COMPARATIVE STUDY BETWEEN STATISTICAL FRAUD DETECTION METHODS ON ECOMMERCE NETWORKS

by

Larkin Liu

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial
Engineering University of Toronto

Abstract

Comparative Study between Statistical Fraud Detection Methods on eCommerce Networks

Larkin Liu

Master of Applied Science

Graduate Department of Mechanical and Industrial Engineering

University of Toronto

2017

In the eCommerce industry, the problem of consumer fraud is becoming an increasingly troublesome problem posing a multitude of challenges. Large eCommerce firms, such as Alibaba, eBay, Amazon (to name a few) serve the needs of hundreds of millions customers. On a daily basis, a large number of transactions are made and recorded, as more consumers are becoming susceptible to fraudulent behaviour provided the scale of operations. This research aims to develop a robust real-time fraud detection algorithm based on the construction of a behavioural reference model. Time series features pertaining to a data set of customers on a large eCommerce service are collected and assigned to two classes (non-fraudulent or fraudulent) and modelled using a general stochastic model. The classification performance of this model is compared with more conventional classification techniques, such as logistic regression, tree-based methods, and Naïve Bayesian Classifier.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

In the following work, we analyse the problem of fraud prevention from an internet based eCommerce platform. We propose to approach the problem considering both a transaction level and customer level and develop a robust behavioural model that characterizes the legitimacy of transactions. For example, each customer may have a preference of purchasing certain types of merchandise, and behavioural models can generalize the global behaviour of all customers on the platform. Basic methods of fraud detection include comparison of observed data with expected values, however, this method is dependent on how the expected data is generated, or how the data is modelled (Bolton, 2002).

In the context of fraud, large firms increasingly recognize that consumer fraud is an organized crime, and fraudsters change their patterns to avoid detection (Cavusoglu, 2004). Such fraudsters will attempt to mask their devious behaviour by masquerading as genuine customers, finding exploits, and executing fraudulent transactions in batch on multiple accounts. This presents a challenge for investigators to detect fraud with certainty. Such fraudsters are also extremely high risk due to the amount of cost incurred by parties affected, the consumer, and the merchant. Fraudulent customers can nest in a large complex system very well where classification of fraudulent customers can be hard to perform due to the complex relationships and scale of an eCommerce network.

Classification of customer behaviour is of value both from a marketing standpoint and a fraud detection prevention standpoint. Risk management domain experts must be able to understand a multidisciplinary collection of factors and processes that impact the eventual loss numbers, rather than look for malicious actions everywhere (Samet, 2013). High risk fraudsters should take precedence over low risk fraudsters and thus the process thus should be cost optimized.

(Moskovitch, 2009) presents a two tier classification system for fraud on eBanking networks. One based on login verification, and one based on continuous verification. Login based verification is one time, and examines the credentials of the customer before they are allowed to execute actions on the financial platform. Continuous verification focuses on the continual real time evaluation of a customer, constantly monitoring and surveying customer actions over time. In our research, we present modified approaches to both methods, with the primary focus on the latter method. We present our implementation in Chapter 5.

Industry oriented approaches will apply methods that yield immediate and tangible results, and thus model complexity should also be a key focus of our experiment. Many approaches to the classification of customers and the prescription of optimal policies have been published recently due to the rapid advancement of the eCommerce industry. However, it will be a major challenge to build a scalable and practical solution that should allow even non-scientific parties to appreciate added value of robust fraud detection program.

1.1 Industry Overview

The objective of this research is to provide a means to robustly detect fraud on large scale eCommerce networks, a severe problem costing the eCommerce industry major financial loss if it is not cost-effectively resolved. Traditionally, heuristic rules are applied to classify whether each transaction or customer is fraudulent. Evidently, on large scale eCommerce platforms with over millions of customers transacting regularly, manual heuristic evaluation is not an ideal solution as it requires the human input from an ever-growing team of customer analysts. Furthermore, the thresholds and rules designated which detect fraud may be subject to change over time. As an eCommerce service expands, it is essential to construct an automated system that will be able to replicate human judgement on a large scale, and potentially supersede the predictive capability of human judgement.

The customer transaction data provided in this thesis is supplied by our client, a multimillion dollar valuated company based in India, serving over 120 million customers as of June 2016, which we

will refer to as Company A. Company A's regular operations primarily consists of online bill payment services, business-to-consumer (B2C) marketplace, mobile banking, and virtual currency. They constitute a direct-to-home bill payment service system, allowing its customers to perform bill payments for landline, cell phone, internet service, electricity and hydro bills. Furthermore, Company A facilitates an online marketplace where customers can purchase physical goods or services using credit card and/or virtual cash, and have such goods shipped to their physical address.

The reduction of fraud on our client's existing infrastructure is crucial to the long-term cost-reduction effort. The need for a robust, efficient, and accurate form of automated fraud detection is higher than ever considering Company A's recent business objectives to move into the realm of mobile banking. As the platform scales to meet the demands of almost a billion people, the problem of fraud is becoming an ever-increasing factor in determining the success of Company A. The fact that the eCommerce industry in general is experiencing a major surge in scale and volume only serves to exacerbate the problem of fraudulent behaviour due to rapid expansion.

Company A is not alone in the emerging eCommerce market in India. Key players have already existed prior to Company A's emergence in the eCommerce industry. Direct competitors that offer an online marketplace to hundreds of millions of Indian consumers include *Flipkart*, *Snapdeal*, and *Amazon India*, all of which are billion dollar valuated companies. Provided this, Company A's recent emergence can be viewed as an eager attempt to break into the already existing eCommerce industry. However, Company A offers the unique advantage over its competitors by providing a direct integration between an online marketplace, a direct payment system, and online banking platform. This integration increases the volume of transactions on the platform, requiring a large and sophisticated fraud detection team to assist with.

1.2 Objectives & Motivation

In this work, our aim is to develop significant improvements to the fraud detection mechanism of Company A, via the application of statistical learning methods presented in the literature, as well as devising a new robust real time system that applies stochastic modelling and analysis. This new

fraud detection system will account for both categorical variables and numerical variables, and will run in simulated real time to capture fraudulent behaviour with acceptable confidence levels. We segment our research effort into 3 phases.

1. The first phase will be to apply standard data analysis to extract useful insight from the observed features, aimed specifically at detecting fraud.
2. The second phase will apply conventional statistical learning models found in literature to our data, subsequently examining each model's effectiveness. These models include logistic regression, Naïve Bayesian classifier, and tree-based methods.
3. The third phase involves constructing a stochastic framework that will capture the behaviour of customers on the eCommerce network, and will be able to eliminate instances of fraud in real-time, allowing us to capture fraud in short succinct time intervals with high probability.

1.3 Research Contribution

This thesis concentrates on the application of established methodologies and development of new methodologies to classify fraudulent behaviour on a large-scale eCommerce network. It aims to provide a solid theoretical foundation to understand each of the models applied, and the benefits and disadvantages of each model. In the second part of the thesis, we aim to develop a novel stochastic model based on the Hidden Markov Model. We compare the effectiveness of the model, and propose a framework to effectively classify and combat fraud. In summation, this thesis addresses two main concepts, and is presented in two major sections.

1. A comparative study between commonly applied classification methods for fraud detection for eCommerce platforms based on the collected data. The models investigated are Logistic Regression, Naïve Bayesian Classifier, Decision Tree and Random Forest.

2. The development of an algorithm, based on Hidden Markov Modelling, to better describe customer behaviour over time, and classify fraudulent behaviour on the eCommerce network.

From an application standpoint, this is the first time that HMM modelling has been applied to detect fraud on eCommerce networks. We provide a substantial amount of new research considering stochastic modelling for this purpose. First, we develop a comparative study between our algorithm, and conventional methods of classification, as investigated in Chapter 3. Secondly, we construct observations from a multidimensional set of features related to customer spending, as opposed to the previous implementations of HMM for fraud detection, which considers only a single source of numerical spending data (Srivastava 2008). This allows us to base our model on higher dimensionality, increasing the robustness the model. Furthermore it constitutes a combination of transactional data from multiple sources, where each data source contains a different kind of information. With respect to the clustering mechanism that assigns time windowed features to a cluster, there is a slight modification to the K-Means algorithm to include an observation symbol to indicate that no spending has occurred within that time period.

1.4 Thesis Organization

The thesis is organized into 6 distinct Chapters. Chapter 1 presents a high level overview of the industry and the motivation behind our research. Chapter 2 outlines the definitions of the observed variables, and illustrates how to build aggregate features over a customer's entire lifetime. Chapter 3 explores the many classical classification techniques used to detect fraud and evaluates each classifier's performance. Chapter 4 provides theoretical background for Hidden Markov Models. Chapter 5 illustrates the process of building discrete time observation symbols used for Hidden Markov Modelling, and provides an online fraud detection system based on Hidden Markov Modelling. Chapter 6 summarizes our findings and present further ideas to improve our modelling process.

Chapter 2. Data Analysis and Feature Extraction

2.1 Platform Design and Architecture

Though Company A's platform consists of a single eCommerce service, it is separated into two distinct entities, the *marketplace*, the *virtual wallet*, and the *registration portal*. The marketplace is an exchange where vendors, also known as merchants, list goods and services for sale and their respective prices. Vendors are registered with the system, and the marketplace provide consumers with an online store where consumers can browse the merchandise, and have the merchandise shipped to their desired location from the merchant.

The *registration portal* consists of relevant data that the user inputs one-time upon registering on the platform. The data is categorical in nature, and refers to registration parameters such as email verification, phone verification, and the type of account. All of the data is categorical in nature and non-dynamic in time.

The *virtual wallet* is a place where customers can convert real currency into virtual currency. This virtual currency is usable on Company A's marketplace to purchase goods and services. One advantage of using virtual currency is that it provides a simplified means to obtain rewards and rebates as promotional offers. Furthermore, Company A's virtual currency allows customers to exchange currency amongst one another. With the emergence of mobile banking in developing countries, virtual currency can now be used as a substitute for real currency, where users are carrying around their mobile devices to pay for physical goods and services in lieu of cash[1].

[1] The use of virtual currency on mobile device instead of real currency is an area of rapid development in developing countries, such as India and China.

Marketplace data refers to records of everyday transactions taking place on the eCommerce platform. Virtual wallet data refers to the transactions that alter the amount of virtual currency stored on each customer's virtual bank accounts. And registration data refers to parameters that the customer used to register an account. Data from all three sources are combined to form a summary table. Specifically, marketplace and virtual wallet data are aggregated into discreet time windows, which is further outlined in Chapter 5. Alternatively, numerical features can summarized across the entire time history each customer, as outlined in Chapter 2. Subsequently, transactional data is merged with one-time only registration data to form a complete data set for our experiment.

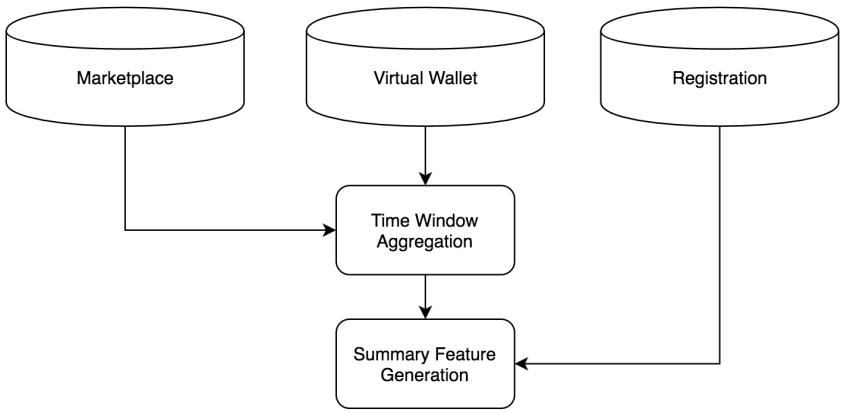

Figure 2.1.1 Raw data obtained to generate summary features originate from multiple sources on the database.

The marketplace and the virtual wallet keep track of customer spending and purchasing behaviour in two separate databases. These two databases contain transactional data and the registration database contains one-time registration data, which is in a categorical format. All the data can be merged together using the *customer_id* key, which traces each record on the database to its respective unique customer.

2.2 Description of Observed Features

The transaction period spans over the entire time frame of 6-months, from March 1st 2016 to July 9th 2016, consisting of a total of 7575 customers, and 336550 total transactions. All customers, from our perspective are provided a label, indicating whether they are fraudulent (1) or healthy (0). We define,

- **Training Set** – A majority random sample of the customers in our data set, where the observations are constituted by each of the customer's transactions. The data in this set are used to construct the model and estimate the classification model parameters. The training data set is a 75% random sample without replacement relative to the full data set of customers.
- **Validation Set** – A random sample of the data set that is selected for testing and validation purposes. This sample is completely separate from the training set as it simply contains all of the customers and their respective transactions that do not belong to the training set. The validation set is used for testing purposes to evaluate the classification accuracy of the model. The validation set is a 25% random sample without replacement relative to the full data set of customers.

Notation	Full Data Set	Training Set	Validation Set
N_i (Number of Total Customers)	7575	5681	1894
$N_{F=1}$ (Number ofTotal Fraudulent Customers)	348	262	86

Figure 2.2.1 Training and validation data set.

2.3 Feature Definitions & Effectiveness

In this section, we explore the feature definitions of each customer, and present how features are defined. Categorical features are transformed into binary dummy variables (0, 1), and numerical features can either be aggregated over discrete time periods. Time window aggregation will be presented in Chapter 5. In this section we present aggregation over the entire customer lifetime. We utilize the following features for fraud classification, when aggregating over the customer lifetime (Udo & Taudes, 1987).

Feature Name	Coding	Definition
sum_gmv	v_1	Total spending (GMV) of a customer.
count_txn	v_2	Total number of transactions of a customer.
mean_diff_purc	v_3	Average time in between transactions of a customer.
count_distinct_cat	v_4	Number of distinct purchasing categories exhibited by customer.
mean_gmv_per_txn	v_5	Average GMV per purchase per customer.
sum_gmv_wal	v_6	Total GMV in the customer's virtual wallet.
count_txn_wal	v_7	Total number of transactions in the customer's virtual wallet.

mean_diff_purc_wal	v_8	Average time in between transactions of a customer's virtual wallet.
mean_gmv_per_txn_wal	v_9	Average GMV per purchase per customer in virtual wallet.
phone_isverified	b_1	Indicates whether or not a customer's email address was verified using SMS[2] services.
email_isverified	b_2	Indicates whether the customer has verified their email via email reply confirmation.
wallet_type_SCW	b_3	Indicates whether or not the customer has activated a regular (SCW) type virtual wallet.
wallet_type_NO_WALLET	b_4	Indicates if the customer has no virtual wallet.
wallet_type_PRIME	b_5	Indicates if the customer has activated a premium (PRIME) wallet.

Figure 2.3.1 Feature definitions of categorical and numeric features.

2.3.1 Feature Aggregation

Several methods can be applied to model the behaviour of customers over time, some of which are outlined in (Udo & Taudes, 1987). In our particular application, we provide two methods of aggregation. The *full time period* aggregation consists of the summary statistic aggregation over each customer's entire purchasing history. Whereas the *discrete time* aggregation method aggregates each customer's purchasing history over sliding time windows.

Identification Key	Aggregation	Description	Example
Customers	Full Time Period	Accumulation of feature over the full 6-month time frame.	Total GMV
Customers	Discrete Time	Aggregation of a time window set of features over each customer's lifetime.	Total GMV per 12 hours.

Table 2.3.1 Feature aggregation definitions.

Fraud identification will function on a customer-centric basis, defined as the *identification key*, all transactions are attributed to individual customers and aggregated over fixed and independent time intervals. The experiment will seek to identify fraudulent customers by examining each customer's features. The following features are obtained from complete transaction history time window, denoted by Φ, aggregate observed time series data for each customer over the entire transaction history. This will be further detailed in Chapter 5.

[2] SMS or *Short Messaging Service* is a standard communication protocol for telephone and world wide web services.

2.4 Performance Metrics

For purposes of model evaluation, the model will classify each customer, provided their purchasing history, as 0 or 1. We refer to this as the predicted label, which is output from the proposed classification model. Each customer will also be assigned a true label of 0 or 1. The true label is provided by business analysts, where we assume that their knowledge about the customer constitutes the actual state of affairs. Model performance inform us of the comparative accuracy of the predicted label and the true label. We define the model metrics in Table 2.4.1.

Metric	Definition
True Positives (TP)	The number of occurrences where both the feature value and the actual label are true (1).
False Positives (FP)	The number of occurrences where the feature value is true (1) but the actual label is false (0).
True Negatives (TN)	The number of occurrences where the feature value is false (0) but the actual label is also false (0).
False Negatives (FN)	The number of occurrences where the feature value is false (0) but the actual label is true (1).
True Positive Rate (TPR)	The rate of detection, formula described by Equation (2.4.1)
False Positive Rate (FPR)	The rate of false detection, formula described by Equation (2.4.2).

Table 2.4.1 Metric definition table.

2.4.1 Detection Rate (True Positive Rate - TPR)

The detection rate, also known as true positive rate (TPR), is defined as the proportion of the detected true positives (TP) to all the positives in the set, which includes the number of true positives (TP) in addition to the number of false negatives (FN). Having a high proportion ensures that most of the positives in the set are detected. In our application, a high detection rate, or TPR, ensures that the majority of the cases of fraud are detected, thus this is a metric which should be maximized.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \qquad (2.4.1)$$

2.4.2 False Positive Rate

The false positive rate (FPR) of the classifier determines the proportion of inaccurate predictions, known as false positives (FP), relative to all of the negatives of the set, which is the addition of the number of true negatives (TN) and false positives (FP). Therefore, a good classifier minimizes the FPR. For fraud detection, the FPR denotes the proportion of people who were detected to be fraudulent, but were in fact non-fraudulent.

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP} \qquad (2.4.2)$$

2.4.3 Effectiveness of Classification

First, we evaluate each classifier's effectiveness by the following metrics. Provided the output of a classifier we can produce the following metrics to measure the classifier's effectiveness. For classification methods, such as logistic regression, it is possible to tune the algorithm to maximize TPR, while reducing FPR, or vice-versa. For example this can be accomplished by adjusting the threshold levels of a logistic regression model. In binary classification, the effectiveness of the classification algorithm depends on both the maximization of detection rate (TPR) and the minimization of the false positive rate (FPR). We construct a receiver operating characteristic (ROC) curve to measure the performance of each classification model.

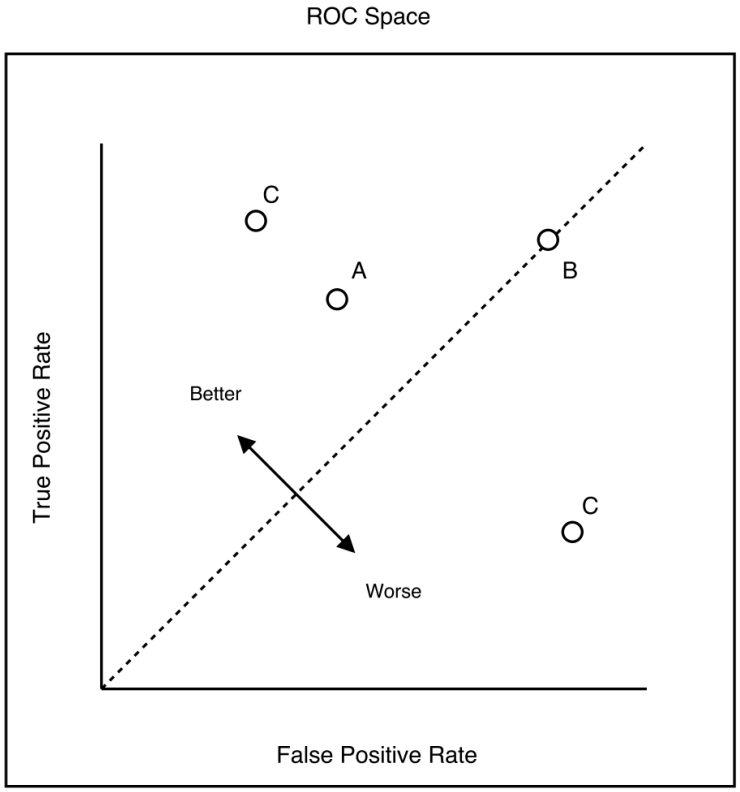

Figure 2.4.1 Illustration of the ROC Plane.

As illustrated in Figure 2.4.1 a point that lies in the perfect diagonal along the ROC plane is a bad predictor as it is as good as randomly guessing which class a variable belongs to. Point A is a good classifier because it has a higher TPR than FPR. For point C located below the random guess line, we see that a bad classifier with FPR consistently lower than the TPR can be inverted to produce a good classifier. In binary classification, a classifier, denoted by *f(x)* assigns a probability as to what class (0, 1) it believes the data point to belong to. To accomplish this, a threshold T is set, and each data point is evaluated based on its value relative to that data point.

$$f(x_1) < T \Rightarrow x_1 \epsilon\, 0 \qquad (2.4.3)$$
$$f(x_1) \geq T \Rightarrow x_1 \epsilon\, 1$$

According to the classification rule set out in Equation (2.4.3) if T = 0.3 for example, then if $f(x_1) \geq 0.3$ it would belong to class 0, else it would belong to class 1. In our case the threshold

parameter can take on multiple values. By varying the threshold value, *T*, we adjust the true positive rate (TPR) which is generally inversely proportional to the false positive rate (FPR). For purposes of fraud detection, more value is placed on obtaining higher TPR values while allowing for a higher FPR. This is because organizations typically place more importance on finding all instances of fraud, over reducing the number of false positives, as the cost of investigation is usually less than the cost suffered due to undiscovered fraud.

Ultimately we evaluate the effectiveness of a model based on the shape of the ROC curve. As we adjust the classification threshold *T*, there is a TPR and FPR value associated with it, constituting a point on the ROC plane. A collection of many points forms a curve on the ROC plane, characterizing the behaviour of the classifier. In general, the greater the area under the ROC curve the better the performance of the classifier. However, for our application, since we specify an FPR threshold, to which we maximize the detection rate of the algorithm, we do not need to examine every point on the ROC curve, only the point at which the detection rate is maximized without exceeding the FPR limit. However, we present a full ROC plot in Figure 3.5.1 of each of our classifiers to present a comparison on ROC between each classifier.

2.5 Individual Feature Effectiveness

To evaluate the effectives of binary variables, which are variables which can only take on a value of 0 or 1, we examine the classification metrics by counting the distinct features which match the label of the customer (0, 1). The feature effectiveness table is presented in Table 2.5.1,

Feature Name	Coding	TP	FP	TN	FN	TPR	FPR
phone_isverified	b_1	272	7199	28	76	0.7816	0.9961
email_isverified	b_2	70	5756	1471	278	0.2011	0.7964
wallet_type_SCW	b_3	50	2818	4409	298	0.1436	0.3899
wallet_type_NO_WALLET	b_4	285	3570	3657	63	0.8189	0.4939
wallet_type_PRIME	b_5	2	195	7032	346	0.0057	0.0269

Table 2.5.1 Feature effectiveness of binary registration features.

