function returns the number of unique values for an attribute. As per Figure 3.8, there are 292715 unique sender accounts and 652266 unique receiver accounts in the dataset.

```
#Identify numbers of unique sender account
unique_SenderAccounts = df['Sender_account'].nunique()
print(f"Number of unique sender accounts: {unique_SenderAccounts}")

#Identify numbers of unique receiver account
unique_ReceiverAccounts = df['Receiver_account'].nunique()
print(f"Number of unique receiver accounts: {unique_ReceiverAccounts}")

Number of unique sender accounts: 292715
Number of unique receiver accounts: 652266
```

Figure 3.8: Number of unique sender and receiver accounts

3.4.2 Data Type Transformation

Data type transformation is the process of converting columns into different data format. For example, convert integer data types into string format. This process is important to improve data usability for analysis and support decision making. As per Figure 3.9, the data type for 'Sender_account' and 'Receiver_account' have been changed from integer to string format since these columns are not applicable for arithmetic operations.

```
#Convert Sender_account and Receiver_account to string format
df['Sender_account'] = df['Sender_account'].astype(str)
df['Receiver_account'] = df['Receiver_account'].astype(str)
```

Figure 3.9: Changing data type for sender and receiver account

In Figure 3.10, data type for column ‘Time’ is changed to datetime format. Then, ‘Hour’ is extracted from time to create a new column and stored as string format. The value stored in ‘Hour’ column is now categorical variable from ‘0’ until ‘23’.

```
#Convert Time to datetime format
df['Time'] = pd.to_datetime(df['Time'], format='%H:%M:%S')

#Extract Hour from Time column and create new column
df['Hour'] = df['Time'].dt.hour

#Convert Hour into string format
df['Hour'] = df['Hour'].astype(str)
print(df['Hour'].unique())

['10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21' '22' '23' '0'
 '1' '2' '3' '4' '5' '6' '7' '8' '9']
```

Figure 3.10: Changing data type for time and create new column named Hour

As per Figure 3.11 and Figure 3.12, ‘Date’ column has been changed into datetime format. Then, ‘Year-Month’ and ‘Day’ are extracted from the date to create two new columns, and both are stored as string format. The value for ‘Year-Month’ is from ‘2022-10’ until ‘2023-08’. Meanwhile, the value for ‘Day’ is from ‘1’ until ‘31’.

```
#Extract Year-Month from 'Date' column and create new column
df['Year-Month'] = pd.to_datetime(df['Date']).dt.to_period('M')

#Convert Year-Month into string format
df['Year-Month'] = df['Year-Month'].astype(str)
print(df['Year-Month'].unique())

['2022-10' '2022-11' '2022-12' '2023-01' '2023-02' '2023-03' '2023-04'
 '2023-05' '2023-06' '2023-07' '2023-08']
```

Figure 3.11: Changing data type for Date and create column Year-Month

```
#Extract day from 'Date' column and create new column
df['Day'] = pd.to_datetime(df['Date']).dt.day

#Convert Day into string format
df['Day'] = df['Day'].astype(str)
print(df['Day'].unique())

['7' '8' '9' '10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21'
 '22' '23' '24' '25' '26' '27' '28' '29' '30' '31' '1' '2' '3' '4' '5' '6']
```

Figure 3.12: Changing data type for Date and create column Day

3.4.3 Data Cleaning

Data cleaning is the process to identify and fix inaccurate, missing, and irrelevant data in the dataset to improve data quality and achieve reliable result from the analysis. Figure 3.13 shows that this dataset has no duplicated data using the function `df.duplicated()`. Furthermore, this dataset is also free from missing values as per Figure 3.14 using the function `df.isnull()`.

```
#Check for duplicates
df.duplicated().sum()

0
```

Figure 3.13: Identify duplicated data

```
#Check for missing values
print(df.isnull().sum())

Time      0
Date      0
Sender_account      0
Receiver_account      0
Amount      0
Payment_currency      0
Received_currency      0
Sender_bank_location      0
Receiver_bank_location      0
Payment_type      0
Is_laundering      0
Laundering_type      0
dtype: int64
```

Figure 3.14: Identify missing data

However, some of the columns are no longer relevant for this project since new columns have been derived from the original column, thus there is a need to remove the columns 'Time' and 'Date' using function '.drop()' as per Figure 3.15.

```
#Drop the original Date and Time columns
df = df.drop(columns=['Date', 'Time'])
```

Figure 3.15: Remove irrelevant columns

Finally, using the function '.dtypes', recheck the data types and validate the current number of columns to ensure that it reflects all the changes that are done. As per Figure 3.16, it is confirmed that column 'Date' and 'Time' has been removed from the dataset and the data type for columns 'Sender_account', 'Receiver_account', 'Hour', 'Year-Month', and 'Day' has been transformed into categorical variables.

```
#Recheck the current column with it data types
print(df.dtypes)
```

Time	object
Date	object
Sender_account	int64
Receiver_account	int64
Amount	float64
Payment_currency	object
Received_currency	object
Sender_bank_location	object
Receiver_bank_location	object
Payment_type	object
Is_laundering	int64
Laundering_type	object
dtype:	object

3.16: Check data type for each column

3.5 Model Training

3.5.1 Support Vector Machines

The first supervised machine learning that this project will use is Support Vector Machine (SVM) to predict suspicious transactions for money laundering detection. SVM is one of the techniques that are widely used in classification, outlier detection and is very useful for nonlinear and complex model. SVM use hyperplane to split two classes in the sample. It is important to optimize the SVM model by identifying the best hyperplane so that the model can achieve better predictions on normal and suspicious transactions. The optimization of SVM model is done by tuning the SVM kernels and hyperparameters.

