Financial Fraud Detection and Data Mining of Imbalanced Databases using State Space Machine Learning

by

Deitra Sawh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2015

©Deitra Sawh 2015

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Risky decisions made by humans exhibit characteristics common to each decision. The related systems experience repeated abuse by risky humans and their actions collude to form a systemic behavioural set. Financial fraud is an example of such risky behaviour. Fraud detection models have drawn attention since the financial crisis of 2008 because of their frequency, size and technological advances leading to financial market manipulation. Statistical methods dominate industrial fraud detection systems at banks, insurance companies and financial marketplaces. Most efforts thus far have focused on anomaly detection problems and simple rules in the academic literature and industrial setting. There are unsolved issues in modeling the behaviour of risky agents in real-world financial markets using machine learning. This research studies the challenges posed by fraud detection, including the problem of imbalanced class distributions, and investigates the use of Reinforcement Learning (RL) to model risky human behaviour.

Models have been developed to transform the relevant financial data into a state-space system. Reinforcement Learning agents uncover the decision-making processes by risky humans and derive an optimal path of behaviour at the end of the learning process. States are weighted by risk and then classified as positive (risky) or negative (not-risky). The positive samples are composed of features that represent the hidden information underlying the risky behaviour.

Reinforcement Learning is implemented as unsupervised and supervised models. The unsupervised learning agent searches for risky behaviour without any previous knowledge of the data; it is not "trained" on data with true class labels. Instead, the RL learner relates samples through experience. The supervised learner is trained on a proportion (e.g. 90%) of the data with class labels. It derives a policy of optimal actions to be taken at each state during the training stage. One policy is selected from several learning agents and then the model is exposed to the other proportion (e.g. 10%) of data for classification. RL is hybridized with a Hidden Markov Model (HMM) in the supervised learning model to impose a probabilistic framework around the risky agent's behaviour.

We first study an insider trading example to demonstrate how learning algorithms can mimic risky agents. The classification power of the model is further demonstrated by applying it to a real-world based database for debit card transaction fraud. We then apply the models to two problems found in Statistics Canada databases: heart disease detection and female labour force participation.

All models are evaluated using appropriate measures for imbalanced class problems: “sensitivity” and “false positive”. Sensitivity measures the number of correctly classified positive samples (e.g. fraud) as a proportion of all positive samples in the data. False positive counts the number of negative samples classified positive as a proportion of all negative samples in the data. The intent is to maximize sensitivity and minimize the false positive rate. All models show high sensitivity rates while exhibiting low false positive rates. These two metrics are ideal for industrial implementation because of high levels of identification at a low cost.

Fraud detection rate is the focus with detection rates of 75-85% proving that RL is a superior method for data mining of imbalanced databases. By solving the problem of hidden information, this research can facilitate the detection of risky human behaviour and prevent it from happening.

Acknowledgements

Thank you Professor K. Ponnambalam, my supervisor, for your support, guidance, and patience during my PhD education.

Thank you to my committee members, Professors Hipel, Karray and Kolkiewicz for providing valuable feedback and believing in my ability to produce this body of work.

Until now I have not had professional mentors I could look up to; now I have four.

Thank you to my friends and colleagues who have supported me through this process. Your technical support, advice, stimulating conversation, and friendship were all fundamental to the success of this work.

I wish to express my love and gratitude to my beloved family for their support, feedback and consistent love for the duration of this degree, and throughout my life. I love you.

Dedication

I dedicate this thesis to anyone who has ever left the tribe to find a better source of food and found paradise. Gassho.

Table of Contents

List of Figures

List of Tables

List of Acronyms

NN	Artificial Neural Network
AUC	Area under the ROC curve
BAV	Bellman Action-Value equation
CHHD	Canadian Heart Health Database
CVD	Cardiovascular disease
DB-PDF	Double Bounded Probability Density Function
DCFD	Debit Card Fraud Database
EM	Emissions matrix
FICO	Fair Isaac Corporation
FIS	Fuzzy inference system
FN	False Negative
FP	False Positive
HyF	RL-HMM hybrid with fuzzy discretization
HyQ	RL-HMM hybrid with quantile discretization
HMM	Hidden Markov Model
IC	Inconsistency criterion
IIROC	Investment Industry Regulatory Organization of Canada
KNN	K-Nearest Neighbour
LFS	Labour Force Survey
LR	Linear Regression
LVF	Las Vegas Filter
MDP	Markov Decision Process
NASDAQ	National Association of Securities Dealers Automated Quotations
RL	Reinforcement Learning
RL-HMM	Reinforcement Learning - Hidden Markov Model (hybrid)
ROC	Receiver Operator Characteristic
SMOTE	Synthetic minority over-sampling technique
SVM	Support Vector Machine
TM	Transition probability matrix
TN	True Negative
TP	True Positive
ε-greedy	epsilon greedy (learning algorithm)