As evident in Table 2.5.1 we see that feature b_1 is prone to detecting false positives among fraud. Which suggests that the inverted feature[3] $\sim b_1$ value is better suited at predicting fraud. The same can be said for feature b_2 to a lesser degree. In fact, for all features, except b_4, it is more appropriate at detecting false positives among fraud. This is because b_1 and b_2 are verification parameters. b_3 and b_5 and indicators of whether or not a customer has registered for an online virtual wallet (of which there are two types). As more of the following features take on a value of 1, the likelihood of the customer committing fraud decreases. Whereas, if b_4 is true, meaning a customer has no virtual wallet, the likelihood of that customer committing fraud is higher.[4] For numerical features, which are features that fall in the range $[0, \infty)$, we set a binary threshold as the median between the largest and smallest value of the dataset.

$$T = median(S) = \frac{Max(S) - Min\ (S)}{2} \qquad (2.5.1)$$

And perform classification based on the threshold T, illustrated in Equation (2.4.3).

Feature Name	Coding	TP	FP	TN	FN	TPR	FPR
sum_gmv	v_1	276	3511	3716	72	0.7931	0.4858
count_txn	v_2	82	3685	3542	266	0.2356	0.5099
mean_diff_purc	v_3	62	3592	3393	261	0.1919	0.5142
count_distinct_cat	v_4	45	3373	3854	303	0.1293	0.4667
mean_gmv_per_txn	v_5	305	3482	3745	43	0.8764	0.4818
sum_gmv_wal	v_6	302	3485	3742	46	0.8678	0.4822
count_txn_wal	v_7	103	3683	3544	245	0.2959	0.5096
mean_diff_purc_wal	v_8	115	3559	3446	229	0.3343	0.5081
mean_gmv_per_txn_wal	v_9	306	3481	3746	42	0.8793	0.4817

Table 2.5.2 Feature effectiveness of ordinal features.

[3] The inversion of a feature implies positive values (1) becoming 0, and negative values (0) becoming (1).

[4] Note that the inversion of parameters to be more effective at detecting is not necessary when implementing a logistic regression.

Each feature itself can be used as a weak classifier, for certain features where the TPR is significantly greater than the FPR (or vice versa for inverted features). There is a demonstrable indication of effectiveness of such a feature.

Chapter 3. Classification

This chapter presents the traditional classification techniques that have been applied for online fraud detection. The main approaches are described in detail, and a specific case study is presented to illustrate the function and effectiveness of each classification technique on real data provided by our industry partner. We investigate the application of Logistic Regression, Naïve Bayesian Classifier, Decision Tree, and Random Forest to detect occurrences of fraud within aggregated customer data. Subsequently, we evaluate the effectiveness of each algorithm based on the *Receiver Operator Curve* (ROC), a common metric to evaluate the performance of classification algorithms. This indicates the performance of the True Positive Rate (TPR) to the False Positive Rate (FPR).

3.1 Logistic Regression

In this section, the theoretical background for logistic regression and parameter estimation using maximum likelihood estimation is presented. Subsequently, we present the relevant metrics specific to Logistic Regression evaluation. We measure the accuracy of the logistic regression as a regression tool by examining the deviance or AIC. We can also measure the performance of logistic regression as a classifier, by examining its ROC.

The application of Logistic Regression to fraud detection has been explored in a wide array of research. Logistic regression is widely applied because it provides easily interpretable results and straightforward implementation compared to other classification models. Research proposed in (Shen, et al., 2007) investigates the use of logistic regression for detecting credit card fraud. It finds that logistic regression is strongly suited for fraud detection applications. Furthermore, research proposed in (Marazanto, 2010) applied the use of logistic regression for fraud detection in eCommerce networks. (Marazanto, 2010) focused on detecting eCommerce vendors which committed fraud by deceiving the reputation systems of the eCommerce network. They discovered that logistic regression is a powerful tool to detect this specific type of fraud as it yielded a very low false positive rate.

Our approach involves fitting a logistic regression model on the available data, both categorical and numerical, and measuring the model performance. To compare, first, we fit a logistic regression on only the available categorical variables. Second, we fit a logistic regression model on only available numerical features. And finally, we fit a logistic regression model on an optimal combination of both numerical and categorical variables, as determined by the *backward elimination process*.

Notation	Definition
F_i	Indicator function (0, 1) that customer *i* is fraudulent, also known as the true label.
N_t	Maximum number of total iterations for the Newton-Raphson algorithm.
β_j	Logistic regression coefficient on the jth feature.
$\mathcal{L}$	Log-likelihood function for all observations.
P_i	Probability of fraud for customer *i*.
(n)	Indicator for the nth iteration of the Newton-Raphson method.
f(x)	Logistic function on *x*.
*P(**X**)*	Probability of fraud, provided set of variables $\boldsymbol{X}$
$\boldsymbol{X}$	Set of numeric and/or categorical variables, per customer.
$\boldsymbol{\beta}$	Coefficient matrix for logistic regression.
β_j	Coefficient for feature *j*.
x_{ij}	Variable *x* for customer *i* on feature *j*.
$\boldsymbol{X_i}$	Set of x_{ij} belonging to customer *i*.
N_c	Number of customers in the dataset.
γ	Tolerance limit for Newton-Raphson method.

Table 3.1.1 Logistic regression notation.

3.1.1 Introduction to Logistic Regression

Logistic regression provides a parametric form to express the probability of an event occurring as probability between 0 and 1. Logistic regression applies the use of the logistic function which was first introduced in the 19th century to describe population growth and autocatalytic chemical reactions (Cramer, 2002). We denote the logistic function as *f(x)*,

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x} \qquad (3.1.1)$$

The logistic function exhibits the specific property where the derivative of *f(x)* can be written as,

$$\frac{d}{dx}f(x) = f(x)(1 - f(x)) \qquad (3.1.2)$$

The fact that the derivative of the logistic regression can be written in this form will prove useful for parameter estimation of the model. Another important property of the logistic function is that it is bounded between 0 and 1, which is ideal for modelling the probability of events occurring, in our case the probability of fraud.

3.1.2 Theoretical Background

For fraud detection, we define the probability of fraud as $P(\boldsymbol{X})$, bounded between 0 and 1. This value indicates the probability of fraud for each customer's observed variables $\boldsymbol{X}$. We define the log odds as the logarithm of the odds of $P(\boldsymbol{X})$, where the odds are defined as the $P(\boldsymbol{X})/(1- P(\boldsymbol{X}))$. The odds can be understood as the ratio of fraud, expressed as $P(\boldsymbol{X})$, to the ratio of non-fraud, expressed as $1 - P(\boldsymbol{X})$. As an extension to linear regression, logistic regression involves expressing the log odds of $P(\boldsymbol{X})$ as a linear combination of the observed variables $\boldsymbol{X}$.

$$log\frac{P(\boldsymbol{X})}{1 - P(\boldsymbol{X})} = \boldsymbol{\beta X} \qquad (3.1.3)$$

Where for j features,

$$\boldsymbol{\beta X_i} = \beta_0 + \beta_1 x_{i1} \ldots + \beta_j x_{ij} \qquad (3.1.4)$$

For j features, and customer i, we can represent all features x as a single vector, and the universal coefficients related to each feature β_j (we note that the logistic regression coefficients apply to all customers),

$$\boldsymbol{\beta X_i} = [\beta_0 \quad \beta_1 \ldots \quad \beta_j] \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{ij} \end{bmatrix} \qquad (3.1.5)$$

Where each feature value can be both *categorical* or *numerical*. Categorical variables must be in the form of binary notation (0, 1). In our model β represents the unknown parameters of the model, and *x* represents the data. Numerical values can be expressed as any real number. Alternatively, solving for *P(**X**)*, logistic regression can be expressed as,

$$P(\boldsymbol{X}) = \frac{e^{\boldsymbol{\beta X}}}{1 + e^{\boldsymbol{\beta X}}} = \frac{1}{1 + e^{-\boldsymbol{\beta X}}} \qquad (3.1.6)$$

Logistic regression creates a linear classifier over the aggregate features to forecast the log odds that a customer is fraudulent on the network. We can use this approach to test the effectiveness of each feature by examining the p-values associated with each parameter value to examine the effectiveness of features.

3.1.3 Maximum Likelihood

The maximum likelihood estimation method provides a means to estimate the coefficients of a logistic regression model. We express the probability of fraud, *P*, for each customer *i*, as, P_i

$$P_i = \frac{1}{1 + e^{-\boldsymbol{\beta X_i}}} \qquad (3.1.7)$$

Let F_i denote the true fraud label of the customer *i*, as it can take on values (0, 1). *P* represents the general probability of fraud for any customer. Consider N_c number of customers in the population. The likelihood of the model can be represented as,

$$L(P) = \prod_{i=1}^{N} [P_i]^{F_i} [1 - P_i]^{1 - F_i} \qquad (3.1.8)$$

To simplify the maximization of the likelihood function, we can take the log of the likelihood function, otherwise referred to as log likelihood.

$$\mathcal{L} = \log[L(P)] = \sum_{i=1}^{N} [F_i \; log \; (P_i(X_i, \boldsymbol{\beta})) + (1 - F_i) \; log \; (1 - P_i(X_i, \boldsymbol{\beta}))] \tag{3.1.9}$$

We posit the likelihood function as a function of model parameters $\boldsymbol{\beta}$,

$$\mathcal{L}(\beta) = \sum_{i=1}^{N} [F_i \; log \; (P_i(X_i, \boldsymbol{\beta})) + (1 - F_i) \; log \; (1 - P_i(X_i, \boldsymbol{\beta}))] \tag{3.1.10}$$

$$\mathcal{L}(\beta) = \sum_{i=1}^{N} [\; log \; (1 - P_i(X_i, \boldsymbol{\beta}))] + \sum_{i=1}^{N} [\; F_i \; log \; (P_i(X_i, \boldsymbol{\beta}))] \tag{3.1.11}$$

$$\mathcal{L}(\beta) = \sum_{i=1}^{N} [\; log \; (1 - P_i(X_i, \boldsymbol{\beta}))] + \sum_{i=1}^{N} [\; F_i \; log \; \frac{P_i(X_i, \boldsymbol{\beta})}{1 - P_i(X_i, \boldsymbol{\beta})}] \tag{3.1.12}$$

$$\mathcal{L}(\beta) = \sum_{i=1}^{N} [\; log \; (1 - P_i(X_i, \boldsymbol{\beta}))] + \sum_{i=1}^{N} [\; F_i \;\; \boldsymbol{\beta X_i}] \tag{3.1.13}$$

$$\mathcal{L}(\beta) = \sum_{i=1}^{N} [\; -(\log 1 + e^{\boldsymbol{\beta X_i}})] + \sum_{i=1}^{N} [\; F_i \;\; \boldsymbol{\beta X_i}] \tag{3.1.14}$$

Taking the derivative of the log likelihood function we obtain,

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\sum_{i=1}^{N} \left(\frac{1}{1 + e^{\boldsymbol{\beta X_i}}} e^{\boldsymbol{\beta X_i}} \right) x_{ij} + \sum_{i=1}^{N} [\; F_i \;\; x_{ij}] \tag{3.1.15}$$

We see that P_i is a function of the weights $\boldsymbol{\beta X_i}$. We can say for each observation (or customer) i, we maximize the log-likelihood. And therefore for all N_c observations, we compute the overall log likelihood as,

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = \sum_{i=1}^{N} (F_i - P_i(X_i, \boldsymbol{\beta})) x_{ij} \tag{3.1.16}$$

Where the objective is to find the optimal parameters β that will maximize $\mathcal{L}$, by setting the partial derivative of the log-likelihood function to 0 for all parameters β_j, such that,

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = 0, for\ i = 0,1,..j \qquad (3.1.17)$$

Provided that the gradient of the likelihood is a concave function, we can employ a gradient ascent algorithm to discover the local maxima until the condition in Equation (3.1.17) is met. There are a variety of methods to computationally solve the problem, as there rarely exists a closed form solution. We present the Newton-Raphson Algorithm as a potential solution.

3.1.4 Newton Raphson Algorithm for Parameter Estimation

In most cases, the solution to the maximization of the log likelihood, $\mathcal{L}$, as illustrated in Equation (3.1.16) will not have a closed form solution. Therefore, an iterative numerical method must be applied to compute an approximation to the parameter values that satisfy the equation. We follow the Newton-Raphson iterative gradient descent algorithm, illustrated in (Shalizi, 2017) to sufficiently approximate the solution to Equation (3.1.16). First, we denote $f(\beta_j)$ as,

$$f(\beta_j) = \frac{\partial \mathcal{L}}{\partial \beta_j} \qquad (3.1.18)$$

Suppose $f(\beta_j)$, is a smooth well defined function of only β_j. It is possible to apply the Newton-Raphson method to find the solution. We initialize $\beta^{(0)}$ equal to a random number (in our example we select 0). Subsequently, we iteratively compute $\beta^{(n)}$, and for each subsequent iteration n + 1, we define the parameter update algorithm for the univariate case as,

$$\beta^{(n+1)} = \beta^{(n)} - \frac{f'(\beta^{(n)})}{f''(\beta^{(n)})} \qquad (3.1.19)$$

By extension, we can apply the Newton-Raphson method to the multivariate case, where f is a multivariate function consisting of several dimensions $f(\boldsymbol{\beta})$, where $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_j)$. In this scenario, we see that, our smooth well defined function illustrated in Equation (3.1.18) becomes the gradient of the likelihood function, as defined as,

$$f(\beta_j) = \nabla \mathcal{L}(\boldsymbol{\beta}^{(n)}) \tag{3.1.20}$$

Where the standard vector calculus gradient is defined as,

$$\nabla \mathcal{L}\left(\boldsymbol{\beta}^{(n)}\right) = \left(\frac{\partial \mathcal{L}}{\boldsymbol{\beta}_0^{(n)}}, \frac{\partial \mathcal{L}}{\boldsymbol{\beta}_1^{(n)}}, \dots, \frac{\partial \mathcal{L}}{\boldsymbol{\beta}_j^{(n)}}\right) \tag{3.1.21}$$

In this case, Equation (3.1.19) becomes,

$$\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} - H^{-1}[\mathcal{L}\left(\boldsymbol{\beta}^{(n)}\right)]\nabla \mathcal{L}(\boldsymbol{\beta}^{(n)}) \tag{3.1.22}$$

Where $\boldsymbol{\beta}$ is a vector, representing the coefficients for j features, and $H[\mathcal{L}\left(\beta^{(n)}\right)]$ is the Hessian matrix of the likelihood function.

$$\boldsymbol{\beta} = \left[\beta_0, \beta_1, \beta_2, \dots, \beta_j\right] \tag{3.1.23}$$

In the multivariate case the gradient $\nabla \mathcal{L}\left(\boldsymbol{\beta}^{(n)}\right)$ is defined as,

$$\nabla \mathcal{L}\left(\boldsymbol{\beta}^{(n)}\right) = \left[\frac{\partial \mathcal{L}}{\partial \beta_1^{(n)}}, \frac{\partial \mathcal{L}}{\partial \beta_2^{(n)}}, \frac{\partial \mathcal{L}}{\partial \beta_3^{(n)}}, \dots, \frac{\partial \mathcal{L}}{\partial \beta_n^{(n)}}\right] \tag{3.1.24}$$

$H[\mathcal{L}\left(\beta^{(n)}\right)]$ is the Hessian matrix of the likelihood function, representing a matrix of the second derivative with respect to different combinations of β's.

$$H[\mathcal{L}(\boldsymbol{\beta}^{(n)})] = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \beta_1^2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \beta_1 \partial \beta_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{L}}{\partial \beta_j \partial \beta_1} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \beta_j^2} \end{bmatrix} \quad (3.1.25)$$

As follows, the inverse of the Hessian matrix $H^{-1}[\mathcal{L}(\boldsymbol{\beta}^{(n)})]$ is the inverse of the matrix illustrated in Equation (3.1.25). The computation of the inverse Hessian matrix can be computationally intensive. Therefore, a series of computational techniques have been developed to approximate it. In this example, we will not apply such techniques because the Newton-Raphson method is illustrated on a small scale for example purposes only, and calculating $H^{-1}[\mathcal{L}(\boldsymbol{\beta}^{(n)})]$ without approximation methods will be sufficient.

For iterations 1 to n, we initialize the coefficients $\boldsymbol{\beta}^{(0)} = [0, 0, 0]$ and begin to solve for β. Subsequently, we iteratively compute the solution, iterating through β's, for n = 1 until the iteration limit. The iteration limit is imposed so that the program does not run for an infinite amount of time. $\boldsymbol{\beta}^{(n)}$ represents the computed coefficients up until the nth iteration. $\boldsymbol{\beta}^{(n)}$ is iteratively computed until one of two stopping conditions is met. The first stopping condition stipulates that the difference between the coefficients is less than some tolerance value γ. We compute the difference using the Euclidean norm,

$$\left\| \boldsymbol{\beta}^{(n+1)} - \boldsymbol{\beta}^{(n)} \right\| < \gamma \quad (3.1.26)$$

Where the Euclidean norm is defined as,

$$\|\boldsymbol{\beta}\| = \sqrt{\beta_1^2 + \beta_2^2 + \cdots + \beta_j^2} \quad (3.1.27)$$

The second stopping condition stipulates that the total number of iterations must not exceed some maximum value, N_t, that is,

$$n < N_t \qquad (3.1.28)$$

The Newton-Raphson method continues to iteratively compute new values of $\boldsymbol{\beta}^{(n)}$ until either one of the stopping conditions stipulated in Equation (3.1.26) or Equation (3.1.28) is met. Ideally $\left\|\boldsymbol{\beta}^{(n+1)} - \boldsymbol{\beta}^{(n)}\right\| \to 0$, as $n \to \infty$, however, as long as n is sufficiently large, the final parameters estimated by the Newton-Raphson method will sufficiently estimate the parameters of the model.

3.1.5 Deviance for Logistic Regression

We define the proposed model as the model that may apply only to a subset of the parameters, whereas the saturated model is a model that contains as many parameters as observations. Such a model will produce a perfect fit, and therefore is a redundant model. We present the standard metric known as deviance, which is a measure of goodness-of-fit for logistic regression models. We calculate deviance by computing the difference between the log-likelihood, $\mathcal{L}$, of the proposed model, with model parameters $P\left(F\middle|\widehat{\theta_M}\right)$ with the log likelihood of the saturated model $P\left(F\middle|\widehat{\theta_s}\right)$. We calculate deviance as,

$$D(x) = -2\left[\mathcal{L}\left(P\left(F\middle|\widehat{\theta_M}\right)\right) - \mathcal{L}\left(P\left(F\middle|\widehat{\theta_s}\right)\right)\right] \qquad (3.1.29)$$

3.1.6 Akaike Information Criterion

The Akaike Information Criterion (AIC) is a measure of the goodness of fit taking into account a penalty for an increased number of parameters. AIC is defined as,

$$AIC = \mathcal{L}\left(P\left(F\middle|\widehat{\theta_M}\right)\right) - k_M \qquad (3.1.30)$$

Where $\mathcal{L}$ is the likelihood of the estimator based on the observed data x, and $\mathcal{L}\left(P\left(F\middle|\widehat{\theta_M}\right)\right)$ represents the log likelihood. k_M represents the number of parameters in the model. The AIC penalizes the model for having a high number of parameters, and rewards the model maximizing the log-likelihood of the model.

3.1.7 Applying Logistic Regression

We perform a logistic regression on the binary features specified in Section 3.1. We see that the majority of the variables have a significant effect on predicting whether or not a customer is fraudulent. Data is randomly split, model fitting is performed on that training data, and model performance is checked on the validation data. The logistic regression models are constructed in the R programming language, and output is presented accordingly.

```
Call:
glm(formula = isFraudPred ~ b1 + b2 + b3 + b4 + b5, data = summaryDataEncoded)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.76744  -0.02324  -0.00410  -0.00410   0.99976

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.730185   0.019483  37.479  < 2e-16 ***
b1          -0.635748   0.018779 -33.854  < 2e-16 ***
b2          -0.108452   0.005785 -18.746  < 2e-16 ***
b3           0.018112   0.008112   2.233   0.0256 *
b4           0.037252   0.008021   4.644 3.47e-06 ***
b5           0.014257   0.015151   0.941   0.3467
---
```

Model Performance	Count
TP (True Positives)	80
TN (True Negatives)	1752
FP (False Positives)	420
FN (False Negatives)	21
FPR (False Positive Rate)	0.1933702
TPR (True Positive Rate)	0.7920792

Figure 3.1.1 Logistic Regression model performance.

3.1.8 Logistic Regression on Numerical Features

We perform logistic regression on a set of numerical variables. Numerical variables in our data set fall within the range [0, ∞]. Though, practically the upper bound of this range is limited by a realistic number. Applying parameter estimation for logistic regression we obtain the following output.

```
Call:
glm(formula = isFraudPred ~ v1 + v2 + v3 + v4 + v5 + v6 + xv7 + xv8 + xv9,
family = binomial, data = summaryDataEncoded)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2586  -0.2376  -0.1559  -0.0858   4.8640

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.971e+00  1.878e-01 -10.495  < 2e-16 ***
v1           8.883e-09  1.114e-09   7.977 1.51e-15 ***
v2          -6.819e-03  2.826e-03  -2.413 0.015812 *
v3          -1.551e-06  2.766e-07  -5.607 2.06e-08 ***
v4          -5.135e-01  5.587e-02  -9.190  < 2e-16 ***
v5          -7.188e-08  2.717e-08  -2.645 0.008162 **
v6          -1.884e-05  4.095e-06  -4.600 4.23e-06 ***
v7           6.105e-03  1.667e-03   3.662 0.000251 ***
v8           1.763e-07  1.147e-07   1.538 0.124139
v9           1.383e-03  8.256e-05  16.746  < 2e-16 ***
---
```

3.1.9 Backwards Stepwise Selection

In order to reduce the effects of overfitting, we perform the process of backwards stepwise selection (Symonds 2010) to select the most explanatory variables, and reduce the usage of redundant variables. The backwards selection. For logistic regression in particular we use the deviance metric for variable elimination.