SVM kernels can be categorized into two types either linear or nonlinear. Linear type has only 1 category, while nonlinear type has three categories which are 'radial basis functions' (RBF), 'polynomial' (poly), and 'sigmoid' functions. As for important hyperparameters for tuning SVM, it involves 'gamma' and 'C'. 'Gamma' indicates the curvature shape that we want in decision boundary and only needed if using RBF kernel. Meanwhile, 'C' is a parameter to control error. The tuning is performed by using k-fold cross validation.

3.5.2 Decision Tree

The second supervised machine learning that this project will use is Decision Tree. It is widely used in classification because of its simplicity, interpretability, and ability to handle both categorical and numerical data. A Decision Tree splits the dataset into subsets based on the most significant features to create a tree-like structure of decision rules that guide the classification process. Decision Tree use branch to separate normal and laundering transactions. It creates a well-defined branch by identifying the best split at each node which is measured using criteria such as Gini Impurity or Entropy.

The optimization of Decision Tree model is done by tuning the hyperparameters which are max_depth, min_samples_split, and min_samples_leaf. Max_depth is the maximum depth of the tree, min_samples_split is the minimum number of samples to split the node, and min_samples_leaf is the minimum number of samples required at a leaf node. The tuning is performed using Grid Search or Random Search with k-fold cross-validation.

3.6 Model Evaluation

Model evaluation is the stage where the prediction results from the test dataset is evaluated using testing criteria. For this project, the evaluation metrics selected to assess the performance of models are confusion matrix, Precision, Recall, and Area Under Precision-Recall Curve (AUPRC).

Confusion matrix includes True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR). Table 3.2 below shows the confusion matrix model for financial transactions.

Table 3.2: Confusion Matrix Table

		Predicted Class	
		Positive (Normal)	Negative (Suspicious)
True Class	Positive (Normal)	True Positive (TP)	False Negative (FN)
	Negative (Suspicious)	False Positive (FP)	True Negative (TN)

The True Positive Rate measures the proportion of suspicious transactions that are correctly labelled. It is important to achieve high TPR so that the banks can immediately takes further action to report the flagged account without further ado as it is most likely to be true.

$$TPR = \frac{TP}{TP + FN} \quad (3.1)$$

The True Negative Rate measures the proportion of normal transactions that are correctly identified. It is better to get high TNR so that normal transactions are not wrongly labelled.

$$TNR = \frac{TN}{TN + FP} \quad (3.2)$$

The False Positive Rate measures the proportion of normal transactions that are inaccurately labelled as suspicious. Lower FPR is better as high FPR leads to wasted resources and high operational cost to double confirm the status of transactions.

$$FPR = \frac{FP}{FP + TN} \quad (3.3)$$

The False Negative Rate measures the proportion of suspicious transactions that are incorrectly labelled as normal. It is critical to have low FNR as high FNR means that many suspicious transactions are undetected which threaten the financial integrity and economic stability.

$$FNR = \frac{FN}{FN + TP} \quad (3.4)$$

In addition, Precision, Recall, and Area Under Precision-Recall Curve (AUPRC) are also used to evaluate the model performance. While precision indicates how precise a model is, recall indicates how robust a model is. It is important to note that, precision and recall does not have a linear relationship, thus it is not guaranteed that a high precision model will also be high recall model. Therefore, to solve this problem, F1 score can be used as it measures the harmonic average of precision and recall. These metrics are measured as per following equations.

$$Precision (normal) = \frac{TP}{TP + FP} \quad (3.5)$$

$$Precision (fraud) = \frac{TN}{TN + FN} \quad (3.6)$$

$$Recall (normal) = TPR = \frac{TP}{TP + FN} \quad (3.7)$$

$$Recall (suspicious) = TNR = \frac{TN}{TN + FP} \quad (3.8)$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.9)$$

3.7 Model Findings and Presentation

This final phase emphasizes on producing a comprehensive report and clear visualizations to present the model's performance, key findings such as the most important features that highly correlates to suspicious transactions and specific time periods that have high occurrence of suspicious transactions. In addition, limitations and recommendations are also discuss for improvement in the future project. These insights and findings will be compiled into a report and presentation will be delivered with the aid of visualization tools such as Microsoft PowerBI to capture the interest and build engagement with the audience.