List of Variables

α	Learning parameter in Bellman Action-Value equation
$\alpha_t(i)$	forward probability variable
$A(i)$	Number of actions available at state i
β_a^W	Precision of W_a
β_a^n	Precision update variable
c	Cut-off value representing the percentage of states classified as fraudulent
δ_m	Number of discrete values feature m is represented by in the database
ε	Percentage of time policy explores the database (i.e. randomly select an action)
λ	Model parameter set in HMM
$P(a)$	Probability of selecting action a (for Boltzmann learning policy)
p_{ij}	Probability of transitioning from state i to state j
Φ	Emissions probabilities in HMM
π_i	Initial state probability for state i
K	Observation paths in HMM
M	Number of features in a database
N	Number of states
O	path of observable emissions
Ω	Finite set of γ types of observations
Q	Subset of states in HMM
$Q(i,a)$	Q-value for state-action pair (i,a) in RL
$r_{ij}(a)$	Reward in RL when agent moves from state i to state j by selecting action a
S	finite set of N states in HMM
$\sigma_{W_a}^2$	variance of W_a (for optimal learning policy)
t	time
τ	temperature (for Boltzmann learning policy)
θ_a^n	Estimate of μ_a
μ_a	mean of W_a
V_π	Optimal weights on each state in RL for policy π
W_a	Reward associated with RL action a in the optimal learning policy
X_t	Random variable at time t
γ	Number of observations/emissions
ζ	Difference in optimal state weights between two policies

Chapter 1
Introduction

Data mining of large databases has increased in societal importance as computers are faster, storage is larger and a large amount of data of varying data types is encountered. Generally, data mining is the search for patterns, behaviours and anomalies that cannot be easily calculated from the data or gleaned from a visual representation.

Databases in industry generated by human behaviour often may have examples that reflect an exchange of risk for reward, for example, financial market trading. Risk may be both a qualitative and quantitative measurement of what someone is willing to sacrifice for a potential reward [[1], [2], [3]]. High risk is usually associated with high rewards. An example of a risk-reward system is a financial market in which the corresponding risky behaviour is insider trading. Health is another example of a system in which there is a risk-reward relationship associated with lifestyle characteristics such as high stress and poor diet [4].

A paradigm shift in the approach to data mining of risk-reward systems is introduced in this thesis by modeling, isolating and exploiting two types of risky human behaviour: collusion and repeated abuse. Collusion is a secret agreement between two or more persons for a deceitful or illegal purpose [5]. This definition is extrapolated to be "two or more database features" for a "risky" purpose where a "feature" is a column in the dataset. A recent example of collusion in finance is in the foreign exchange markets where traders at different leading global financial institutions colluded to manipulate benchmark foreign exchange rates between 2009 and 2012 [6].

Repeated abuse occurs when consistently higher than average rewards are associated with risky activity. "Abuse" is a term known to be associated with taking advantage of a system, such as finance or the human body.

A general exploration of collusion and repeated abuse has not been previously researched in data mining literature and is the foundation of this thesis [7]. In addition to a lack of modeling general behaviours in risk-reward systems, current data mining solutions are limited by unadaptive solutions [8], domain-centric models [7], a lack of knowledge-based components [8] as well as lack of focus on misclassification [7].

In the following sections, the risk-reward problem will be described in the context of data mining as well as the representative learning and decision-making models. Performance evaluation and the need for algorithm design to meet the needs of industry are also discussed. In the last section the thesis outline will be presented.

1.1 Problem Description

The main objective of most data mining applications is to accurately identify specified samples in the dataset. The objective of this research is to detect risky behaviour with a classifier. The classifier is evaluated based on its ability to find the risky samples (called sensitivity or recall) and misclassify non-risky samples as risky (called false positive). The datasets are primarily bi-class imbalanced; these are datasets in which there are two classes and the majority class (non-risk) is much larger than the minority class (risk). Ultimately the problem is to construct the most sensitive classifier with the lowest misclassification rate, not necessarily the most accurate classifier. The problem of characterizing risky behavior is approached using machine learning algorithms and statistical models.

1.1.1 Description of risk – reward relationship for data mining

Risky behaviour in risk-reward systems is modeled by feature values in the database sample corresponding to high rewards. The features are types of "behaviour" and the reward is a feature in the database summing up the benefit of that behaviour. Collusion and repeated abuse are represented by several samples having the same feature values corresponding to high rewards. In specific scenarios, high rewards can be associated with smart investing, or healthy living, for example. It is assumed that cases such as these do not exhibit the same feature values in all samples whereas risky samples in the thesis applications do share feature values.

1.1.2 Learning and decision making

To learn is to acquire knowledge, understanding, or mastery (of) by study, or experience [9]. Learning to solve a problem requires decision-making: selecting the optimal steps to reach a solution.
Machine learning has been employed by data miners across many industries to solve problems such as forecasting consumer behaviour (retail [10], online advertising [11]), and anomaly detection (health [4], finance [7] [12]). In this thesis, machine learning is used to learn the behaviour of the risky person through experience. The learning process consists of an "agent" making decisions as it explores the database to derive optimal decisions for achieving the highest rewards. Machine learning algorithms are known to require a lot of experience to draw conclusions [13]; this means they use a lot of data and time. To minimize this aspect, a statistical model was integrated with the optimal behaviour paths to uncover the samples with groups of feature values resulting in the highest rewards.