1. Build a saturated model, where all explanatory variables are included..
2. Fit a model with the remaining variables, evaluate the AIC of each variable, including the null model.
3. Eliminate the variable with the lowest AIC, still greater than the null model.
4. Go to step 2 and continue until all deviances are greater than the null AIC.

```
Step:  AIC=839.52
isFraudPred ~ b1 + b2 + b3 + b4 + b5 + v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9
```

B5 was removed.

```
Step:  AIC=838.46
isFraudPred ~ b1 + b2 + b3 + b4 + v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9

       Df Deviance     AIC
- v8   1   810.81  836.81
- v2   1   810.85  836.85
<none>     810.46  838.46
- v6   1   813.10  839.10
- v7   1   816.00  842.00
- v5   1   818.27  844.27
- v3   1   838.64  864.64
- b3   1   862.65  888.65
- b4   1   872.49  898.49
- v1   1   884.40  910.40
- b2   1   894.45  920.45
- v4   1   916.45  942.45
- v9   1  1117.15 1143.15
- b1   1  1157.24 1183.24
```

V8 was removed.

```
Step:  AIC=836.81
isFraud ~ b1 + b2 + b3 + b4 + v1 + v2 + v3 + v4 + v5 + v6 + v7 + v9

       Df Deviance     AIC
- v2   1   811.16  835.16
<none>     810.81  836.81
- v6   1   813.35  837.35
- v7   1   816.20  840.20
- v5   1   818.61  842.61
- v3   1   841.46  865.46
- b3   1   862.97  886.97
- b4   1   872.67  896.67
- v1   1   885.09  909.09
- b2   1   895.47  919.47
- v4   1   917.10  941.10
- v9   1  1117.61 1141.61
- b1   1  1159.87 1183.87

Step:  AIC=835.16
isFraud ~ b1 + b2 + b3 + b4 + v1 + v3 + v4 + v5 + v6 + v7 + v9
```

V2 was removed.

```
       Df Deviance     AIC
<none>     811.16  835.16
- v6   1   813.40  835.40
```

```
- v7   1   816.20  838.20
- v5   1   818.61  840.61
- v3   1   841.49  863.49
- b3   1   864.12  886.12
- b4   1   874.41  896.41
- v1   1   886.40  908.40
- b2   1   897.79  919.79
- v4   1   919.00  941.00
- v9   1  1126.75 1148.75
- b1   1  1165.40 1187.40
```

We find the optimal logistic regression by applying the backwards selection algorithm. And evaluate the effectiveness of each feature based on deviance. From our logistic regression output we see that all of the explanatory variables have a strong significance (as determined by their respective p-values) in detecting instances of fraud versus non-fraud customers.

```
Call:
glm(formula = isFraud ~ b1 + b2 + b3 + b4 + v1 + v3 + v4 + v5 +
    v6 + v7 + v9, family = binomial, data = trainSummaryDataEncoded)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6763  -0.0987  -0.0512  -0.0228   5.4212

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.349e+00  3.135e+00  -2.983 0.002858 **
b11         -6.071e+00  4.435e-01 -13.688  < 2e-16 ***
b21         -2.220e+00  2.990e-01  -7.422 1.15e-13 ***
b3           1.305e+01  3.103e+00   4.206 2.60e-05 ***
b4           1.349e+01  3.110e+00   4.336 1.45e-05 ***
v1           1.636e-08  2.201e-09   7.430 1.09e-13 ***
v3          -1.024e-06  2.792e-07  -3.668 0.000244 ***
v4          -6.759e-01  8.918e-02  -7.579 3.49e-14 ***
v5          -1.032e-07  4.867e-08  -2.120 0.033987 *
v6           1.982e-06  6.290e-06   0.315 0.752668
v7           4.968e-03  1.950e-03   2.548 0.010823 *
v9           1.593e-03  1.199e-04  13.282  < 2e-16 ***
```

3.1.10 Model Performance

We examine the performance of the logistic regression model and find that it performs moderately well at detecting fraud. Our experiments produce an effective true positive rate of approximately 0.85, and a minimal false positive rate of 0.049. It is possible to adjust the classification threshold

to obtain a higher TPR (which is the main objective) whilst sacrificing the FPR, thereby increasing it. However, this is only used for model comparison only, thus it is not necessary.

Model Performance	Count
TP (True Positives)	86
TN (True Negatives)	2066
FP (False Positives)	106
FN (False Negatives)	15
FPR (False Positive Rate)	0.04880295
TPR (True Positive Rate)	0.8514851

Figure 3.1.2 Logistic Regression model performance.

3.2 Naïve Bayesian Classifier

Naïve Bayesian Classifiers are a well-established form of probabilistic classifiers that provide a means for binary classification. Naïve Bayesian classifiers are relatively simple to implement and the non-parametric form eliminates the computation step required for parameter estimation, thereby reducing computational complexity. However, Naïve Bayesian classifiers rely on the conditional independence between observed variables, which can hinder the predictive accuracy of the model if the data contains cross-correlated variables. In this section, we provide the theoretical background for the development of Naïve Bayesian classifiers, and in addition, we provide an in depth example concerning the application of Naïve Bayesian classifier on real data, in order to evaluate the performance of the model.

Notation	Definition
F	Classification of fraud, 1 for fraud, 0 for healthy.
B	Categorical variables.
B_i	The set features belonging to customer *i*.
F_i	Indicator function (0, 1) that customer *i* is fraudulent.
N_B	Number of categorical features.
N_i	Total number of customers.
$N_{F=1}$	Number of fraudulent customers.
b_i	Categorical feature, that takes on indicator value 1 if b_i is true and 0 vice-versa.
$N_{F=0}$	Total number of non-fraudulent customers
$N_{b_i=1}^{F=1}$	Total number of fraudulent customers, where the feature value b_i was true.
$N_{b_i=0}^{F=1}$	Total number of fraudulent customers, where the feature value b_i was false.
$N_{b_i=0}^{F=0}$	Total number of non-fraudulent customers, where the feature value b_i was false.

$N_{b_i=1}^{F=0}$	Total number of non-fraudulent customers, where the feature value b_i was true.
$\hat{F}$	Predicted fraud label of customer via Naïve Bayesian Classifier.

Table 3.2.1 Notation for Naïve Bayesian Classifier.

3.2.1 Applications of Naïve Bayesian Classifiers

In the specific context of fraud detection, Naïve Bayes Classifiers were applied in (Viaene, et al., 2004) to detect insurance claim fraud. (Sahami, et al., 1998) introduces an effective application of Naïve Bayesian Classifiers for email spam detection. We follow closely the examples presented in (Eberhardt, 2015) which applies the Naïve Bayesian Classifier to detect fraudulent and unwanted emails. We repurpose the Naïve Bayesian Classifier for fraud detection on our eCommerce platform. Using only the categorical customer registration features $\boldsymbol{B_i}$, defined in Chapter 2.

In our application of the Naïve Bayesian Classifier, we classify each set of features for each customer as belonging to class of fraud (F = 1) or non-fraud (F = 0), limiting our classification scope to two classes.

i	F	b_1	b_2	b_3	b_4	b_5
1	0	1	1	0	1	0
2	1	1	0	0	1	0
3	0	1	0	0	1	0
4	0	1	1	0	1	0
5	0	1	1	1	0	0
6	0	1	1	0	1	0
7	0	1	1	0	1	0
8	0	1	0	0	1	0
9	0	1	1	1	0	0
10	0	1	1	0	1	0
11	0	1	1	1	0	0
12	0	1	1	1	0	0
13	1	1	0	0	1	0
14	0	1	1	0	1	0

Table 3.2.2 A sample of real categorical data used for Naive Bayesian Fraud Classification.

3.2.2 Introduction to Naïve Bayes Classifiers

The Naïve Bayesian Classifier (NB) is a classification method based on Bayes theorem. Naïve Bayesian Classifiers assume that the effect of a feature value on a labelled class is independent of the values of the other observed features. This assumption is known as conditional independence and is often regarded as a disadvantage of the NB classifier because, in reality, observed features are often correlated with one another. To illustrate, we take the probability of events, A, B, and C occurring expressed as,

$$P(A, B|C) = P(A|B)P(B|C) \qquad (3.2.1)$$

We see that the probability of event A occurring independent of the whether or not B has occurring and vice-versa, provided the known state C. This is holds true for any number of events, and is the basic assumption of the Naïve Bayesian Classifier. We provide a general extension to categorical features, $b_{i1}, \dots, b_{i1}$, the probability of observing features $b_{i1}, \dots, b_{i1}$, provided the customer, i, is fraudulent, can be expressed as,

$$P(b_{i1}, \dots, b_{ij}|F) = \prod_{j=1}^{N_B} P(b_{ij}|F) \qquad (3.2.2)$$

Where two classes are present $F = 1$, indicates that the customer is fraudulent, and $F = 0$ indicate that the customer is healthy. Hence, we define the model parameters of a Naïve Bayesian Classifier as $P(\boldsymbol{B_i}|F)$ for all $\boldsymbol{B}$ categorical features. We apply this method across the training data, to calculate the model parameters. By extension, for observations $\boldsymbol{B_i} = b_{i1}, \dots, b_{ij}$, we say that, for j categorical features, the probability of observing the set of features $\boldsymbol{B_i}$, for customer i, is defined as,

$$P(\boldsymbol{B_i}|F) = \prod_{j=1}^{N_B} P(b_{ij}|F) \qquad (3.2.3)$$

Therefore, the probability of observing the features $\boldsymbol{B_i}$ is the summation of the conditional probabilities $P(\boldsymbol{B_i}|F)$ multiplied by the prior $P(F)$, for both fraud classes $F \in (0, 1)$.

$$P(\boldsymbol{B_i}) = \sum_{F} P(\boldsymbol{B_i}|F)P(F) \qquad (3.2.4)$$

The probability of a customer being fraudulent, after observing features $b_{i1}, \dots, b_{ij}$ belonging to customer *i*, can be expressed as,

$$P(F|\boldsymbol{B_i}) = \frac{P(\boldsymbol{B_i}|F)P(F)}{P(\boldsymbol{B_i})} \qquad (3.2.5)$$

We see that $P(\boldsymbol{B_i})$ acts as a normalizing factor as $P(F|\boldsymbol{B_i}) \propto P(\boldsymbol{B_i}|F)P(F)$. Note that the normalizing factor is constant, thus it does not need to be recomputed. We predict whether or not a customer is fraudulent given its features by finding the classification that maximizes the probability of observing the sequence $\boldsymbol{B_i} = b_{i1}, \dots, b_{ij}$, assuming conditional independence across all features. We produce a prediction, $\hat{F}$, of whether a customer is fraudulent or not ($\hat{F} = 0$ or $\hat{F} = 1$), based on finding the class label that provides the maximum likelihood over all possible states.

$$\hat{F} = \underset{F \in \{0,1\}}{\arg\max}\, P(F) \prod_{j=1}^{n_B} P(b_{ij}|F) \qquad (3.2.6)$$

The predicted class of each customer's set of observations $\boldsymbol{B_i}$ is the label, *F*, that would provide the greatest likelihood provided the observed features $\boldsymbol{B_i}$ and model parameters $P(\boldsymbol{B_i}|F)$. We provide a method of how to obtain the model parameters $P(\boldsymbol{B_i}|F)$ subsequently in Section 3.2.3.

3.2.3 Parameter Estimation via Maximum Likelihood Estimation

In this section, we describe the methodology to estimate the parameters of the Naïve Bayesian Classifier, also known as the conditional probabilities and *a priori* probabilities of the model. We define the parameters of the model as,

- $P(F)$ represents the probability of the fraud class label F (0 for healthy, 1 for fraudulent) occurring for the entire population.
- $P(\boldsymbol{B_i}|F)$ represents the conditional probability of observing the features provided the fraud state (F).

Where $\boldsymbol{B_i}$ is a vector representing a set of features b_{ij} for customer *i*.

$$\boldsymbol{B_i} = \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{ij} \end{bmatrix} \tag{3.2.7}$$

Therefore, provided label *F,* each set of observations, $\boldsymbol{B_i}$ can be expressed as,

$$P(\boldsymbol{B_i}|F) = P(F) \prod_j P\left(b_{ij}\middle|F\right)^{b_{ij}} \left(1 - P(b_{ij}|F)\right)^{1-b_{ij}} \tag{3.2.8}$$

To calculate the conditional probabilities of each binary variable b_{ij} provided class label *F*, we apply the following formulas,

$$P\left(b_{ij} = 1\middle|F = 1\right) = \frac{N_{b_{ij}=1}^{F=1}}{N_{F=1}} \tag{3.2.9}$$

$$P\left(b_{ij} = 1\middle|F = 0\right) = \frac{N_{b_{ij}=1}^{F=0}}{N_{F=0}} \tag{3.2.10}$$

$$P\left(b_{ij} = 0\middle|F = 0\right) = \frac{N_{b_{ij}=0}^{F=0}}{N_{F=0}} \tag{3.2.11}$$

$$P(b_{ij} = 1 | F = 0) = \frac{N_{b_{ij}=1}^{F=0}}{N_{F=0}} \quad (3.2.12)$$

Noting that $P(b_{ij} = 1 | F = 0) = 1 - P(b_{ij} = 0 | F = 0)$ and $P(b_{ij} = 1 | F = 1) = 1 - P(b_{ij} = 0 | F = 1)$, we see that the conditional probabilities that maximizes the likelihood in Equations (3.2.9) to (3.2.12) are exactly the proportion of the customers that have the matching b_{ij} classification, to the proportion of customers with the matching F classification. Table 3.2.3 presents the conditional probabilities computed from the observed data.

Feature	$\boldsymbol{P(b_{ij} = 1 \mid F = 1)}$	$\boldsymbol{P(b_{ij} = 1 \mid F = 0)}$
b_{i1}	0.78160920	0.99612564
b_{i2}	0.2011494	0.7964577
b_{i3}	0.1436782	0.3899267
b_{i4}	0.8505747	0.5830912
b_{i5}	0.005747126	0.026982150

Table 3.2.3 Table of conditional probabilities computed from observed data.

Similarly, to calculate the prior probability of any customer being fraudulent, which is our prior probability, we take the proportion of all fraudulent customers to all customers on the platform, expressed as,

$$P(F) = \frac{N_f}{N} = \frac{\#\ of\ total\ fraudulent\ customers}{\#\ of\ total\ customers} \quad (3.2.13)$$

Solving for the probability values, we obtain,

A Priori Probability	**Value**
P(F = 1)	0.04611864
P(F = 0)	0.95388136

Figure 3.2.1 Prior probabilities of fraud for all customers.

There is roughly a 5% chance that a customer is fraudulent in our dataset, as the majority 95% of customers are non-fraudulent. Provided the conditional probabilities, $P(B_i|F)$, provided in Table 3.2.3, we can calculate the posterior probabilities $P(F|\boldsymbol{B_i})$ applying Equation (3.2.3).

i	F	b_1	b_2	b_3	b_4	b_5	$P(F=1\|B_i)$	$P(F=0\|B_i)$	$\hat{F}$	*Result*
1	0	1	1	0	1	0	0.2932	0.7068	0	TN
2	1	1	0	0	1	0	0.8656	0.1344	1	TP
3	0	1	0	0	1	0	0.8656	0.1344	1	FP
4	0	1	1	0	1	0	0.2932	0.7068	0	TN
5	0	1	1	1	0	0	0.0264	0.9736	0	TN
6	0	1	1	0	1	0	0.2932	0.7068	0	TN
7	0	1	1	0	1	0	0.2932	0.7068	0	TN
8	0	1	0	0	1	0	0.8656	0.1344	1	FP
9	0	1	1	1	0	0	0.0264	0.9736	0	TN
10	0	1	1	0	1	0	0.2932	0.7068	0	TN
11	0	1	1	1	0	0	0.0264	0.9736	0	TN
12	0	1	1	1	0	0	0.0264	0.9736	0	TN
13	1	1	0	0	1	0	0.8656	0.1344	1	TP
14	0	1	1	0	1	0	0.2932	0.7068	0	TN

Figure 3.2.2 Sample customer feature values and posterior probabilities.

3.2.4 Naïve Bayes Classifier Performance

Each customer's features in the testing data is used to predict the customer's probability of fraud. Specifically, for fraud detection, we place importance on the *TPR* of the model which is the number of detected positives or true positives *(TP)*, over the total number of fraud cases *(TP + FN)*. High TPR indicates a high detection rate among all cases of fraud, however, our Naïve Bayesian Classifier does not produce the high TPR value that we hope to obtain. The *FPR* value of our classifier, which is the proportion of true fraud among all detected cases fraud, is also significantly high.

Model Performance	Count
TP	70
TN	1763
FP	409
FN	31
FPR	0.1883057
TPR	0.6930693

Figure 3.2.3 Model performance metrics for Naive Bayesian Classifier.

As evidenced, the Naïve Bayesian Classifier performs poorly at obtaining high values of TPR or recall, though it does obtain satisfactory levels of precision (higher precision indicates lower FPR). The example presented in this Section provide a strong motivation to develop a more accurate, and robust classifier that produces higher recall values for the purpose of fraud detection.

3.3 Decision Trees & Random Forest

Decision trees are classification algorithms that provide more flexibility for classification on complex non-linear systems over than parametric models such as regression. Decision Trees are intuitive in nature, and easily interpretable to the general public. Tree based algorithms typically scale well in terms of computational complexity, and are simple to visualize. Unlike regression based methods, there is no need to create dummy variables, as they can easily handle categorical data. However, a downside to decision tree methods is that they are very sample dependent and prone to overfitting. This problem can be mitigated by ensemble learning methods such as the Random Forest algorithm. The application of Random Forest, has been applied to detect online retail fraud in (Altendorf, et al., 2005).

In this section, we introduce the theory and application of the decision tree algorithm, proposed by (Quinlan, 1986), and subsequently provide a basic example of how to build a decision tree from categorical and numerical features in our database. Secondly, we follow closely the model applied in (Altendorf, et al., 2005) to develop a system to detect fraud using the Random Forests algorithm, introduced by (Breiman, 2001), on our data set.

Notation	Definition
S	Split data; data incoming to a decision tree node, and classified in to one of two binary values.

$S_{q,j}$	Split at node q on feature j.
F	True fraud label.
$\hat{F}$	Predicted fraud label.
β	Entropy gain threshold.
$H(S)$	Entropy of the split, S.
P_j	Probability of S being classified as value f (fraud class = 0 , 1).
y	Fraud label (0, 1)
$y`$	Predicted fraud label (0, 1)
q	Node number
n_f	Number of observations at with fraud class f.
j	Feature index.

Table 3.3.1 Decision Tree notation.

3.3.1 Decision Tree

First introduced in (Quinlan, 1986), decision trees are predictive models which map observations through a flow-chart like structure, splitting the incoming features at each intermediate node of the tree. Each node receives an incoming input, and outputs a label based on the feature values. Decision trees can be applied for classification purposes, or regression purposes. For categorical classification, each node is classified with regard to the node's feature value. For numerical variables, an optimal split based on information entropy, divides the incoming observation into sub-classes. In the context of decision trees, we define the following three types of nodes,

Node Type	**Definition**
Root Node	The top-most initial node consisting of no incoming edges and two outgoing edges.
Internal Node	All of the intermediate nodes, consisting of one incoming node and two outgoing nodes.
Terminal Node	The bottom most node of the decision tree consisting of one incoming node and the classification output.

Table 3.3.2 Decision tree node types.

In a decision tree, each terminal node is assigned a class label, in our case (0 for healthy, and 1 for fraud). The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

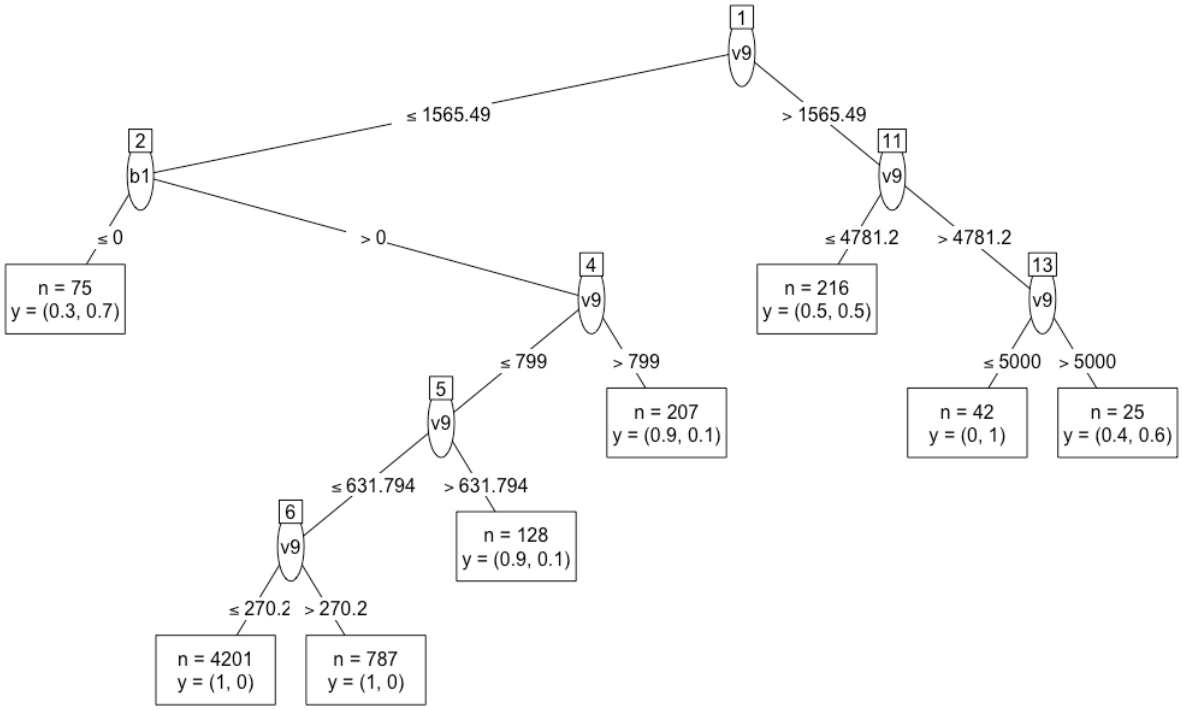

Figure 3.3.1 An example of how a fraud classification decision tree would look like.

In Figure 3.3.1, we construct an example classification tree using one categorical feature, b_1, and one numerical feature, v_9, applying the CART (Classification and Regression Tree) algorithm, introduced in (Breiman, 1984). The CART algorithm splits the data at each node into only two branches only. This prevents the decision tree from overfitting by limiting the number of discrete features each split can be classified as. From this diagram, we note that both categorical features and binary features can be repeatedly split multiple times. To decide on which feature to split on, and at which level, we must find the feature that minimizes the impurity of the split. The measure of impurity, which we define as information entropy, refers to the level of ambiguity at each node, q, after the split S, on feature j, which we denote as $S_{q,j}$. Information entropy measures the purity of the outcome of each decision tree. The concept of entropy, $H(S_{q,j})$, is defined as,

$$H(S_{q,j}) = -\sum_{j} P_j \log_2 P_j \qquad (3.3.1)$$

Where $H(S_{q,j})$ is the entropy of the data, provided split data S. We define the split S as all the incoming data. Each data point can take a numerical value, or categorical value. For numerical values, a split value is defined, and the incoming data is partitioned at the corresponding split value. For example, if the split value is determined to be 5.6, then all data points greater than 5.6 will fall into one class, and all data less than or equal to 5.6 will fall into another class. P_j denotes the probability of obtaining value $F \in (0,1)$ at split S. This probability P_j can be represented as the number of data points classified as fraudulent or non-fraudulent (0, 1) at any given node, we use n_f to denote this number**.** For example, at node 2,

Feature	n_f	P_j
b1 > 0	75	0.01389403
b1 <= 0	5398	0.986106

Table 3.3.3 Split on feature b1.

We see that of the S pertaining to b_1, of the 53985 incoming observations, 75 were labelled as fraud (1) and 5323 were classified as non-fraud (0), of the two possible classes. Therefore, at that node, we compute entropy as,

$$H(S_{2,b_1}) = -\frac{75}{5398}\log_2\frac{75}{5398} - \frac{5323}{5398}\log_2\frac{5323}{5398} = 0.1056226 \qquad (3.3.2)$$

For categorical variables, a split is set at a threshold value and the incoming data is split into partitions based on whether the value exceeds or falls below a set threshold accordingly. For example,

Feature	$\boldsymbol{n_i}$	$\boldsymbol{p_i}$
v9 <= 5000	42	0.6268657
v9 > 5000	25	0.3731343

Table 3.3.4 Split on feature v9.

$$H(S_{13,v_9}) = -\frac{42}{67}\log_2\frac{42}{67} - \frac{25}{67}\log_2\frac{25}{67} = 0.8732 \qquad (3.3.3)$$

When training decision trees, the objective is to find the split that will produce the maximum entropy. We define the concept of information gain as,

$$\nabla(S_{q,j}) \stackrel{\text{def}}{=} H(S_{q,j'}) - \sum_{j} \frac{N(S_j)}{N} H(S_j) \qquad (3.3.4)$$

We define the information entropy gain $\nabla(S_{qj})$ of incoming data S at node q, and feature j. $H(S_{q,j'})$ which is the entropy at node q which was previously split on feature j', calculated by Equation (3.3.1). S_j represents the data at outgoing nodes on feature j (for binary splitting j can take on two values). $N(S_j)$ represents the number of values at node S_j after the split has been made, and N represents the total number of data points incoming from node $S_{q,j'}$, where $\mathrm{N}(\mathrm{S_j})/N$ is the fraction of the number of data points split at feature S_j to the total amount of data at node incoming node $S_{q,j'}$.

For all features, we find the maximum entropy gain at each node by iterating over features j. Since all of our categorical features are split in a binary manner (0, 1), the split on categorical features is predefined. For numerical features, it is necessary to iterate through multiple discrete values of the feature find the split value that produces the highest information entropy. Subsequently, we compute the entropy gain $\nabla(S_{ij})$. An example of this is the split on feature v_9 in Figure 3.3.1. Furthermore, we see that the categorical variable is split at b_1, and subsequently the node is further split by numerical value v_9. Thus it is possible for a single feature, such as v_9 to be split multiple times to compute the predicted value. Evidently, it is possible for the decision tree to continue branching incessantly for decreasing entropy gains $\nabla(S_{q,j'})$, therefore we impose a stopping condition for the feature splitting process. The stopping condition is the case when the maximum entropy gain $\nabla(S)$ of the model is less than a defined threshold β.

$$Max\left(\nabla(S_{q,j'})\right) \leq \beta \qquad (3.3.5)$$

3.3.2 Application of Decision Trees

We fit a decision tree on the categorical, b, and numerical, v, variables in the training set. We produce results, where F is the actual fraud label, and $\hat{F}$ is the forecasted fraud label via decision tree prediction. We display a sample of the predictions results against both binary (categorical) **b** type, and numerical **v** type features. There is a combined total of 5 categorical variables, and 9 numerical variables. We fit a single CART decision tree to the model, consisting of 37 nodes with binary splits, on 5681 customer centric features, which is 75% of the selected data. For validation and model performance, we use the remaining 25% of the data set, denoted as the validation set. We present a sample of the raw data with corresponding response values y, and predicted response values $\hat{F}$, below,

i	b_1	b_2	b_3	b_4	b_5	F	$\hat{F}$
1	1	1	0	1	0	0	0
7	1	1	0	1	0	0	0
8	1	1	0	1	0	0	0
9	1	1	0	1	0	0	0
11	1	0	0	1	0	0	0
15	1	1	1	0	0	0	0
18	1	1	1	0	0	0	0
19	1	0	0	1	0	0	0
21	1	0	0	1	0	0	0
24	1	1	1	0	0	0	0
25	1	1	0	1	0	0	0
31	1	0	0	1	0	1	1
32	1	1	0	1	0	0	0
33	1	1	1	0	0	0	0
37	0	0	1	0	0	1	1

Figure 3.3.2 Categorical binary feature sample values.

i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	F	$\hat{F}$
1	1145000	36	274350.8571	2	31805.55556	693	37	265302.1389	18.72972973	0	0
7	12998000	23	484797.2727	6	565130.4348	8000	8	1147244.286	1000	0	0
8	1066000	14	897189.2308	2	76142.85714	52	12	1060644.364	4.333333333	0	0

9	62669000	44	284641.3953	8	1424295.455	22806	32	401035.8387	712.6875	0	0
11	1490000	5	44745	2	298000	4200	13	100253.5	323.0769231	0	0
15	7429000	10	1331926.667	5	742900	58	2	1732510	29	0	0
18	9665000	38	308130.8108	6	254342.1053	3096	42	248057	73.71428571	0	0
19	2853000	31	418976	3	92032.25806	356	6	1311991.6	59.33333333	0	0
21	5495000	35	97775.29412	7	157000	60	2	422308	30	0	0
24	18334000	51	247200	10	359490.1961	7521	67	191193.4091	112.2537313	0	0
25	1134000	11	922326	5	103090.9091	2107	5	1361338.75	421.4	0	0
31	68691000	36	74201.14286	3	1908083.333	70414	33	139415.4063	2133.757576	1	1
32	1306000	13	843800	8	100461.5385	10	2	857324	5	0	0
33	385000	8	989314.2857	5	48125	80	4	665029.6667	20	0	0
37	2369000	5	1179675	3	473800	85	6	997568.6	14.16666667	1	1

Figure 3.3.3 Numerical feature values and response values.

We provide measures of model performance,

Model Performance	Count
TP (True Positives)	85
TN (True Negatives)	2081
FP (False Positives)	91
FN (False Negatives)	16
FPR (False Positive Rate)	0.04189687
TPR (True Positive Rate)	0.8415842

Figure 3.3.4 Decision tree model performance parameters.

It is evident that the application of a decision tree, with additional numerical features, greatly increases the detection rate of the fraud predictor, when compared to the Naïve Bayesian Classifier mentioned earlier in Section 3.2, however, the decision tree model produces a lower TPR when compared to the logistic regression model presented in Section 3.1.

3.4 Random Forest Classifier

The Random Forest classifier is an extension to the decision tree algorithm, where a collection of trees are utilized to generate a result by taking an average of the results as the final result. Various benchmark studies have indicated that random forests can attain strong performance in terms of predictive power. Random forests have been applied to a wide range of applications, from machine

fault detection (Han & Lee, 2006), financial banking fraud (Liu, et al., 2015), to online retail fraud (Altendorf, et al., 2005). In each use case, random forests have shown excellent performance for robust classification purposes, mitigating the results of overfitting and cross-correlation. Random forests have also shown to have strong predictive capability in small data sets.

Notation	Definition
m	Feature sample parameter.
T	Number of decision trees in the Random Forest.
d	Max depth of each decision tree.
n	Feature sample parameter
y'_t	Predicted response of the t^{th} tree.

Table 3.4.1 Random forests notation.

3.4.1 Random Forests Algorithm

The Random Forest algorithm is an ensemble model. It combines the outputs of several weaker models to generate a more robust prediction. Therefore, instead of generating one decision tree from all of the features, we build T smaller decision trees using only a maximum of m out of j total features. Each small tree, t of T builds a model on n out of N observations form the training set. We formally state the Random Forests algorithm as,

Random Forest Algorithm

1. For each tree t = 1,…,T
    a. Sample with replacement n observations from the training set.
    b. Select m of j total features
    c. Perform the decision tree generation algorithm presented in Section 3.3.2, with a maximum tree depth stopping condition of d[5].
2. Proceed to the *prediction step*.

[5] The maximum stopping condition is used to prevent memory overflow if the decision tree becomes too large. Ideally, the decision tree is grown until the entropy gain is below β.

Table 3.4.2 Random Forest Algorithm outline.

The *prediction step* involves generating a prediction from the average value of the prediction out of *T* trees of the random forest. Where the prediction *R(F)* is,

$$R(F) = \frac{1}{T}\sum_{t=1}^{T} y_t' \qquad (3.4.1)$$

Where y_t' is the predicted response of the t^{th} tree in the Random Forest. The output of the Random Forest algorithm is essentially the average output among all of the outputs of the trees contained within the random forest. For categorical prediction, we set a threshold of 0.4 (to obtain an FPR of under 0.05), where,

$$R(F) > 0.4 \rightarrow F = 1 \qquad (3.4.2)$$

$$R(F) \leq 0.4 \rightarrow F = 0 \qquad (3.4.3)$$

Equations (3.4.2) and (3.4.3) express the predicted fraud of each customer's observation as a majority vote among *T* trees of the random forest. Table 3.4.3 conveys the selection of model parameters, *m* and *T*.

Model Parameters	**Value**
m	4
T	500

Table 3.4.3 Random forest model parameters,

We present the classification results of this model,

Model Performance	**Count**
TP (True Positives)	94
TN (True Negatives)	2081
FP (False Positives)	91
FN (False Negatives)	7
FPR (False Positive Rate)	0.04189687
TPR (True Positive Rate)	0.9306931

Table 3.4.4 Random forest model performance.

The model performance is drastically improved using the Random Forest method. Using 500 smaller decision trees, with observations constantly sampled with replacement[6] from the training data set using a subset of the features *m*, we create a much more robust classifier that greatly reduces bias towards any specific feature.

3.5 Model Comparison

In this section we compare the performance of the 5 classifiers, which we refer to as *aggregate feature classifiers*, due to the fact that the features are aggregate over the entire lifespan of the customer.

Model Identifier	Model Name
LR1	*Logistic Regression on Customer Registration Features*
LR2	*Logistic Regression on All Available Features*
NB1	*Naïve Bayesian Classifier on Customer Registration Features*
DT1	*Decision Tree on All Available Features*
RF1	*Random Forest on All Available Features*

We compare the performance of multiple models by examining the classification accuracy of each algorithm as measured by the detection rate, which is constrained by the *FPR*. We also examine the behaviour of the classifier based as the overall shape of the ROC, refer to figure Figure 3.5.1. Inevitably, classifiers which make use of both registration (categorical) and numeric features perform better than classifiers which only use registration features (i.e. *LR1* and *NB1*).

Though a single decision tree, as represented by model *DT1*, is prone to deficiencies such as variable bias and overfitting. *RF1* provides best detection rate among all classifiers presented in

[6] Sampling with replacement is also known as 'bootstrap' sampling in machine learning terminology.

this section. The Random Forest model serves as a major improvement to a single decision tree, and is evidently able to produce a better detection rate for cases of fraud.

When applied without additional higher order interaction terms, Logistic Regression, represented by *LR2* & *LR1*, can be hindered by multicollinearity as it does not consider any complex variable interactions. However, adding in higher order interaction terms greatly increases the complexity of the model. *LR2* considers all variables to be independent of one another, whereas *RF1* is capable of reducing correlation of variables via its robust sampling mechanism. *LR2* does exhibit a relatively sufficient detection rate despite the assumption that all variables are independent.

LR2 exhibits similar ROC behaviour to *RF1*, although *RF1* produces better accuracy provided the 0.05 maximum *FPR* constraint, the variation is not drastic across the entire curve, and in fact, there are some instances where *LR2* will outperform *RF1* provided different *FPR* constraint. Because Logistic Regression can be expressed in parametric form, it is simpler to deploy across complex systems. We propose that both *RF1* and *LR2* can be used interchangeably as *aggregate feature classifiers*. Though *RF1* is preferred due to its stronger accuracy, *LR2* is less complex to implement and deploy across a real-time system.

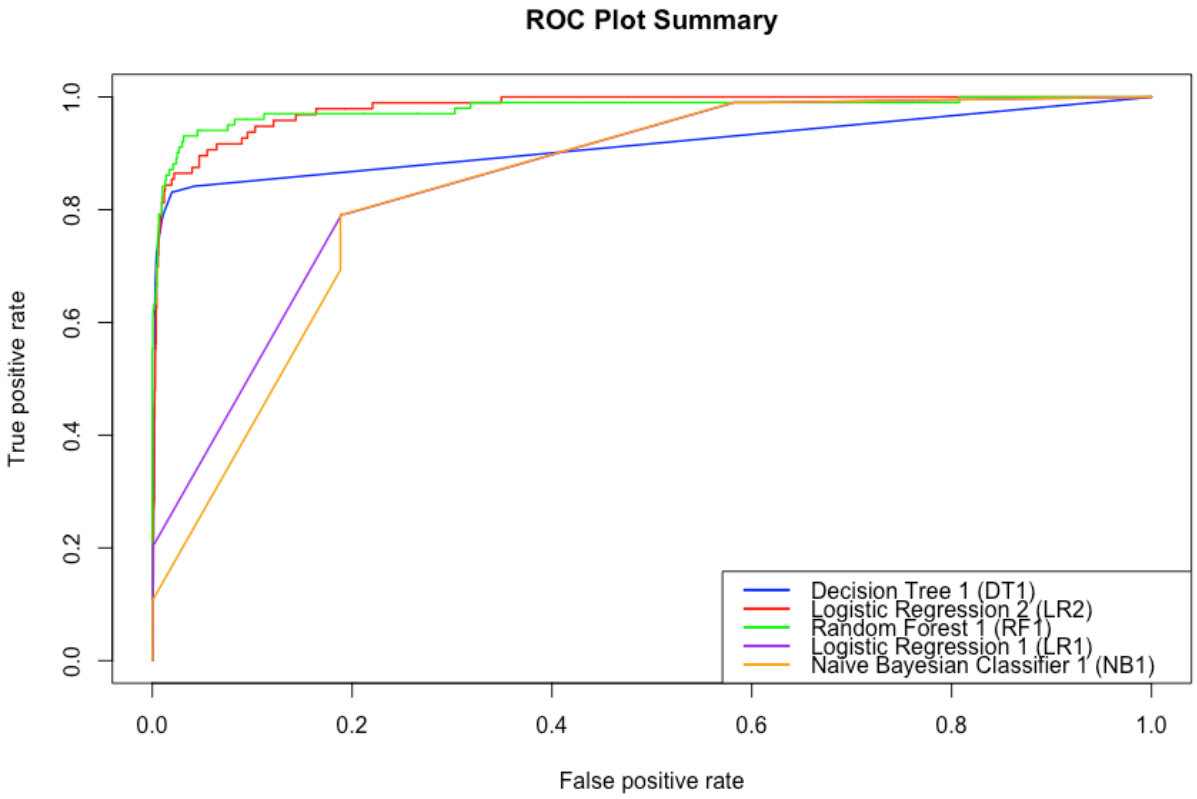

Figure 3.5.1 ROC comparison of various classifiers.

Chapter 4. Hidden Markov Modelling

Discrete time Hidden Markov Models (HMM) are stochastic models which have a wide range of applications for modeling stochastic processes in various industries. Discrete time HMM's are ideal for modeling discrete auto-correlated processes, where the observed variables depend on a *hidden state*. The hidden states are unobservable, and obey the Markov property, indicating that the conditional probability of future states depends only on the present state. The observable symbols at the current time epoch, t, are conditionally dependent only on the hidden state at the time t. The purpose of this chapter is to further outline the theoretical details of HMM's. From an application standpoint, HMM's have been applied in industry to many practical use-cases, from speech recognition (Rabiner & Juang, 1986) to image analysis (Choi, et al., 2000). The wide application of stochastic process modelling can be found in (Zucchini & MacDonald, 2009), which illustrates many use cases of Hidden Markov Modelling, including crime rate modeling and DNA sequencing.

In this thesis, we refer to non-fraudulent customers as *healthy customers*. If an incoming transaction is not accepted by the classification algorithm, the transaction is deemed fraudulent, and so is the customer associated with that transaction. The customer responsible for the transaction is subsequently subject to human review from a fraud review team, to confirm if that customer is fraudulent. Consequently, that customer is terminated from making further transactions. Therefore, it is critical to ensure that the false positive rate (FPR) remains below a certain threshold (in this project we stipulate a maximum *FPR* of 5%), so that minimal effort is expended in correcting misclassified fraudulent customers.

4.1.1 Motivation and Application

The application of HMM's specifically for the application of fraud detection is presented in both (Srivastava, et al., 2008) and (Dhok & Bamnote, 2012), which we base our methodology from. First, a reference model is constructed using data from regular, non-fraudulent, customers. And subsequently, this model is compared against transactions generated from both healthy and fraudulent customers to determine the TPR against the FPR of the algorithm. Both (Srivastava, et

al., 2008) and (Dhok & Bamnote, 2012) illustrate that for purposes of credit card fraud detection, HMM's produce a high detection rate combined with a low false positive rate. However, (Raj & Portia, 2011) presents a comparative study considering the application of HMM and other methods of classification, claiming that the application (Netzer, et al., 2008)of HMM to fraud detection instead produces a high false positive rate, and is relatively ineffective. In our research, we seek to further investigate the application of HMM to effectively detect the occurrence of fraud specifically on eCommerce networks. Our application differs slightly from credit card fraud detection specifically because although the majority of eCommerce transactions are credit card based, such credit card transactions occur entirely online. However, the principles of financial transaction-based stochastic modelling remain the same.

We refer to the models previously developed in Chapter 3 as *aggregate feature classification* models, provided that they require an aggregate of all time series observations over a customer's lifetime to generate a single feature. There are two major set-backs with this approach. Firstly, such models do not consider the dynamic time-variant behaviour of customers. Secondly, such models fail to detect anomalous behaviour until the cost incurred by such behaviour becomes overwhelmingly detrimental. This motivates the development of a model which provides consideration for the dynamic behaviour of healthy customers, and provides a signal for early action if an anomaly spending pattern is detected.

This chapter details the methodology of constructing an HMM from the observed data belonging to healthy customers only, we refer to this as the *reference model*. The reference model is subsequently used to generate a probability of observing a sequence produced by each respective customer. We update a new observation symbol to a customer's sequence at every 12-hour interval. With each additional observation, an HMM-based fraud detection algorithm used to determine if the customer is fraudulent, this will be further elaborated in Chapter 5. The periodic update allows for early detection of fraudulent behaviour. Furthermore, HMM's provide the ability to forecast the behavioral state of each customer into the future. For example, (Netzer, et al., 2008) applies the use of HMM's to model customer behavior, providing a framework for customer-relation-management (CRM). Though this additional benefit is not specifically addressed in our

research, the HMM's capability for generating forecasts should be noted as an advantage over aggregated models. In our application, we choose to develop the HMM for the specific purpose of fraud detection on financial transactions for eCommerce platforms, by setting the hidden states to describe the spending behaviour of each healthy customer, and calculating the acceptance probabilities of each customer based on the observation symbols.

The HMM considers both the dynamic time-variant behaviour of customers and is capable of providing a signal for early preventive action. In addition, the HMM also outlines solid statistical foundation for the understanding of customer behaviour as the structure of the HMM represents the real spending behaviour of customers on the eCommerce platform. The Markov property of the hidden state in an HMM accounts for the auto-correlated behaviour of customer spending. Since the social-economic factors that contribute to the amount of spending produced by a customer are not directly observable to the eCommerce service, it is ideal to use a hidden state to model this process. The *hidden state* represents the true nature of the customer at time t, and consequently affects the actual spending of the customer, represented by the observation symbols of the HMM at time t. The main concept behind the HMM is to assign a probability to the observation symbol sequence, and determine how likely the aforementioned sequence is representative of a healthy customer.

4.2 Hidden Markov Model (HMM)

Notation	Definition
S_t	Hidden state at time *t*.
θ	Parameters of the HMM.
c	Customer index.
O_t	Emitted observation symbol at time *t*.
$O_{1:T_c}^{(c)}$	Observation symbols from $t = 1$ to $t = T_c$ for customer *c*.
i	Value of hidden state.
j	Value of hidden state.
Q'	Set of all permutations of sequence $\{S_1, S_2, \dots, S_T\}$.
N	Total number of hidden states S_t.
C	Total number of customers (or observation sequences).
M	Total number of observation symbols O_t.
$Q = \{1,2, \dots, N\}$	Set of all possible hidden states.
$V = \{v_1, v_2, \dots, v_M\}$	Set of all possible observation symbols.
$\alpha_t(j)$	Forward probability at time *t*.
$\beta_t(j)$	Backward probability at time *t*.
$\zeta_t(i,j)$	Joint probability of being in state *i* at time t, to being in state *j* time *t +1*, provided the observation sequence.
$A = \{a_{ij}\}$	Transition matrix A, representing transition probability from hidden state *i* to *j*.
$B = \{b_j(v_k)\}$	Emission probability of observation symbol probability, v_k, at hidden state *j*.
$\pi = \{\pi_i\}$	Initial state probability distribution of state *i*.
$A' = \{a_{ij}\}$	Transition matrix representing transition probability from hidden state *i* to *j* for all C customers.
$B' = \{b_j(v_k)\}$	Observation symbol probability at hidden state *j* for all C customers.
$\pi'(i)$	Initial probability distribution matrix generated from multiple sequences for all C customers.
T_c	Final time epoch for customer c.
$\boldsymbol{O}$	Set of all observation sequences.
n	Iteration number of the Baum – Welch Algorithm.

Table 4.2.1 Hidden Markov Model notation.

4.2.1 HMM Model Structure

Hidden Markov Models constitute an extension to the Markov model, where the observed variables are dependent on both an underlying hidden state, and the preceding hidden state. The hidden states of an HMM follow a multi-state Markov property. We denote the hidden state at time *t*, S_t, as a condition representing spending behaviour of the customer.

$$P(S_{t+1}|S_1, \dots, S_t) = P(S_{t+1}|S_t) \qquad (4.2.1)$$

We refer to $P(S_{t+1}|S_t)$ as the transition probability of the hidden state. At each time epoch *t*, the arrival of a new observation occurs. Provided *N* hidden states, the HMM can be expressed as a sequential state process, where the hidden state sequence constitutes a Markov process.

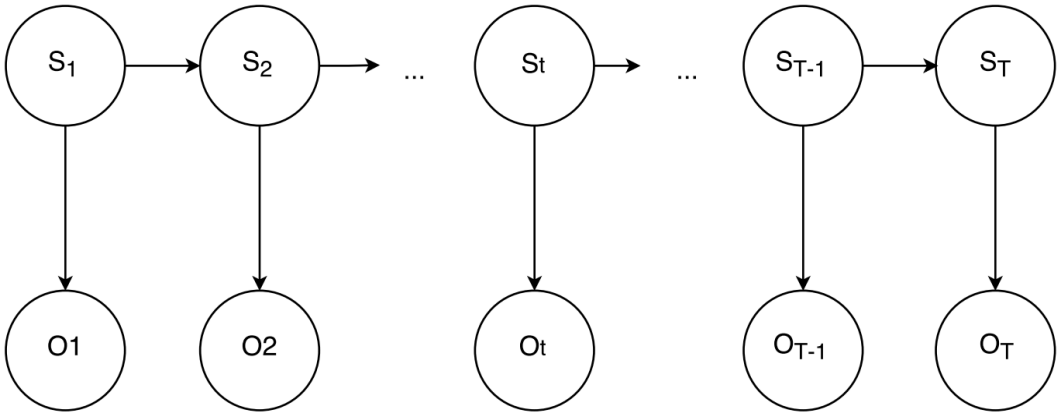

Figure 4.2.1 Structure of an HMM, displaying the hidden state S and observed variable y.

To begin with, we examine the observation sequence pertaining to one customer, *c*, specifically. We refer to $O_{1:T}$ as a *single observation sequence*. The Hidden Markov model is constructed from three parameters, denoted as $\theta = (A, B, \pi)$. Where $\pi[\theta]$ is the initial *N* x 1 probability matrix, for *N* hidden states. We see that, $\pi_i = P(S_1 = i)$, for i from 1 to *N*.

$$\pi[\theta] = \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_N \end{bmatrix} \qquad (4.2.2)$$

By convention, we define at time *t*,

$$S_t = i \qquad (4.2.3)$$

And,

$$S_{t+1} = j \tag{4.2.4}$$

Where the hidden state corresponding to j is always one time epoch ahead of i. $A_c = \{a_{ij}\}$ is the transition matrix for observation sequence $O_{1:T}$, that defines the transition probabilities from hidden states i to j, where, $a_{ij} = P(S_{t+1} = j | S_t = i)$. We can summarize the transition probabilities conveniently as an N x N matrix A.

$$A = \begin{bmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NN} \end{bmatrix} \tag{4.2.5}$$

And $B = \{b_i(v_k)\}$ represents the emission probability matrix for observation sequence $O_{1:T}$. Where $b_i(v_k) = P(O_t = v_k | S_t = i)$, for N hidden states, and M observation symbols. We summarize this as the N x M emission probability matrix B.

$$B_c = \begin{bmatrix} b_1(v_1) & \dots & b_1(v_M) \\ \vdots & \ddots & \vdots \\ b_N(v_1) & \dots & b_N(v_M) \end{bmatrix} \tag{4.2.6}$$

In our HMM, we impose a model where the set of all possible observation symbols and hidden states are identical for all independent observation sequences, $O_{1:T}$. We specify $M = 4$ observation symbols and $N = 3$ hidden states, this will be further discussed in Chapter 5.

4.2.2 HMM Parameter Estimation

The estimation procedure of the HMM consists of three key problems which must be resolved in order to obtain the parameters of the model (Rabiner & Juang, 1986).

1. *Evaluation Problem* – Provided parameters $\theta = (A, B, \pi)$, how do we compute the maximum probability of the observation sequence $P(O_{1:T}|\theta)$?
2. *Expectation Problem* – Selecting an optimal hidden state sequence, $S_{1:T}$, according to some meaningful metric, provided θ.

3. *Maximization Problem* – How to maximize the probability of the sequence $S_{1:T_c}$ by adjusting parameters $\theta = (A, B, \pi)$.

We propose the solution to these fundamental problems by presenting the *forward* and *backward* algorithms, along with the *Baum-Welch Algorithm*, presented subsequently in this chapter.

4.3 Forward-Backward Algorithm

Naively, we can compute the evaluation problem by summing over all of the combinations constituting the hidden state sequence. We express this as,

$$P(O_{1:T}|\theta) = \sum_{\{S_1, S_2, \dots S_T\} \in Q'} \pi_{S_1} b_{S_1}(O_1) a_{S_1 S_2} b_{S_2}(O_2) \dots a_{S_{T-1} S_T} b_{S_T}(O_T) \qquad (4.3.1)$$

Where Q' denotes all permutations of the sequence of hidden states $\{S_1, S_2, \dots, S_T\}$. We can see that computing $P(O_{1:T}|\theta)$ in this manner requires heavy computation, as the number of permutations of Q' will be on the order of $2TN^T$. To reduce the computational complexity using a dynamic programming approach, we introduce the *forward* and the *backward* algorithms.

4.3.1 Forward Algorithm

The *forward algorithm* involves the calculation of the joint probability of being in state S_t, provided all of the observation symbols up until time *t*, denoted as $O_{1:t}$. We define $S_{t-1} = i$ and $S_t = j$, with state *j* conditionally dependent on *i*. Provided the information about the entire observation sequence, $O_{1:t}$. We express $\alpha_t(j)$ as,

$$\alpha_t(j) = P(S_t = j, O_{1:t}) \qquad (4.3.2)$$

$$\alpha_t(j) = \sum_{w=1}^{N} P(S_t = j, S_{t-1} = i, O_{1:t})$$

Due to conditional independence, we can simplify the expression to,

$$\alpha_t(j) = \sum_{i=1}^{N} P(O_t|S_t = j)P(S_t = j|S_{t-1} = i, O_{1:t-1}) \tag{4.3.3}$$

$$\alpha_t(j) = \sum_{i=1}^{N} P(O_t|S_t = j)P(S_t = j|S_{t-1} = i)P(S_{t-1} = i, O_{1:t-1})$$

The forward algorithm allows us to apply recursion to compute, at each step, the current value recursively utilizing the forward probability from the previous time epoch.

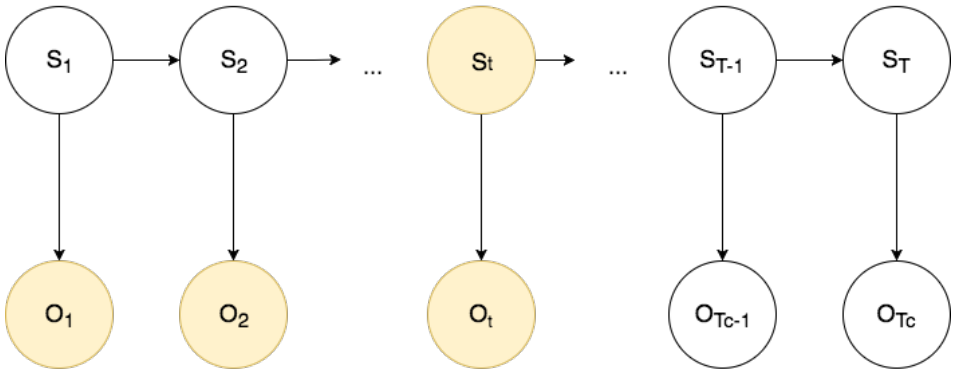

Figure 4.3.1 An illustration of the states involved in calculating the forward probability.

The forward algorithm can be applied recursively as,

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}\right] b_j(O_t) \tag{4.3.4}$$

For purposes of initialization, we compute,

$$\alpha_1(j) = P(S_1 = j, O_1) = b_j(O_1)\pi_j \tag{4.3.5}$$

And $P(O_{1:T}|\theta)$ simplifies to,

$$P(O_{1:T}|\theta) = \sum_{i=1}^{N} P(O_{1:T}, S_T = i) = \sum_{i=1}^{N} \alpha_T(i) \qquad (4.3.6)$$

Applying the forward algorithm to calculate $P(O_{1:T}|\theta)$ reduces the time of computation for $P(O_{1:T}|\theta)$ from $2TN^T$ to N^2T (Rabiner & Juang, 1986), allowing for a more computationally efficient calculation of the observation sequence probability at any time *t*.

4.3.2 Backward Algorithm

We define the backward probability as the probability of observing future observations from time *t + 1* to time *T*, denoted as $O_{t+1:T}$, conditional upon knowing S_t. By convention we define $i = S_t$ and $j = S_{t+1}$.

$$\beta_t(i) = P(O_{t+1:T}|S_t = i) \qquad (4.3.7)$$

$$\beta_t(i) = \sum_{j=1}^{N} P(O_{t+1:T}, S_{t+1} = j|S_t = i)$$

$$\beta_t(i) = \sum_{j=1}^{N} P(O_{t+2:T}|S_{t+1} = j)P(O_{t+1}|S_{t+1} = j)P(S_{t+1} = j|S_t = i)$$

Where the backward probability of $\beta_t(i)$ is equivalent to the marginal sum over all possible states $S_{t+1} = j$, multipled by its transition probability, its emission probability, and the backward probability $\beta_{t+1}(j)$.

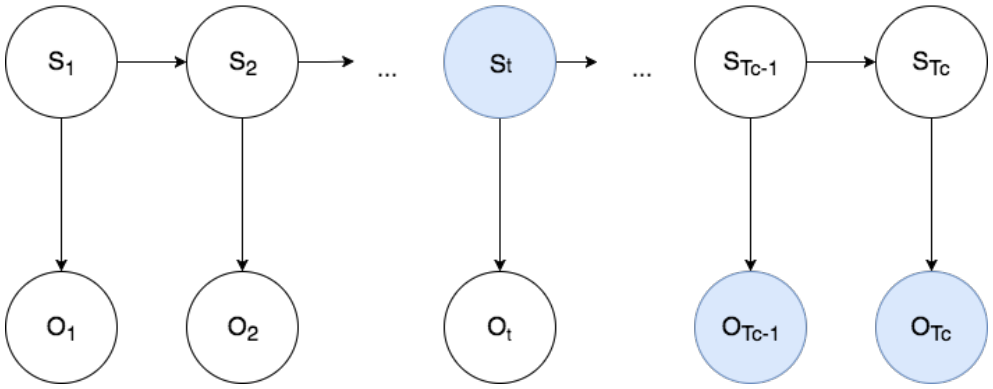

Figure 4.3.2 An illustration of all the observations and states involved in computing the backward probability.

We express $\beta_t(i)$ recursively, with reference to the emission matrix $b_j(O_t)$ and transition matrix a_{ij} probabilities as,

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) b_j(O_{t+1}) a_{ij} \qquad (4.3.8)$$

In order to be able to compute $P(O_{1:t}|\theta)$ for any *t*, we impose the final end condition for T_c, where,

$$\beta_T(i) = 1 \qquad (4.3.9)$$

4.3.3 Forward-Backward Algorithm

The Forward-Backward (FB) Algorithm provides a means of inferring, from an observation symbol sequence, of being in state S_t at time *t*, provided the full sequential observations up to time *T*, represented by $O_{1:T} = (O_1, O_2, \ldots O_T)$. The FB algorithm calculates the complete probability over the entire sequence as the product of the forward and backward algorithms at time *t*. Combining both the forward and backward algorithms, we can express the joint probability of the hidden state at time *t*, denoted by S_t, in terms of the forward and backward probabilities.

$$\alpha_t(i)\beta_t(i) = P(S_t = i, O_{1:T}) \qquad (4.3.10)$$

We can denote the conditional probability of S_t being in hidden state *i* as,

$$\gamma_t(i) = P(S_t = i|O_{1:T}) \qquad (4.3.11)$$

We express $\gamma_t(i)$ as the joint probability of being in state i and observing $O_{1:T}$ while at time *t*, divided by the probability of observing the sequence $O_{1:T}$.

$$\gamma_t(i) = \frac{P(S_t = i, O_{1:T})}{P(O_{1:T})} \quad (4.3.12)$$

Which can be expressed as the probability of *i*, divided by the marginal probability across all *N* possible hidden states q_w.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \quad (4.3.13)$$

The forward-backward algorithm reduces the complexity of computing the conditional probability of being in state *i* at time *t* provided the full observation sequence $O_{1:T_c}$.

4.3.4 Baum-Welch Algorithm

The Baum-Welch algorithm, developed by (Baum & Welch, 1966), is an application of the Expectation Maximization (EM) algorithm to estimate the transition and emission probabilities denoted by matrices *A* and *B* respectively (a more detailed description of the EM algorithm can be found in the Appendix). The EM algorithm presents an iterative solution to calculate *expectation* and *maximization* steps of HMM parameter estimation. We denote the transition from hidden state *i* to state *j* at time *t* as, $\zeta_t(i,j)$,

$$\zeta_t(i,j) = P(S_t = i, S_{t+1} = j | O_{1:T}) \quad (4.3.14)$$

We express $\zeta_t(i,j)$ as,

$$\zeta_t(i,j) = \frac{\alpha_t(i)P(S_{t+1} = j|S_t = i)P(O_{t+1}|S_{t+1} = j)\beta_{t+1}(j)}{P(O_{1:T})} \quad (4.3.15)$$

Where $P\left(O_{1:T_c}\right)$ is the sum of the marginal probabilities across hidden states *i* and *j*,

$$P(O_{1:T}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) P(S_{t+1} = j | S_t = i) P(O_{t+1} | S_{t+1} = j) \beta_{t+1}(j) \quad (4.3.16)$$

Subsequently, denoting a_{ij} as the transition matrix probability, and $b_j(S_{t+1})$ as the emission matrix probability, $\zeta_t(i,j)$ becomes,

$$\zeta_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(S_{t+1}) \beta_{t+1}(j)}{P(O_{1:T})} \quad (4.3.17)$$

4.4 Single Sequence Parameter Estimation

We consider the scenario where we have one sequence of observations from times 1 to T_c. At each step *n*, we compute the parameters of the HMM, $\theta^{(n)} = (\pi^{(n)}, A^{(n)}, B^{(n)})$. For the first iteration *n = 0*, we build a random set of parameters for $\theta^{(0)}$, and update the parameters at each iteration. By applying relative frequencies, we can determine the initial state distribution, π_i, at time *t = 1*.

$$\pi_i^{(n)} = \gamma_1(i) \quad (4.4.1)$$

We can calculate the transition probability, A_{ij}, outlined by the transition probability, which can be expressed as the total expected number of transitions from state *i* to state *j*, over the expected number of transitions from state *i*, at time step *t*.

$$A_{ij}^{(n)} = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (4.4.2)$$

Similarly, for the emission matrix we can calculate the emission probabilities denoted by matrix $B_i(v)^{(n)}$, at each iteration *n*,

$$B_i(v_k)^{(n)} = \frac{\sum_{t=1}^{T} 1(O_t = v) \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad (4.4.3)$$

Where $1(O_t = v)$ is the indicator function indicating whether-or-not the observed symbol O_t, is equal to v_k. We repeat the steps until convergence of $\theta^{(n)} = (\pi_c^{(n)}, A_c^{(n)}, B_c^{(n)})$ via the Baum-Welch algorithm with respect to model accuracy.

4.5 Multiple Sequences HMM Parameter Estimation

The time series transaction histories of all customers are not contained in one continuous time series sequence because each customer, *c*, exhibits a specific sequence of observations that begins at the first time step. Therefore, it is necessary to modify the Baum-Welch algorithm to allow it to estimate the parameters for multiple sequences. Each independent sequence, denoted as $O_{1:T_c}^{(c)}$, contains the sequence of observation symbols for customer c.

$$O_{1:T_c}^{(c)} = (O_1^{(c)}, O_2^{(c)}, \dots, O_{T_c}^{(c)}) \qquad (4.5.1)$$

We assume that all observation sequences, $O_{1:T_c}^{(c)}$, are independent of one another for all *c* customers. However, all independent sequences are modeled the same HMM parameters θ. Given this, we can express the probability of observing all individual sequences $\mathrm{P}(\boldsymbol{O})$, as,

$$\mathrm{P}(\boldsymbol{O}) = \prod_{c=1}^{C} P(O_{1:T_c}^{(c)}|\theta) \qquad (4.5.2)$$

For multiple observation sequences, we calculate the initial probability for state *i* as the simple average of initial probability across all observation sequences via the Baum-Welch Algorithm.

$$\pi'(i)^{(n)} = \frac{\sum_{c=1}^{C} \gamma_1(i)^{(c)}}{C} \qquad (4.5.3)$$

For each customer's observation sequence, we denote $Z(i,j)^{(c)}$ as the expected number of transitions from state *i* to state *j* for the entire time series $1{:}\,T_c$ specific to the observation sequence pertaining to customer *c*.

$$Z(i,j)^{(c)} = \sum_{t=1}^{T_c-1} \zeta_t(i,j) \qquad (4.5.4)$$

The transition probability matrix A_{ij} is calculated as the sum of the expected number of transitions from state *i* to *j*, represented as $Z_t(i,j)^{(c)}$, divided by the expected number of transitions from *i*, represented by $\sum_{t=1}^{T_c-1} \gamma_t(i)$, for each customer c.

$$A'^{(n)}_{ij} = \frac{\sum_1^C Z(i,j)^{(c)}}{\sum_1^C \sum_{t=1}^{T_c-1} \gamma_t(i)^{(c)}} \qquad (4.5.5)$$

The emission probability matrix $B_i(v_k)$ is computed as the expected number of instances where the observation symbols O_t had been equal to v_k provided the hidden state *i*, divided by the expected number of instances where *i* occurred.

$$B'_i(v_k)^{(n)} = \frac{\sum_{c=1}^{C} \sum_{t=1}^{T_c} 1(O_t = v_k)\gamma_t(i)^{(c)}}{\sum_{c=1}^{C} \sum_{t=1}^{T_c} \gamma_t(i)^{(c)}} \qquad (4.5.6)$$

Similar to the parameter estimation step for a single observation sequences for multiple independent observation sequences, we perform a parameter update on the model parameters $\theta^{(n)} = (\pi'^{(n)}, A'^{(n)}, B'^{(n)})$ for all customers' observation sequences for *n* iterations until convergence of the Baum-Welch Algorithm with respect to model accuracy.

Chapter 5. Customer Classification via Hidden Markov Model

This chapter investigates the construction of a Hidden Markov Model designed to detect fraud on eCommerce networks by closely examining financial transactions. Firstly, it outlines the feature construction phase where time series observations, on a per transaction basis, are aggregated to produce what we referred to as *window features*. Subsequently, we apply a clustering algorithm to categorize each transaction summary into an *observation symbol* or *centroid*, which describes the relative spending behaviour of the customer's transactions, allowing for discrete HMM modelling. From each healthy customer's observation symbols we construct an HMM via the Baum-Welch algorithm. We combine the HMM classifier with a logistic regression model constructed from customer registration features to build a robust fraud classifier capable of detecting fraud in real-time, concurrent with the arrival of transactions within a fixed time window, *w*. Subsequently, we evaluate the performance of the classifier on its ability to maximize the detection rate (or TPR) provided a constraint on the false positive rate (FPR) in order to provide an online fraud detection procedure to detect fraud in an automated fashion.

5.1 Time Series Feature Construction

Notation	Definition
c	Customer index.
C	Total number of customers.
t	Timestamp.
Δt	Time interval length, we specify 12 hours.
w_t	Time window index. Represents the time interval $[t, t + \Delta t)$.
$x_i^{(w_t)}$	i^{th} cash flow transaction within time window w_t.
x_i	i^{th} cash flow transaction within a customer's complete transaction history.
n	Number of transactions within each time window w_t.
$\Phi_d(w_t)$	Window function that combines the set of transaction $\{x_1^{(w_t)}, \dots, x_n^{(w_t)}\}$ to a single observation.
$\{\Phi_1, \dots, \Phi_d\}$	Set of *d* aggregation functions, where *d* = 6.

$\Phi(w_t)_c$	Vector of window functions of $\{\Phi_1, \dots, \Phi_d\}$ over time window w_t for customer c.
N_{T_c}	Number of time windows within $T_c - t_c^1$, rounded down to the nearest multiple of Δt.
T_c	Timestamp of final transaction for customer c.
t_c^1	Timestamp of initial transaction for customer c.
t_c	Timestamp of nearest multiple of T_c, less than T_c. Defined as $t_c^1 + N_{T_c} * \Delta t$.
ΔT_c	Timespan of a customer's entire transaction history.
$\Lambda_c^{(HM)}$	The sequence of all window functions, $\{\Phi_1, \dots, \Phi_6\}$, over all time windows pertaining customer *c*, for *marketplace* transactions.
$\Lambda_c^{(HW)}$	The sequence of all window functions, $\{\Phi_1, \dots, \Phi_6\}$, over all time windows pertaining customer *c*, for *virtual wallet* transactions.
$\boldsymbol{x}^{(w_t)}$	Represents $\{x_1^{(w_t)}, \dots, x_n^{(w_t)}\}$ which is the set of all transactions within w_t.

Table 5.1.1 Notation for feature time windowing.

5.1.1 Aggregation Functions

For financial fraud detection, it is a standard practice to define a set of time windows over a fixed time frame, where transactions belonging to that time frame are aggregated within a specific window, we refer to this as *time windowing*. (Whitrow, et al., 2009) demonstrated that financial fraud classification using a transaction aggregation approach is more feasible and proposes a window period of $\Delta t = 1,3,7$ days. There are several reasons why we perform time windowing on our transaction data.

- Certain combinations of transactions may be more indicative of fraudulent behaviour than single transactions alone. Time windowing allows us to capture the behaviour of multiple transactions within a single time window.
- Enforces a uniform time increment, Δt, to provide an expected time period indicative as to when each decision is to be made.
- Provides the capability for multiple time series sequences be aligned meaningfully. All features are indicative of spending within a Δt time period.
- Accounts for the time in between transactions, as well as the amount of each transaction, in a simple and feasible manner (instead of two features to model both transaction amount and arrival time, one feature captures both sources of information).

In our research, we opt for a 12-hour time interval for Δt instead of $\Delta t = 1,3,7$ days, due to the high volume of eCommerce transactions and also the necessity for a more expedited decision process. A window function is applied over a moving time window. Since Δt is constant, we can define the time window, $[t, t + \Delta t)$, simply as w_t, where, w_t denotes both the time at the beginning of the time window, up until the end of the time window $t + \Delta t$.

$$w_t \coloneqq [t, t + \Delta t) \tag{5.1.1}$$

We define $\Phi(w_t)$ as some function over all observations $\{x_1^{(w_t)}, \dots, x_n^{(w_t)}\}$ within time window w_t, aggregating $\{x_1^{(w_t)}, \dots, x_n^{(w_t)}\}$ to a single value defined as $\Phi(w_t)$.

$$\Phi(w_t) \coloneqq \Phi(\{x_1^{(w_t)}, \dots, x_n^{(w_t)}\}) \tag{5.1.2}$$

Φ is an aggregate function that operates on the transactions within each time interval. Each aggregate function Φ provides distinct information about the spending behaviour of the customer at its respective time window, w_t. For each discrete time window, w_t, the set $d = 6$ where there are 6 window functions $\{\Phi_1, \dots, \Phi_6\}$ summarizes all the numerical value data points.

Window Function Φ	Notation	Definition
Sum	Φ_1	$\Phi_1(w_t) = \sum_{i=1}^{n} x_i^{(w_t)}$
Count	Φ_2	$\Phi_2(w_t) = n$
Arithmetic Mean	Φ_3	$\Phi_3(w_t) = \frac{1}{n}\sum_{i=1}^{n} x_i^{(w_t)}$
Max	Φ_4	$\Phi_4(w_t) = Max\left(x_1^{(w_t)}, \dots, x_n^{(w_t)}\right)$
Min	Φ_5	$\Phi_5(w_t) = Min\left(x_1^{(w_t)}, \dots, x_n^{(w_t)}\right)$

Sample Standard Deviation	Φ_6	$\Phi_6(w_t) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i^{(w_t)} - \Phi_3(w_t)\right)^2}$

Table 5.1.2 Table of aggregation methods.

We present an illustration of a single customer's transaction sequence on simulated data to demonstrate the application of time windowing. Suppose a single customer performs a total of 4 transactions in their lifetime. In the example, the timespan of the customer is divided into 4 x 12-hour time windows, w_t.

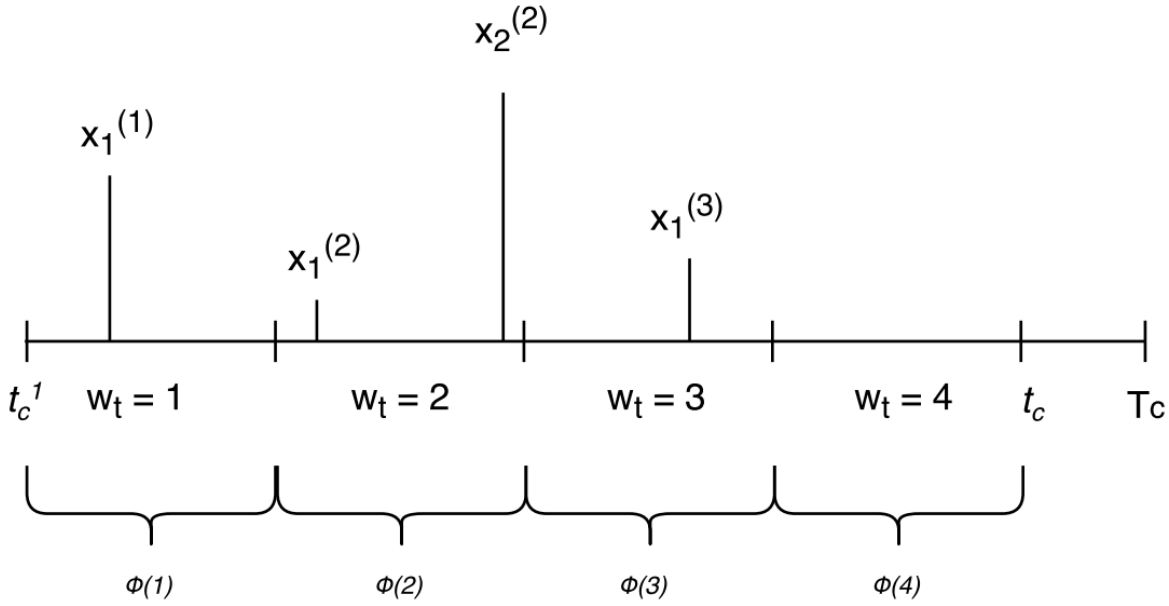

Figure 5.1.1 Graphical illustration of the time window procedure for single customer's transaction sequence.

Figure 5.1.1 illustrates a window function being applied on all time windows pertaining to a single customer's transaction sequence. There are 6 window functions, $\Phi = \{\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, \Phi_6\}$, that operate within each time window. We previously define the set of time window functions, Φ, in Table 5.1.2. The values of Φ and $\boldsymbol{x}^{(w_t)}$ for a single customer's transaction sequence are illustrated Table 5.1.3.

w_t	Transaction Values	$\Phi_1(w_t)$	$\Phi_2(w_t)$	$\Phi_3(w_t)$	$\Phi_4(w_t)$	$\Phi_5(w_t)$	$\Phi_6(w_t)$
0	$\left\{x_1^{(1)}\right\} = \{2000\}$	2000	1	2000	2000	2000	0
1	$\left\{x_1^{(2)}, x_2^{(2)}\right\} = \{500, 3000\}$	3500	2	1750	500	3500	1767.8
2	$\left\{x_1^{(3)}\right\} = \{1000\}$	1000	1	1000	1000	1000	0
3	{} No transactions	0	0	0	0	0	0

Table 5.1.3 Table of sample transaction values, and windowing method.

Let N_{T_c} represents the number of time intervals contained within the customers' entire time history $T_c - t_c^1$, where t_c^1 is the time of the initial transaction, and T_c is the timestamp of the final transaction of customer *c*.

$$N_{T_c} = \left\lfloor \frac{T_c - t_c^1}{\Delta t} \right\rfloor \qquad (5.1.3)$$

It is evident that the time interval $T_c - t_c^1$ is almost never an exact multiple of Δt, therefore we round the number of time intervals down to the nearest integer number. $N_{T_c} * \Delta t$ marks the final time window of the customer nearest to T_c that is less than Δt. Let Λ_c represent the sequence of all window functions, $\{\Phi_1, \dots, \Phi_6\}$, over all time windows pertaining customer *c*. The representation can be presented as follows,

$$\Lambda_c = \begin{bmatrix} \Phi_1(1) & \dots & \Phi_6(1) \\ \vdots & \dots & \vdots \\ \Phi_1(w_{t_c}) & \dots & \Phi_6(w_{t_c}) \end{bmatrix} \qquad (5.1.4)$$

Alternatively, Λ_c, can be denoted as a sequence of vectors $\Phi(w_t)$, belonging to customer c.

$$\Lambda_c = \{\Phi(1), \dots, \Phi(w_{t_c})\} \qquad (5.1.5)$$

Where each row of Λ_c represents the set of all window functions on transactions contained within a specific time window, where w_{t_c} marks the final time window in the customer's transaction sequence. The observations are in sequential ascending order from the first time window to the last time window, for customer *c*.

5.1.2 Time Windows for All Customers

For each customer in the training set, we compute Λ_c for both data sources, the *marketplace* and *virtual wallet*. We denote this as $\Lambda_c^{(HM)}$ and $\Lambda_c^{(HW)}$ respectively. Table 5.1.2 presents an sample of $\Lambda_c^{(HM)}$ for a single customer (the sequence is cut short, for purposes of brevity).

Timestamp	x_i	i
2016-04-21 9:29	100000	1
2016-04-30 18:11	30000	2
2016-05-02 7:32	171000	3
2016-05-02 7:44	98000	4

Table 5.1.4 Abridged sample of a single customer's time series transaction from marketplace.

Timestamp	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6	w_t
2016-04-21 9:00	100000	1	86	86	86	0	1
2016-04-21 21:00	0	0	0	0	0	0	2
2016-04-22 9:00	0	0	0	0	0	0	3
2016-04-22 21:00	0	0	0	0	0	0	4
2016-04-23 9:00	0	0	0	0	0	0	5
2016-04-23 21:00	0	0	0	0	0	0	6
2016-04-24 9:00	0	0	0	0	0	0	7
2016-04-24 21:00	0	0	0	0	0	0	8
2016-04-25 9:00	0	0	0	0	0	0	9
2016-04-25 21:00	0	0	0	0	0	0	10
2016-04-26 9:00	0	0	0	0	0	0	11
2016-04-26 21:00	0	0	0	0	0	0	12
2016-04-27 9:00	0	0	0	0	0	0	13
2016-04-27 21:00	0	0	0	0	0	0	14
2016-04-28 9:00	0	0	0	0	0	0	15
2016-04-28 21:00	0	0	0	0	0	0	16
2016-04-29 9:00	0	0	0	0	0	0	17
2016-04-29 21:00	0	0	0	0	0	0	18
2016-04-30 9:00	30000	1	2036	2036	2036	0	19
2016-04-30 21:00	0	0	0	0	0	0	20
2016-05-01 9:00	0	0	0	0	0	0	21
2016-05-01 21:00	269000	2	2703.5	4448	959	2467.1	22
2016-05-02 9:00	0	1	86	86	86	0	23
2016-05-02 21:00	100000	0	0	0	0	0	24

Table 5.1.5 Abbreviated Λ_c for w_t = 1 to 24 for a single customer.

The end result for each customer is a time window sequence consisting of equally spaced time windows that summarizes the transaction activity within the time windows, w_t. $\Phi_c(w_t)$ constitutes a multivariate observation that describes the spending behavior of the customer at each time window for a specific customer, c. The time window function is applied to all c customers to generate the observation sequence Λ_c.

5.2 Data Preprocessing

In order to construct a discrete time HMM, each of the multivariate numerical observations, $\Phi(w_t)$, in must be discretized into *observation symbols* referencing to the features outlined in Table 5.1.2. This is accomplished using a clustering algorithm. Referencing (Srivastava 2008), we apply the K-means algorithm. The objective of the K-Means algorithm is to classify observations into groups, where each of the observations within each group are significantly closer to one another compared to observations outside their respective group. We refer to this as a *neighborhood* or *cluster*. Assigning all time windowed features $\Phi(w_t)$ to a *cluster*, will allow us to discretize the numerical observations, $\Phi(w_t)$, values into discrete observation symbols, $O(w_t)$.

Notation	Definition
M_k	Specific centroid location.
K	Number of centroids.
$M_k[\Phi]$	The centroid assignment of observation Φ.
$\Phi(w_t)_c$	Window function observation at time window w_t for customer c.
Φ	An abbreviation for $\Phi(w_t)_c$, representing a single multivariate observation.
$\boldsymbol{\Phi}$	Complete set of observations, Φ, for all customers, at all time windows.
$\boldsymbol{\Phi'}$	Subset of $\boldsymbol{Y}$ containing only Φ where there was at least 1 transaction (non-dormant state).
$\boldsymbol{M_K}$	Set of all possible centroids, or centroid matrix.
$d(\Phi, M_k)$	Euclidean distance between observation Φ and centroid M_k.
μ_{ki}	Centroid mean component.
$\{\Phi, M_k[\Phi]\}$	The set of all observations Φ and its associated centroid assignment $M_k[\Phi]$.
$\overline{M_k[\Phi]}$	Centroid update after assignment step.
$O(w_t)_c$	Observation symbol (centroid) at time window *w* for customer *c*.
$\lvert M_k \rvert$	Magnitude of M_k. Measured as the number of observations belonging to M_k.
d	Dimensionality of the observation set, equaling 6 in our case.

Table 5.2.1 K-Means algorithm notation.

5.2.1 K-Means Algorithm

First described in (MacQueen, 1967) and (Hartigan, 1975), the K-Means algorithm presents a geometric interpretation of the classification problem. The algorithm classifies a set of observations into *K centroids* via an iterative algorithm. Each centroid has the same dimensionality of observation $\Phi(w_t)_c$, consisting of the 6 aggregation functions, outlined in Table 5.1.2, acting

on transactions contained in each time window. We use d to denote the dimensionality of the observation set, which is the total number of window functions. For our transaction data, $d = 6$.

$$\Phi(w_t)_c = \{\Phi_1(w_t) \quad \dots \quad \Phi_d(w_t)\}_c \tag{5.2.1}$$

We abbreviate $\Phi(w_t)_c$ as Φ for simplicity, because the K-Means algorithms does not consider for the time window, nor the customer that the time window belonged to. Each multidimensional centroid, M_k, can be expressed as,

$$M_k = [\mu_{k1} \dots \mu_{kd}] \tag{5.2.2}$$

The set of all possible centroids, or the centroid matrix, can be expressed as $\boldsymbol{M_K}$. Where $\boldsymbol{M_K}$ is a K x d matrix, indicating that there are K centroids each with a dimensionality of d.

$$\boldsymbol{M_K} = \begin{bmatrix} M_1 \\ \vdots \\ M_d \end{bmatrix} = \begin{bmatrix} \mu_{11} \dots \mu_{1d} \\ \vdots \\ \mu_{K1} \dots \mu_{Kd} \end{bmatrix} \tag{5.2.3}$$

Each row of the centroid matrix represents a centroid, M_K. Each column represents a dimension, corresponding a an aggregate function, Φ_d. Where an observation Φ can belong to any of K centroids $M_k \in \{M_1, M_2, \dots, M_K\}$, serving as a descriptive label for the frequency of spending, and amount of money spent within time window, w_t, for customer c, relative to the rest of the healthy customers on the platform. The objective of the K-Means algorithm is therefore to assign a spending group M_k to each observation Φ, such that a discrete variable is produced summarizing the monetary expenditure and frequency of transactions within time window w_t . The following section provides the specific procedure of the K-Means algorithm. We begin by defining $d(\Phi, M_k)$ as the Euclidean distance, where,

$$d(\Phi, M_k) = \sqrt{(\Phi_1 - \mu_{k1})^2 + (\Phi_2 - \mu_{k2})^2 \dots (\Phi_6 - \mu_{k6})^2} \tag{5.2.4}$$

We see that for K centroids, each observation, Φ, has a Euclidean distance from itself to M_k. Subsequently, we present the *assignment step* and *computation step* which work in conjunction to solve for fall observations and their assignments $\{\Phi, M_k[\Phi]\}$.

Assignment Step

The assignment step involves assigning Φ to the centroid, M_k, that minimizes the Euclidean distance $d(\Phi, M_k)$, by iteratively selecting M_k from K centroids. In other words, select M_k that minimizes $d(\Phi, M_k)$. This can be expressed by,

$$M_k[\Phi] = arg \min_{g \in K} d\big(\Phi, M_g\big) \qquad (5.2.5)$$

Where $M_k[\Phi]$ is the centroid assigned to observation Φ.

Computation Step

Provided the centroid assignments in the previous *assignment step*, the *computation step* computes a new set of centroids, based on the previous assignment. After each Φ is assigned to a group M_k, a new centroid, $\overline{M_k[\Phi]}$, is calculated, referred to as the centroid update. In the update, we re-compute new centroids based on the previous assigned observations $M_k[\Phi]$.

$$\overline{M_k[\Phi]} = \frac{1}{|M_k|} \sum_{\Phi \epsilon M_k} \Phi \qquad (5.2.6)$$

Where $\|M_k\|$ refers to the number of observations assigned to M_k. Equation (5.2.6) computes the new mean of M_k based on the previous assignment. We repeat the assignment and computation steps, until all of centroid updates, $\overline{M_k[\Phi]}$, no longer change from the current centroid $M_k[\Phi]$ when updated via Equation (5.2.6). Let $\boldsymbol{\Phi}$ denote the set of all observations at all time windows, for all customers. We subsequently summarize the K-Means algorithm as follows,

K Means Algorithm

For all Φ in $\boldsymbol{\Phi}$ with at least 1 transaction, denoted as Φ in $\boldsymbol{\Phi}'$:

1. Select K initial centroids via random initialization
2. **Repeat** the following steps until terminal condition (iii) is **True**:
   i. *Assignment Step*: Assign all Φ in $\boldsymbol{\Phi}'$ to its closest centroid, $M_k[\Phi]$.
   ii. *Computation Step*: Re-compute all centroids based on the previous assignment.
   iii. **If** M_k does not change for a all $\boldsymbol{M}$ since the previous iteration, **Then Return** all the set of all observations and its cluster assignment $\{\Phi, M_k[\Phi]\}$.

Table 5.2.2 K-Means algorithm illustration.

5.2.2 Illustrative Example

We present a brief illustrative example of the K-Means algorithm on simulated multivariate data in 2 dimensions, for K = 3. Figure 5.2.1 presents the graphical illustration of the K-Means algorithm. Centroids, denoted by the *stars* in the graphical illustration, are randomly initialized at the first iteration. All observations are assigned to a centroid, indicated by each observations colour, which corresponds to the centroid closest to each individual observation point on the 2D plane. In the *assignment step* a new centroid is computed via calculating the within cluster mean, provided the previous centroid assignment. We repeat the *assignment* and *computation* steps, until the computed centroids no longer change with each subsequent iteration, and we return the set of all observations and its cluster assignment $\{\Phi, M_k[\Phi]\}$, as indicated by the final iteration, Iteration No. 6.

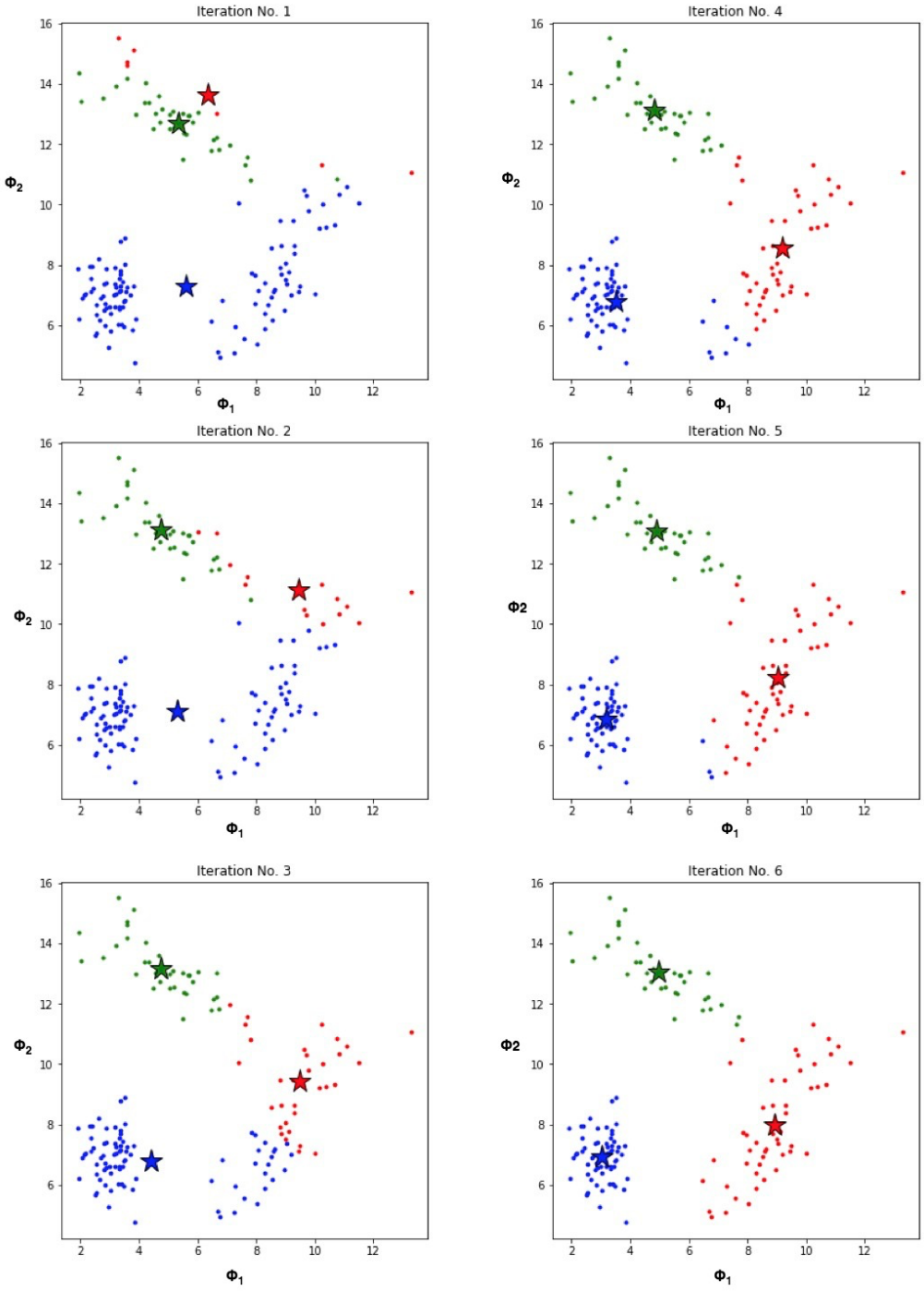

Figure 5.2.1 Graphical illustration of the K-Means algorithm in 2D for K = 3.

Iteration No. 1	μ_{k1}	μ_{k2}

M_1 (Blue)	5.59044019	7.29678711
M_2 (Red)	6.34239707	13.62655217
M_3 (Green)	5.32785306	12.70582335
Iteration No. 2		
M_1 (Blue)	5.30196805	7.12030318
M_2 (Red)	9.45231665	11.14388061
M_3 (Green)	4.75017655	13.11345402
Iteration No. 3		
M_1 (Blue)	4.39464	6.80742905
M_2 (Red)	9.48446621	9.4369295
M_3 (Green)	4.75374681	13.17555242
Iteration No. 4		
M_1 (Blue)	3.50221811	6.77985561
M_2 (Red)	9.19561289	8.56522311
M_3 (Green)	4.81917091	13.14200682
Iteration No. 5		
M_1 (Blue)	3.02152016	6.93357839
M_2 (Red)	8.93325765	8.00186898
M_3 (Green)	4.96890973	13.05344209
Iteration No. 6		
M_1 (Blue)	3.02152016	6.93357839
M_2 (Red)	8.93325765	8.00186898
M_3 (Green)	4.96890973	13.05344209

Table 5.2.3 Centroid outputs at each iteration of the K-Means algorithm on simulated data.

5.2.3 Modified K-Means Algorithm

We perform the K-means algorithm on each customer's time window, consisting of d = 6 features. Each observation, independent of time and the customer it belonged to, is denoted by Φ, in both the *marketplace* and *virtual wallet* domains, where there has been at least one transaction made. For our application, the K-Means algorithm is modified slightly to incorporate a *no spend* group, where it categorizes Φ as a "*no spend"* if no transactions appear for that time window. This is based on knowledge that if a customer does not spend during a time window they are in a *dormant state*, which is drastically different from a *low spend* state. The modified K-Means algorithm effectively discretizes the observed variables. Provided the centroid specifications in Table 5.2.4 and Table 5.2.5, only time windows which contain at least one transaction are classified into one of the non-dormant spending groups for both *marketplace* and *virtual wallet*. Effectively, the modified K-Means algorithm generates list of observation symbols for all time windows belonging

to all customers. We refer to the centroid, M_k , assigned to *observation* $\Phi(w_t)_c$, as $M_k[\Phi(w_t)_c]$, which we will also denote as the *observation symbol* $O(w_t)_c$ for future reference. Therefore for customer c, the observation symbol, $O(w_t)_c$, can be defined as,

$$O(w_t)_c := M_k[\Phi(w_t)_c] \in \{1,2,3,0\} \qquad (5.2.7)$$

Evidently, the K-Means algorithm discretizes the numerical observations of $\Phi(w_t)_c$ into a single categorical value $O(w_t)_c$ constituting the observation symbols of the HMM model we will construct subsequently in this Chapter.

5.2.4 K-Means Algorithm on Full Dataset

We present $\boldsymbol{M_K}$ for both *marketplace* and *virtual wallet* sources for all healthy customers, where K = 3 centroids, with an additional no spend group [0]. $\boldsymbol{M_K}$ was fitted using a the modified K-Means algorithm outlined in Section 5.2.3 for all 5050 customers in the training data and their respective transaction time windows w_t.

k	Type	μ_{k1}	μ_{k2}	μ_{k3}	μ_{k4}	μ_{k5}	μ_{k6}
[1]	high	129082.6	3.64	44843.2	52050.1	34789.4	8781.9
[2]	low	472.86	1.9	255.69	24.03	18.15	3.72
[3]	med	31914.4	3.4	17577.4	307.33	213.42	52.80
[0]	no spend*	0	0	0	0	0	0

Table 5.2.4 Centroid assignments for K-Means algorithm – Marketplace transactions.

k	Type	μ_{k1}	μ_{k2}	μ_{k3}	μ_{k4}	μ_{k5}	μ_{k6}
[1]	high	26422.44	4.7	7635.21	11751.35	4884.93	3243.18
[2]	low	300.49	2.0	145.47	191.63	106.64	47.64
[3]	med	6560.81	4.1	2414.6	3444.33	2467.80	988.5
[0]	no spend*	0	0	0	0	0	0

Table 5.2.5 Centroid assignments for K-Means algorithm – Wallet transactions.

Using these cluster assignments we classify each customer's transaction window into the four distinct observation symbols. High [1], medium [3], low [2], and no spend [0]. Where the *no spend* spending level is a heuristic condition that indicates no transaction has been made in the time

window. We reference (Srivastava, et al., 2008) for the selection of $K = 3$ spending groups, which essentially divides the customers into 3 different spending profiles (high, medium, and low), with an additional *no spend* group[7]. In summary, the preprocessing phase involves aggregating transactions to transform them into time windows with multivariate features, $\Phi(w_t)_c$, and subsequently applying the K-Means algorithm to assign each time window an *observation symbol*, $O(w_t)_c$, for each time window w_t, for customer c.

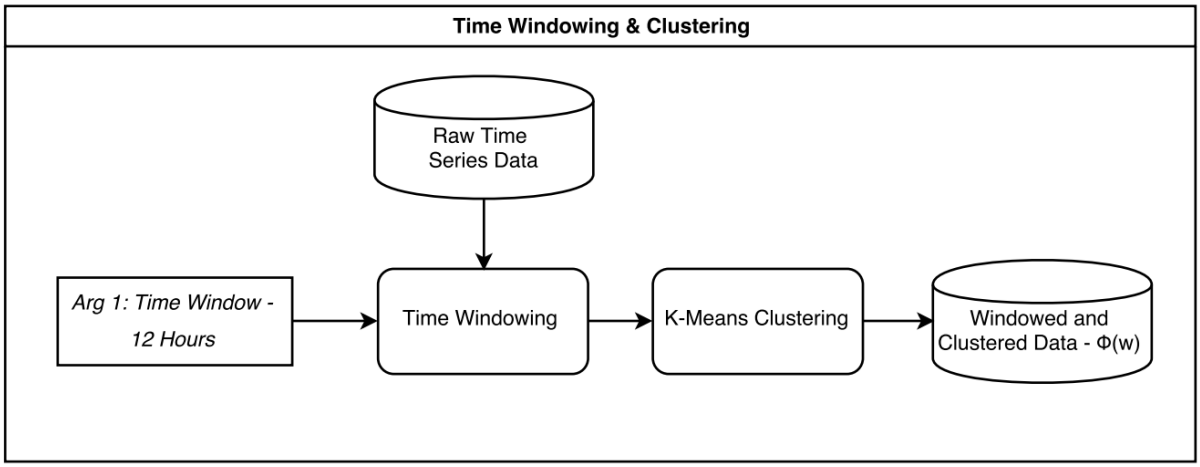

Figure 5.2.2 Data preprocessing procedure.

5.3 HMM Modeling of Customer Spending Behaviour

Based on the concept of stochastic modeling, we present an innovative approach that considers the sequential information contained in a time series process. An HMM can be constructed for each customer separately, implying that each customer must make a series of transactions before an HMM can be constructed. This per-customer HMM is demonstrated in (Srivastava, et al., 2008), and was shown to be effective in detecting cases of financial fraud. In our implementation, we differ from the former model by constructing a general HMM model for all customers in the training set. This allows fraud detection to begin immediately for each customer in the testing set,

[7] With this addition our algorithm does not specifically follow the traditional K-means algorithm, but adheres to a modified version. This change is implemented because we have prior knowledge that the no spending state is significantly different from a state where spending has occurred.

eliminating the cold-start problem that a *per-customer* based HMM would imply. Furthermore, a general model is inherently descriptive of the healthy customer as a whole and is capable of forecasting healthy transactions for any future healthy customers.

We construct an HMM classifier from the transaction histories of healthy customers only, provided the observation symbols outlined Section 5.2,. The classification algorithm considers the probability of observing the sequence, based on the model constructed from healthy customer's transactions. Using the same splitting mechanism introduced in Chapter 2, we divide the training data according to a 75% - 25% split. 75% of the data is randomly sampled without replacement from the complete data set to form the *training set*. Only the healthy customers in the training set are used to construct the HMM representing the behaviour of healthy customers. We refer to this model as the *reference model*. The remaining 25% of the data is denoted as the *test set*, and is used for holdout validation. The test set contains the transaction histories of both healthy and fraudulent customers which are mutually exclusive of the training set. The objective of the classifier to distinguish between the two types of customers in the test set.

Notation	Definition
w_t	Time window w_t.
W_c	Final time window of customer c.

$O(w_t)$	Observation symbol (centroid) at time window w_t for any single customer.
$O(w_t)_c$	Observation symbol (centroid) at time window w_t specific to customer c.
θ_H	HMM parameters (A, B, π) reference model, pertaining to either marketplace or virtual wallet.
θ_{HM}	Marketplace HMM model parameters.
θ_{HW}	Virtual wallet HMM model parameters.
$\pi[\theta_{HM}]$	HMM initial state probability distribution for marketplace.
$\pi[\theta_{HW}]$	HMM initial state probability distribution for virtual wallet.
$O_{1:W_c}^{(c)}$	Full observation sequence of customer c, from the first time window until the final time window W_c.
$A[\theta_{HM}]$	General HMM transition probability matrix for marketplace.
$B[\theta_{HM}]$	General HMM emission probability matrix for marketplace.
$A[\theta_{HW}]$	General HMM transition probability matrix for virtual wallet.
$B[\theta_{HW}]$	General HMM emission probability matrix for virtual wallet.
$O_{1:w_t}^{(c)}$	Observation sequence of customer c, up until time window w_t.
$\Omega_{w_t}^{(c)}$	Probability of observing $O_{1:w_t}^{(c)}$ provided θ_H, calculated via the forward algorithm.
η	Transaction probability ratio threshold indicating the limit for $\eta_{w_t}^{(c)}$. If $\eta_{w_t}^{(c)} > \eta$ then the observation sequence is deemed anomalous.
$\eta_{w_t}^{(c)}$	Transaction probability ratio for customer c at time window w_t.
η_{HM}	η threshold for marketplace model.
η_{HW}	η threshold for virtual wallet model.

Table 5.3.1 HMM construction for classification notation.

5.3.1 HMM Parameters

In our HMM, we specify three distinct hidden states[8], where each hidden state will exhibit a different set of emission probabilities for $O(w_t)_c$. The initial state probabilities for both marketplace $\pi[\theta_{HM}]$, and virtual wallet $\pi[\theta_{HW}]$, at time $w = 1$, will be uniform for 3 hidden states.

$$\pi[\theta_{HM}] = \pi[\theta_{HW}] = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \qquad (5.3.1)$$

[8] Though there could be more hidden states selected, we confine the scope of our research to the 3 hidden state specification.

Applying the Baum-Welch algorithm on the observation sequence of centroids, we construct an HMM to model the transaction behaviour of healthy customers. Three hidden states are defined as, $\{1, 2, 3\}$, which transit between one another, based on the transition probability parameters defined by $A[\theta_{HM}]$, for marketplace transactions only. Each healthy customer in the training set will provide a sequence of transactions beginning with initial state probability $\pi[\theta_{HM}]$ or $\pi[\theta_{HW}]$, depending on whether these transactions belong to the marketplace or virtual wallet respectively. Each customer's complete observation symbols sequence, $O_{1:W_c}^{(c)}$, are then fitted via the Baum-Welch algorithm for multiple sequences via the method presented earlier in Chapter 4. After fitting the multiple sequence HMM on all $O_{1:W_c}^{(c)}$ in the training set for *marketplace*, we obtain the transition matrix $A[\theta_{HM}]$.

$$A[\theta_{HM}] = \begin{bmatrix} 0.3425 & 0.5992 & 0.0582 \\ 0.3427 & 0.6027 & 0.0545 \\ 0.2775 & 0.4356 & 0.2869 \end{bmatrix} \quad (5.3.2)$$

We present the emission probabilities of the HMM as the probability of belonging to each of the 4 observation symbols $\{0, 1, 2, 3\}$, provided the hidden state $\{1, 2, 3\}$. This constitutes the emission matrix of the HMM, $B[\theta_{HM}]$, for marketplace transactions only.

$$B[\theta_{HM}] = \begin{bmatrix} 0.9445 & 3.182 * 10^{-3} & 6.484 * 10^{-3} & 4.578 * 10^{-2} \\ 0.9727 & 1.398 * 10^{-4} & 1.523 * 10^{-2} & 1.189 * 10^{-2} \\ 0.4325 & 6.612 * 10^{-2} & 0.2124 & 0.2889 \end{bmatrix} \quad (5.3.3)$$

Similarly, the transition and emission probability matrices of all observation sequences $O_{1:W_c}^{(c)}$ on the virtual wallet are calculated via the Baum-Welch Algorithm as well, which we refer to as $A[\theta_{HW}]$ and $B[\theta_{HW}]$ respectively.

$$A[\theta_{HW}] = \begin{bmatrix} 0.3481 & 0.5882 & 0.0636 \\ 0.3465 & 0.6012 & 0.0522 \\ 0.3079 & 0.4065 & 0.2855 \end{bmatrix} \quad (5.3.4)$$

$$B[\theta_{HW}] = \begin{bmatrix} 0.9256 & 2.823 * 10^{-2} & 7.365 * 10^{-4} & 4.536 * 10^{-2} \\ 0.9851 & 1.281 * 10^{-3} & 1.522 * 10^{-3} & 1.205 * 10^{-2} \\ 0.4293 & 2.203 * 10^{-1} & 6.584 * 10^{-2} & 2.845 * 10^{-1} \end{bmatrix} \quad (5.3.5)$$

5.3.2 Sequence Acceptance Classification Algorithm

As previously outlined, an HMM is fitted onto the observation symbols belonging to only healthy customers in the training set via the Baum-Welch Algorithm, where the hidden states are characteristic of the spending behaviour of the customers at that specific instance in time. We refer to the parameters of this healthy model as θ_H (where θ_H can pertain any HMM model, either marketplace or virtual wallet). For each customer *c*, in the testing set, we calculate the probability of observing the sequence $O_{1:w_t}^{(c)}$ using forward algorithm with HMM parameters θ_H. We refer to this probability as $\Omega_w^{(c)}$, for customer c.

$$\Omega_{w_t}^{(c)} = P(O_{1:w_t}|\theta_H) \quad (5.3.6)$$

We classify the transaction history of all customers, to construct a spending profile of healthy customers, where Ω_{w_t} refers to the probability of observing $O_{1:w_t}^{(c)}$ provided parameters θ_H computed using the forward algorithm. We compute $\Omega_{w_t}^{(c)}$ every time w_t increments, thus we must account for the change in $\Omega_{w_t}^{(c)}$ at each progressive time window. The difference between each of the calculated probabilities is measured as,

$$\Delta\Omega_{w_t}^{(c)} = \left|\Omega_{w_t}^{(c)} - \Omega_{w_t-1}^{(c)}\right|, \qquad for\ w_t > 0 \quad (5.3.7)$$

The *transaction probability ratio* for customer *c* is defined as $\eta_{w_t}^{(c)}$, where,

$$\eta_{w_t}^{(c)} = \frac{\Delta\Omega_{w_t}^{(c)}}{\Omega_{w_t}^{(c)}} \quad (5.3.8)$$

$\eta_{w_t}^{(c)}$ represents the change in probability when the additional observation $O_{1:w_t}^{(c)}$ is observed. This is represented by $\Delta\Omega_{w_t}$ divided by Ω_{w_t-1}. Subsequently we define a threshold, η, where the stopping condition applies at time window w_t if,

$$\eta_{w_t}^{(c)} > \eta \qquad (5.3.9)$$

Where η is a fixed threshold that determines the threshold where a positive anomaly is detected. If $\eta_{w_t}^{(c)} > \eta$, then the algorithm will signal that the spending behaviour of customer *c* deviates from that of a healthy customer's spending behavior, constituting a fraud signal.

5.4 Fraud Prevention Procedure and Detection Algorithm

Notation	Definition
$\Phi_{1:w_t}$	Observed sequence Φ from time 1 to w_t.
θ_{LR}	Logistic regression model parameters, for customer registration features.
$O_{w_t}^{(c)}$	Single observation symbol for customer c, at time window w_t
$P_f^{(c)}(\theta_{LR})$	Probability (of fraud) value produced by model θ_{LR}, on the registration features pertaining to customer *c*.
$P_f'(\theta_{LR})$	Probability threshold for θ_{LR}, where the θ_{LR} will produce a positive fraud signal (1) if $P_f^{(c)}(\theta_{LR}) > P_f'(\theta_{LR})$.
$\boldsymbol{W^{(c)}}$	The set of all time windows pertaining to customer c.
$\boldsymbol{O^{(c)}}$	Full observation symbol sequence pertaining to customer c.
$I_{LR}^{(c)}$	Indicator function for indicating if θ_{LR} if produced a positive fraud signal
$I_{HM}^{(c)}$	Indicator function for indicating if θ_{HM} if produced a positive fraud signal
$I_{HW}^{(c)}$	Indicator function for indicating if θ_{HW} if produced a positive fraud signal
$\mathbb{Z}(w_t)_c$	Zeta algorithm output at time window w_t customer *c*.
$\mathbb{Z}$	Zeta algorithm for fraud detection.

Table 5.4.1 Fraud detection algorithm notation.

In this section, we outline an ensemble method that combines the output of two HMM's and a logistic regression model on customer registration features, which we refer to as *LR1*. *LR1* serves as a baseline for fraud detection. As *LR1* does not entail the use of time windowed features, this allows *LR1* to provide a fraud detection signal before the customer makes any transactions, as it

relies solely on feature provided upon customer registration. For the same reason, however, *LR1* entails a high false positive rate due to the fact that registration features alone cannot be entirely informative of a customer's intentions on an eCommerce platform. In order to reduce the false positive rate, we combine *LR1* with HMM modelling on time windowed features increasing the overall detection rate of a model. Our research aims to provide an algorithm that significantly enhances the fraud detection rate on any general platform in the eCommerce industry (which can also extend to any industry that exhibits a high volume of financial transactions). We also seek to provide a means to perform fraud detection on a continuous online basis, and to provide further in-depth research into fraud detection simulation where transactions are arriving in real-time, considering the dynamic nature of customers.

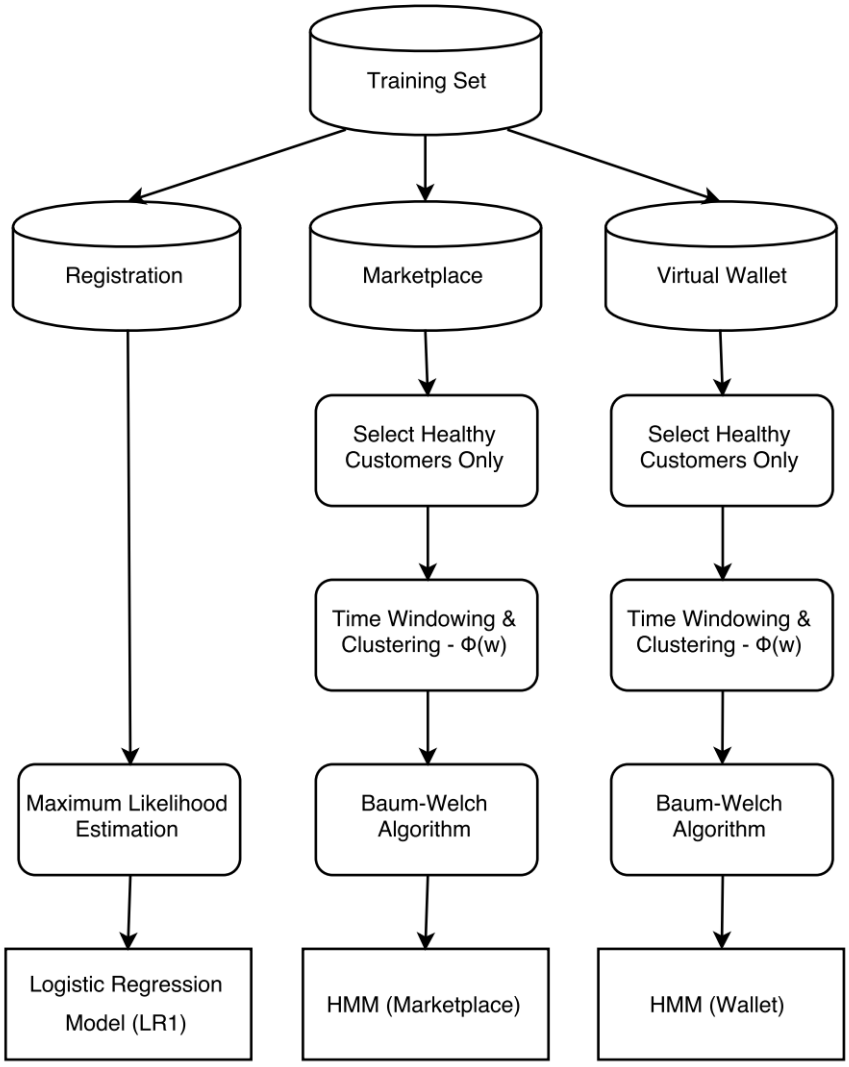

Table 5.4.2 Customer behaviour model construction pipeline.

5.4.1 Automated Fraud Detection Algorithm

The ensemble model which combines HMM modelling with logistic regression is denoted as the $\mathbb{Z}$ algorithm. We opt to apply two HMM's to the $\mathbb{Z}$ algorithm because a single HMM is prone to generate a high amount of false positives. This is due to the fact that, for a single HMM, the amount of transactions from time window to adjacent time window is far greater than the number of customers. At any of these time window transactions, a potential fraud signal can be generated

raising the probability of obtaining false positives. To mitigate this, we combine two HMM's to reduce the False Positive Rate of a single HMM. As one signal from an HMM is not enough to identify that the customer is fraudulent, a combination of signals must be sufficient. To further mitigate this issue, we combine the result of a logistic regression model, adding information about the state of each customer upon registration, features which are not time dependent. The combination of the three models allows us to retain a low FPR while attaining a higher fraud detection rate.

At each 12-hour time window each customer is evaluated based on the output constructed from the three models, whose parameters are represented by θ_{HM} , θ_{HW} and θ_{LR}. Where θ_{HM} are the parameters of the HMM that model customer behaviour on the *marketplace*, θ_{HM} are the parameters of the HMM that models customer behvaiour on the *virtual wallet*, and θ_{LR} are the parameters of *LR1*, the logistic regression model that models customer registration features. The prediction is a combination of the signals of all three models, which we denote as the $\mathbb{Z}$ Algorithm, at time window w_t.

$\mathbb{Z}$ Algorithm for Ensemble Modeling of Customer Behaviour

---

Require HMM Model Parameters of Marketplace transactions θ_{HM}.

Require HMM Model Parameters of Virtual Wallet transactions θ_{HW}.

Require Logistic Regression Model Parameters on Registration Features θ_{LR}.

For each customer, c in *test set* $\boldsymbol{C}$:

Perform Logistic Regression with Parameters θ_{LR} to produce $P_f^{(c)}(\theta_{LR})$.

I. Calculate $I_{LR}^{(c)} = \mathbf{1}[P_f^{(c)}(\theta_{LR}) > P_f'(\theta_{LR})]$, where $\mathbf{1}$ is an indicator function.

II. For each time window, w_t, in subset of $\boldsymbol{W}^{(\boldsymbol{c})}$:

1. Append the future observation, $O_{w_t+1}^{(c)}$, to the current time series sequence, $O_{1:w_t}^{(c)}$.
2. Calculate $\eta_{w_t}^{(c)}(\theta_{HM})$ for $O_{1:w_t}^{(c)}$, for incoming marketplace observation using θ_{HM}.
3. Calculate $\eta_{w_t}^{(c)}(\theta_{HW})$ for $O_{1:w_t}^{(c)}$, for incoming wallet observation, using θ_{HW}.
4. Evaluate $[I_{HM}^{(c)}, I_{HW}^{(c)}]$, where $\mathbf{1}$ is an indicator function defined as,

i. $I_{HM}^{(c)} = \mathbf{1}[\eta_w^{(c)}(\theta_{HM}) > \eta_{HM}]$

ii. $I_{HW}^{(c)} = \mathbf{1}[\eta_w^{(c)}(\theta_{HW}) > \eta_{HW}]$

5. Calculate $\mathbb{Z}(w_t)_c = I_{LR}^{(c)} + I_{HM}^{(c)} + I_{HW}^{(c)}$
6. **If** $\mathbb{Z}(w_t)_c > 1$ **Then Exit** and **Return** 1.
7. **Else If** $O_{w_t+1}^{(c)}$ is the last observation in $\boldsymbol{W^{(c)}}$, **Then Exit** and **Return** 0.

Table 5.4.3 $\mathbb{Z}(t)$ Algorithm outline.

The $\mathbb{Z}$ Algorithm evaluates the time windowed features of each time window of each customer as illustrated in Figure 5.2.2. Subsequently, the algorithm builds an HMM on top of the discrete time series data belonging to both the marketplace and the virtual wallet. In addition, it constructs a logistic regression model that is trained on customer registration features, which are not time series dependent, and exist before any transactions are made. Provided the output of the three models, the $\mathbb{Z}$ Algorithm stipulates a logical condition, enforcing that there must be at least two models out of the three that produce a positive anomaly signal before the customer, at time window *w*, is deemed to be anomalous. In which case, the algorithm will return a true value of 1, otherwise it will return 0 at the end of observation symbol sequence, $\boldsymbol{O^{(c)}}$, indicating that no anomalous behavior was detected for customer *c*.

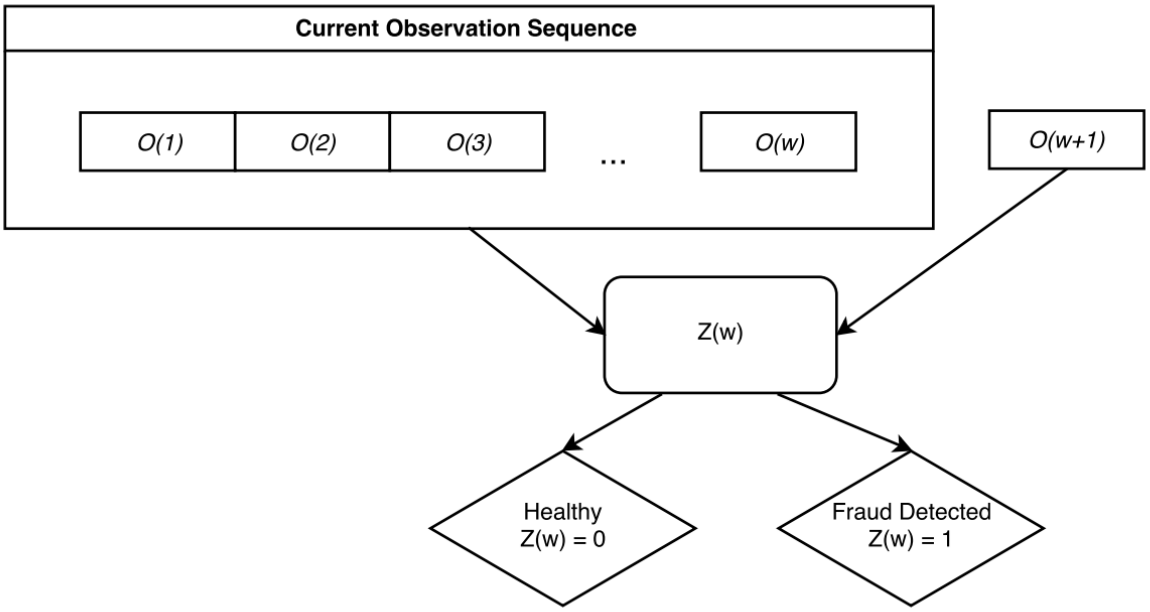

Figure 5.4.1 $\mathbb{Z}(t)$ Algorithm functional diagram.

5.4.2 Fraud Detection Results

We provide a *rule-of-thumb* where the FPR should remain below 0.05, we refer to this are our *constraint*. This constraint on the FPR is strictly enforced on the $\mathbb{Z}$ Algorithm (however we will see later that not all classification methods explored in this thesis can yield such a low FPR value, therefore we relax this constraint for classification algorithms that cannot achieve this). Simultaneously, we seek to maximize the detection rate, we refer to this as the *objective function*. The process of finding the probability thresholds $(P_f', \eta_{HM}, \eta_{HW})$ that both maximizes the *objective function*, and meets our *constraint* is referred to as *hyperparameter optimization*, where the probability thresholds of each model are referred to as a *hyperparameters*.

Since the search space is not unfeasibly large, we manually adjust the *hyperparameters* $(P_f', \eta_{HM}, \eta_{Hw})$ in a randomized manner to maximize the objective function while ensuring that the constraint is met. We refer to this as a randomized search, where not all possibilities of *hyperparameters* are explored, but we complete the search once one set of *hyperparameters* meets

the constraint. By conducting this straight forward search method[9] we determine a set of probability threshold levels, where we can calculate the detection rate (TPR). This implies that although the optimization constraints are met, there may possibly exist be arrangements of *hyperparameters* that can potentially yield lower FPR's, with higher TPR's. However, our selection of probability thresholds values are already sufficient to outperform any of the previously selected classification algorithms, as evidenced in Figure 5.4.3.

Threshold	Value
$P_f{}'$	0.0027143
η_{HM}	0.98107
η_{HW}	0.99835

Table 5.4.4 Threshold values (hyperparameters) for Zeta algorithm.

Model Performance	Count
TP (True Positives)	100
TN (True Negatives)	2072
FP (False Positives)	1
FN (False Negatives)	108
FPR (False Positive Rate)	0.049541
TPR (True Positive Rate)	0.99010

Figure 5.4.2 Decision tree model performance parameters.

The $\mathbb{Z}$ *algorithm*, combining both marketplace and virtual wallet features in addition to customer registration features, obtains excellent classification results superior than that of any *aggregate feature classification* method previously investigated in Chapter 3. This is due to the fact that the $\mathbb{Z}$ Algorithm considers additional information concerning behavioural changes of each customer. The $\mathbb{Z}$ Algorithm considers the acceptance probability of observations symbols related to customer spending as they sequentially move forward in time, whereas an *aggregate feature classification*

[9] However, there exist iterative parameter such as the *Nelder-Mead Simplicial Heuristic* and *Particle Swarm Optimization* (to name a few), that can be used to for the task of hyperparameter optimization. These methods are beyond the scope of this research.

method considers only the final summary of features and performs a traditional classification technique to distinguish between fraudulent and healthy customers.

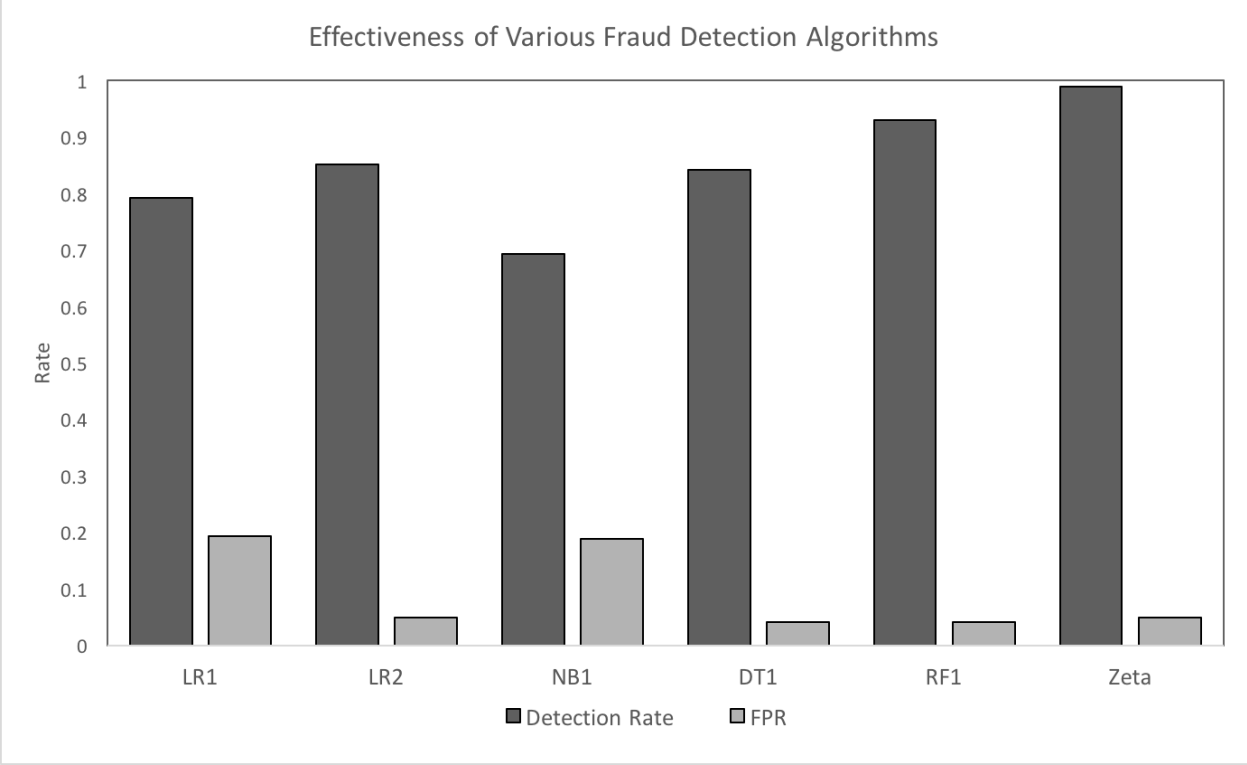

Figure 5.4.3 Comparison of various fraud detection algorithms by detection rate and false positive rate.

Model Identifier	Model Name
LR1	*Logistic Regression on Customer Registration Features*
LR2	*Logistic Regression on Select Registration and Time Series Features*
NB1	*Naïve Bayesian Classifier on Customer Registration Features*
DT1	*Decision Tree on All Available Features*
RF1	*Random Forest on All Available Features*
Zeta	*ℤ Algorithm*

Table 5.4.5 Model identifier dictionary.

Because of the ℤ Algorithm's dynamic classification approach which updates observation symbols moving forward in time, it is able to detect signals that are descriptive of a customer's change in

spending behaviour. This is allows the platform to block fraudulent customers preventively in order to mitigate the financial damage that can occur if such customers allowed to continue making transactions on the system. This approach is not applicable in the *aggregate feature classification* because it requires that a significant amount of transactions accrue, 6-months in our case, before customer classification can occur. In the aggregate feature classification methods, even if the fraudulent customer is accurately identified, a significant time will have passed before a fraudulent customer can be blocked. Therefore, studies that rely on the aggregate feature classification may not be ideal for real-time industry applications in the field of fraud detection.

Ultimately, our HMM detects fraud by examining the difference between the HMM forward probability with each additional observation symbol, denoted by $\Delta\Omega_{w_{t+1}}^{(c)} = \left|\Omega_{w_{t+1}}^{(c)} - \Omega_{w_t}^{(c)}\right|$. However, theoretically under a scenario where the customer starts off in a fraudulent fashion, the additional observational symbol, $\Omega_{w_{t+1}}^{(c)}$, may not generate generate enough of a difference in forward probabilities to trigger a fraudulent signal. In our experiment, we have not found this issue to be a major factor affecting the classification accuracy of our model. However, we propose that if this issue does affect the accruacy of the model we can simply apply an threshold for the initial forward probability, $\Omega_1^{(c)}$, in addition to a threshold for the change in forward probability of the observation sequence with each incrementing time window, $\Delta\Omega_{w_t}^{(c)}$.

5.4.3 Online Fraud Detection Algorithm

For the purpose of online fraud detection, we apply the $\mathbb{Z}$ Algorithm in conjunction with human validation to ensure that healthy customers are not incorrectly blocked from the system. Upon the detection of an anomalous signal, $\mathbb{Z}(w_t)_c = 1$ at time window w_t, the customer is relayed over to a team of human examiners to confirm that the customer is fraudulent. If the examiners find that the customer has been falsely identified, then that customer will be reinstated back into the system as a healthy customer. Because the $\mathbb{Z}$ algorithm produces a false positive rate of just under 5%, we

expect that 5% of all customers flagged by the ℤ algorithm will be falsely classified. This is acceptable because the more critical matter is to ensure that all cases of fraudulent activity do not go undetected, thus we can sacrifice some false positive cases to achieve this end.

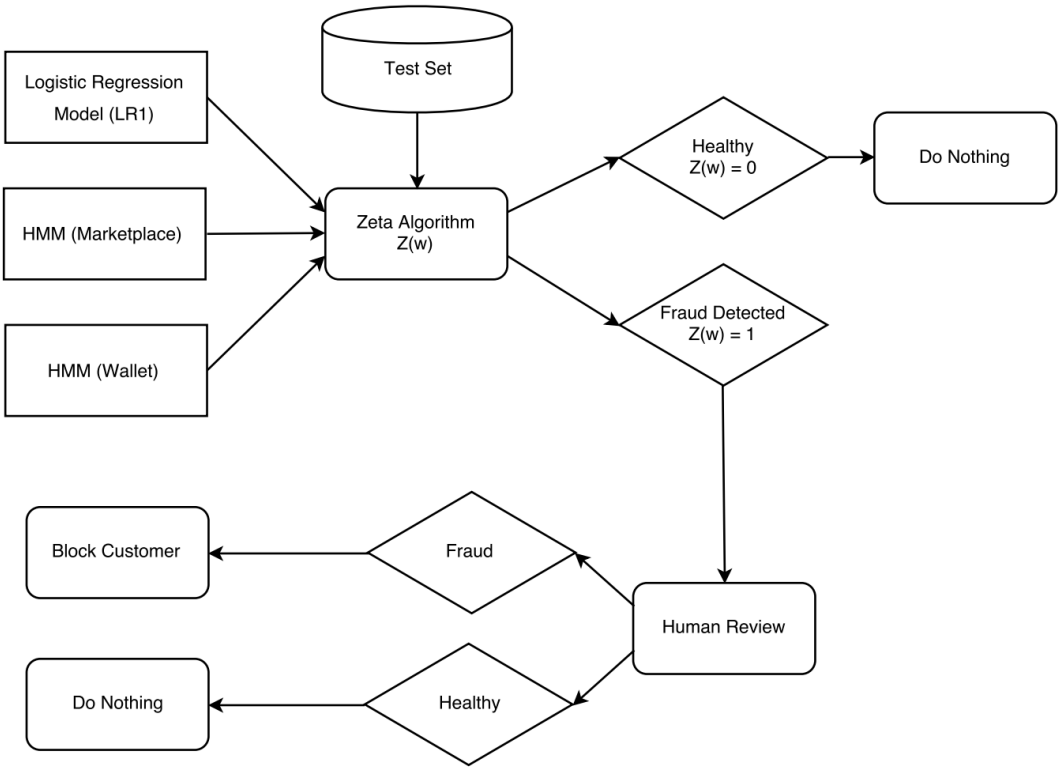

Figure 5.4.4 Online fraud detection and prevention policy.

As evidenced in Section 5.4.2, the ℤ algorithm is capable of detecting almost all fraudulent customers, capable of replicating the detection capabilities human examination. Due to the ever evolving nature of fraud, the need for a human review to detect fraud manually will always exist and there should be no impetus to entirely eliminate the usage of human validation. As reflected in Chapter 2, only a small percentage (5%) of customers are marked as fraudulent. In reality, the percentage of fraudulent customers on an eCommerce network is significantly less than 5% (this is due to the fact that more fraudulent cases were injected into the sample data set for analysis purposes). The ℤ Algorithm will immensely reduce the amount of human effort required to identify fraudulent behviour, as it will filter out the healthy customers that do not need human verification.

The $\mathbb{Z}$ Algorithm is able to identify common trends that are characteristic of fraud, reducing the amount of repetitive work performed by human examiners. To clarify, the $\mathbb{Z}$ algorithm is not capable of detecting *new* methods of fraud, therefore, a human review team will always be required to actively label such instances. Therefore, the goal of the $\mathbb{Z}$ Algorithm is not to replace human examination entirely, but to substantially reduce the amount of repetitive effort required to recognize the pre-identified instances of fraud as labeled by human examiners.

Fraud Detection Procedure:
For each customer *c*, in the set of all customers: For each 12-Hour time window, w_t: 1. Receive all customers where $\mathbb{Z}(w_t)_c = 1$ (potential fraud detected). 2. Perform manual evaluation on that specific customer's transaction history to confirm or deny whether or not the customer is fraudulent. a. If the customer is fraudulent, record the fraudulent behavior, and block the customer from making further transactions. b. If the customer is not fraudulent, continue to allow the transactions of that customer without interference.

Table 5.4.6 Online fraud detection procedure.

The *fraud detection procedure* outlined in this section provides a guideline for human examiners on the application of the $\mathbb{Z}$ Algorithm to reduce the amount of manual evaluation required to maintain a safe and profitable eCommerce platform. The $\mathbb{Z}$ Algorithm provides a means for general eCommerce platforms to efficiently and effectively detect fraud without sacrificing detection rate, while attaining a high level of automation. The $\mathbb{Z}$ Algorithm serves as a form of machine intelligence that is able to distinguish between healthy customers, and fraudulent customers to an accurate degree. Subsequently, fraud detection specialists will be able to focus more of their effort to discovering new instances of fraud, as opposed to reviewing to confirm previously identified patterns of fraud. Once new patterns of fraud are demarcated, model training can be run to update the parameters of the model, as demonstrated in Section 5.3.1. We recommend

the model parameter update to be completed at a lower frequency than the 12-hour window. However, if the computing power is available, model parameters, θ_{HM} , θ_{HW} and θ_{LR}, can be updated as soon as possible. This is up to the discretion of the industry's preference, and the computing resources that are available.

Chapter 6. Conclusion

6.1 Summary of Contribution

This thesis provides two areas of significant improvement in the field of statistical fraud detection, specifically in the quickly advancing industry of eCommerce. We provide a comparative study between various forms of fraud classification methods, and examine the classification performance of each model provided our transaction dataset. Four traditional classification models were tested Logistic Regression, Naïve Bayesian Classifier, Decision Tree, and Random Forest. We discovered that the Random Forest model produces the best classification results of the traditional models, whereas Naïve Bayesian classifier produces the poorest results. The classification power of the Random Forest model can be attributed to its robust parameter estimation procedure which involves extensive resampling on the training data. And the poor performance of the Naïve Bayesian Classifier may be due to the fact that it assumes the probabilistic independence among all features, when in fact this may not always be the case.

In the second part of the thesis, we develop the $\mathbb{Z}$ ensemble algorithm, which combines Hidden Markov Modelling and Logistic Regression. We find that our algorithm performs very well on the provided data set. We limit the false positive rate of the algorithm to 5%, while maximizing the detection rate of the algorithm. With a relatively low FPR, virtually all fraudulent cases are detected. This provides evidence that the $\mathbb{Z}$ algorithm is a good substitute for human judgement concerning fraudulent transactions. Implementing this algorithm, we will allow our industry partner to operate more efficiently by reducing the amount of manual labour required to inspect all transaction histories via human examination, while maintaining a high-level of fraud detection accuracy.

The classification results yielded by the $\mathbb{Z}$ algorithm are highly effective at detecting fraud. The implementation of this fraud detection algorithm will allow our industry client to perform efficient and accurate fraud detection at a much higher scale than previously accomplishable on eCommerce networks. The $\mathbb{Z}$ algorithm is capable of replacing a significant amount of human detection effort. A smaller number of human examiners will be required to examine the customers who have been

classified as anomalous to verify the detections and subsequently terminate them from the platform's services, or re-instate the customer if the algorithm made a false detection. This enhancement on the efficiency of the fraud detection process will improve the profitability of the organization while maintaining a high degree of customer trust due to the retained anti-fraud validity.

6.2 Future improvements

It is possible to improve this study in a multitude of ways. We focus on three areas which can be open to further investigation. These areas were not investigated during the development of this thesis due to the fact that we have already met the objective of devising a new algorithm that is able to classify fraudulent customers with a high detection rate, while maintaining a low FPR. Investigation of the more advanced methods is significantly different from the modeling approaches outlined in this thesis. However, these methods are also well-documented in literature, and should be investigated as future work in order to further understand the behaviour of customers, and achieve even better classification results.

6.2.1 Model Parameter Adjustments

In this project, we have chosen to select 3 hidden states, and 4 observation symbols that constitute the parameters which dictate structure of the Hidden Markov Model. Though it does provide excellent classification accuracy, it may be worthwhile to investigate varying selections of the number of hidden states and observation symbols, subsequently evaluating the effect that selection on the classification accuracy. Another evident area of improvement would be a detailed study of the optimal number of centroids to select when performing K-means clustering and its effect on classification accuracy. Furthermore, we suggest an open investigation into more advanced clustering mechanisms. An example of more advanced clustering techniques is DB Scan (Kriegel, 1998), which is capable of detecting non-linearly separable clusters. An investigation into these techniques may improve the robustness of the clustering mechanism.

6.2.2 Dimensionality Reduction

It would be worthwhile to explore dimensionality reduction techniques capable of capturing the same amount of information contained within the features, while reducing the dimensionality of the feature set. A common approach to this is Principal Component Analysis, or PCA, outlined first in (Bryant & Atchley, 1975) which can reduce the dimensionality of the feature space by computing a set of principal components which represent the majority of the feature space, at a lower dimensionality. Adding an additional step of dimensionality reduction to the classification algorithm can reduce the computation effort of the algorithm, as well as reduce redundancy due to cross-correlation of features.

6.2.3 Exploration with Neural Networks

For purposes of fraud detection, the application of neural networks (Rumelhart, et al., 1986) have been applied successfully. For example in (Fu, 2016), the application of Neural Networks for fraud detection have shown very good classification capabilities. A Neural Network functions by mimicking the structure of neurons in the biological brain. It would be worth additional effort to produce a comparative work between our proposed method via HMM modeling versus a fraud detection approach that applies a Neural Network based model.

Appendix

Appendix A. Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm, first outlined in (Dempster, et al., 1977), provides a framework for estimating the probabilities of unobserved latent variables, and their influence on observed variables. The EM algorithm assesses the probability of an event of a hidden set of events $\boldsymbol{S} = (S_0, S_1, \dots, S_T)$ occurring, provided the observed variables $\boldsymbol{y} = (y_0, y_1, \dots, y_T)$. We denote the probability of event $\boldsymbol{S}$ as,

$$P(\boldsymbol{S}) = \theta_s \tag{6.2.1}$$

Where θ_s denotes the parameters of the model. We use a shorthand, $\theta_{y|s}$, to denote $P(\boldsymbol{y}|\boldsymbol{S})$, which describes the probability of the sequence of observations provided the hidden state sequence $\boldsymbol{S}$.

$$\theta_{y|s} = P(\boldsymbol{y}|\boldsymbol{S}) \tag{6.2.2}$$

The Expectation Maximization (EM) algorithm, allows us to estimate model parameters where there exists a hidden intermediate state, $\boldsymbol{S}$, that behaves as a Markov process. First, we define the log likelihood as,

$$\mathcal{L} = \log P(\boldsymbol{y}|\theta) = \log \prod_S P(y, S|\theta) = \sum_S \log P(y, S|\,\theta) \tag{6.2.3}$$

Therefore, using maximum likelihood estimation, we can estimate the parameters that maximize this log-likelihood value.

$$\theta_{ML} = arg\max_{\theta} \mathcal{L} \tag{6.2.4}$$

The EM algorithm proceeds with a random initialization of parameters $\theta^{(0)}$. In the *Expectation-Step*, we compute the expected value of $\theta^{(i)}$ at iteration i, as,

$$Q\left(\theta, \theta^{(i-1)}\right) = E_{S|\boldsymbol{y},\theta^{(i-1)}}[\log P(\boldsymbol{y}, \boldsymbol{S}|\theta^{(i-1)})] \qquad (6.2.5)$$

Where the expected value is defined as,

$$Q\left(\theta^{(i)}, \theta^{(i-1)}\right) = \sum_{S} P(\boldsymbol{S}|\, \theta^{(i-1)}) \log P(\boldsymbol{y}, \boldsymbol{S}|\theta^{(i)}) \qquad (6.2.6)$$

Next, in the *maximization step*, we maximize $\theta^{(i)}$ with respect to the previous parameters, $\theta^{(i-1)}$.

$$\theta^{(i)} = arg \max_{\theta} Q\left(\theta, \theta^{(i-1)}\right) \qquad (6.2.7)$$

We iterate i through successive iterations, until the algorithm reaches a point of convergence where,

$$\left\|\log P(\boldsymbol{y}, \boldsymbol{S}|\theta^{(t+1)}) - \log P(\boldsymbol{y}, \boldsymbol{S}|\theta^{(t)})\right\| < \epsilon \qquad (6.2.8)$$

In this scenario, ϵ, is so small there is no significant difference between the log likelihood of the previous parameters and current parameters. The proof of the convergence of the EM algorithm can be found in (Wu, 1983).

Appendix B. Code Repository

The computer code written to build the feature aggregation and perform the classification techniques was written in both the R Programming Language and Python. The link to the source code can be found here: https://github.com/larkz/fraudDetection.

References

Altendorf, E., Brende, P., Daniel, J. & Lessard, L., 2005. *Fraud Detection for Online Retail using Random Forests.* s.l.:s.n.

Baum, L. & Welch, L., 1966. Statistical Inference for Probabalistic Functions of Finite State Markov Chains. *Institute for Defense Analysis.*

Bolton, R., 2002. Statistical Fraud Detection: A Review. *Statistical Science,* Volume 17, pp. 235-249.

Breiman, L., 1984. *Classification and Regression Trees.* s.l.:Wadsworth Int. Group.

Breiman, L., 2001. *Random forests.* s.l.:Machine Learning.

Bryant, E. & Atchley, W., 1975. Multivariate Statistical Methods, within Group Covariation.

Cavusoglu, H., 2004. Configuration of Detection Software: A Comparison of Decision and Game Theory Approaches. *Decision Analysis,* p. 131–148.

Choi, H., Romberg, J. & Baraniuk, R., 2000. *Hidden Markov Model Tree Modelling of Complex Wavelet Transforms.* s.l.:Dept. of Electrical and Computer Engineering Rice University.

Cramer, J., 2002. *Logit Models from Economics and Other Fields.* s.l.:Cambridge University Press.

Dempster, A., N.M., L. & D.B., R., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society,* pp. 1-38.

Dhok, S. & Bamnote, G., 2012. Credit Card Fraud Detection Using Hidden Markov Model. *International Journal of Advanced Research in Computer Science.*

Eberhardt, J., 2015. *Bayesian Spam Detection.* s.l.:University of Minnesota Morris Digital Well.

Fu, K., 2016. Credit Card Fraud Detection Using Convolutional Neural Networks. *International Conference on Neural Information Processing.*

Han, X. & Lee, S., 2006. *Application of Random Forest Algorithm to Machine Fault Diagonosis.* s.l.:WCEAM.

Hartigan, J., 1975. Clustering Algorithms.

Kriegel, H.-P., 1998. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery.*

Liu, C., Chan, Y. & Kazimi, S., 2015. *Financial Fraud Detection Model: Based on Random Forest.* s.l.:International Journal of Economics and Finance.

MacQueen, J., 1967. Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Math- ematical Statistics and Probability,* p. 281–297.

Marazanto, R., 2010. Fraud Detection in Reputation Systems in e-Markets using Logistic Regression. *ACM.*

Moskovitch, R., 2009. *Identity Theft, Computers and Behavioral Biometrics.* s.l.:s.n.

Netzer, O., Lattin, J. & Srinivasan, V., 2008. A Hidden Markov Model of Customer Relationship Dynamics. *Marketing Science.*

Quinlan, J. R., 1986. *Induction of decision trees.* s.l.:Machine Learning.

Rabiner, L. & Juang, B., 1986. *An Introduction to Hidden Markov Models.* s.l.:IEEE ASSP Magazine.

Raj, B. & Portia, A., 2011. Analysis on Credit Card Fraud Detection Methods. *International Conference on Computer, Communication and Electrical Technology.*

Rumelhart, D. E., Hinton, G. E. & Williams, R. J., 1986. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*

Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E., 1998. *A Bayesian Approach to Filtering Junk Email.* s.l.:Microsoft Research.

Samet, O., 2013. *Introduction to Online Payments Risk Management.* s.l.:s.n.

Shalizi, C., 2017. *Advanced Data Analysis from an Elementary Point of View.* s.l.:s.n.

Shen, A., Tong, R. & Deng, Y., 2007. Application of Classification Models on Credit Card Fraud Detection. *IEEE.*

Srivastava, A., Kundu, A. & Sural, S., 2008. Credit Card Fraud Detection Using Hidden Markov Model.

Udo, W. & Taudes, A., 1987. Stochastic models of consumer behaviour. *European Journal of Operations Research,* pp. 1-23.

Viaene, S., Derrig, R. & Dedene, G., 2004. *A Case Study of Applying Boosting Naive Bayes to Claim Fraud Diagnosis.* s.l.:IEEE Transactions on Knowledge and Data Engineering.

Whitrow, C., Hand, D., Jusczack, P. & D., W., 2009. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery.*

Wu, J., 1983. On the Convergence Properties of the EM Algorithm. *The Annals of Statistics* .

Zucchini, W. & MacDonald, I., 2009. *Hidden Markov Models for Time Series, An Introduction using R.* s.l.:Taylor & Francis Group.