1.1.3 Classification of imbalanced databases

"Bi-class" or "binary" classification problems can be considered as anomaly detection problems, particularly in imbalanced databases [14]. For the purpose of binary classification, risky samples are

labeled as “positive” and non-risky samples as “negative”. Because of the imbalanced distribution of the classes, traditional classifiers can easily find the negative class because there are so many samples to use for modeling. These classifiers do not describe the positive class data at all and usually consider it as part of the negative class. This results in high misclassification costs, which are unacceptable in industrial applications [15]. Approaches to the problem manifest in three ways: data level techniques to balance the data, algorithm techniques which model the minority class, cost-sensitive methods that combine data and algorithm techniques by assigning misclassification costs to classes [15]. He and Garcia [16] describe the objective of classification for imbalanced data is to obtain high accuracy for the minority class while not jeopardizing the accuracy of the majority class. Therefore, an algorithm technique is used in this thesis to solve the problem. It is evaluated by specific metrics to reflect the ability of the classifier to identify the minority class. This objective is further explored in the thesis.

1.1. Research objectives and scope

The objective of this research is to accurately represent the risky behavioural patterns of agents in risk-reward systems. The problem is approached by constructing three models: an unsupervised Reinforcement Learning Search model, a Reinforcement Learning Classifier and a hybridized classifier using a Reinforcement Learning algorithm and a Hidden Markov Model. To capture the agents’ risky behaviour three learning algorithms are employed: ε-greedy, Optimal Learning, and Boltzmann learning. The classifier is evaluated on four real-world databases, one of which contains balanced classes to test its applicability outside of anomaly detection problems. Ultimately, the intention is to integrate the model into an industrial system; the goal is achieved when a sensitive classifier is constructed with minimal misclassification.

1.2. Thesis outline

The requisite background knowledge and a general literature review of relevant research on machine learning and statistical models in the context of data mining are presented in Chapter 2. In Chapter 3, the Reinforcement Learning algorithms and Hidden Markov Model formulations are proposed. The methods are then applied to financial fraud, heart disease, and female labour force participation in Chapter 4. The conclusions and contributions to knowledge are summarized in Chapter 5.

Chapter 2
Background and Literature Review

2.1 Introduction

The exciting problem of representing risky behavioural patterns of humans in risk-reward systems is best understood contextually. Collusive and repeatedly abusive behaviour are known to be systemic in financial fraud and are both manifestations of "greed" [17]. The research conducted here was motivated by the author witnessing such behaviour for insider trading while working at a bank. The specific type of insider trading observed is called "front running", the practice of manipulating rates on a client's order to skim off extra points for the bank. Manipulation usually involves some knowledge of future trades (e.g. insider information) to know the direction of the market. This type of financial fraud can be collusive and frequently involves repeated abuse; a trader who is willing to front run one client is likely willing to front run many clients. Front running is the main lever Bernie Madoff used [18] [19] to execute his Ponzi scheme in the years preceding the Global Financial Crisis of 2008; it is a common practice in the financial markets regulated in the equity and derivative markets by IIROC [20] in Canada. One of the chief investigators of the Madoff case, Harry Markopolos, claims that front running is a widely occurring, common abuse that is openly tolerated [19] but has widespread consequences. It is therefore necessary to characterize and identify this behaviour in the trading book so as to prevent and minimize the damage to investors and the financial markets.

Palshikar and Apte [21] approached collusion detection using a graph-clustering algorithm, playing on the "network" aspect of collusion sets. Their work inspired Wu et al [22] to focus on one type of collusion-based financial crime – circular trading in stock markets. This is a practice of colluders (called a "collusion set") circulating a large amount of shares within a short period of time to feign demand for the stock. They approached the problem using a standard HMM with the Nelder-Mead simplex method for parameter estimation. The term "collusion" appears [19] several times in descriptions of fraudulent scenarios where multiple participants perpetuated fraud. The manipulation of foreign exchange [6] and interest rates [23] are examples of collusive behaviour for which legal proceedings are in progress for traders at several banks colluding over instant messaging to manipulate rates for profit.

More generally, insider trading is a crime for which systems have been constructed for detection. In the American stock exchange, NASDAQ [24], the problem of insider trading is attacked by reviewing market activity for stocks that will have material news released that day by the stock-issuing company [25]. Their treatment of insider trading is focused on trading that occurs on material non-public information. The main limitation is that the system is not flexible for new domains or markets and requires a lot of maintenance due to its level of detail. Other fraud detection systems rely on scenarios of events (time-

based) and known patterns [8]. The scenarios and patterns are updated after consultation with domain experts, a detection paradigm that lacks automated adaptability.

On a larger scale, financial fraud is a pervasive practice ranging from mortgage fraud to accounting fraud, money laundering to insider trading [7]. This research focuses on two types of financial fraud: insider trading (including front running), and plastic card fraud (debit card fraud). Lu states two challenges to fraud research [26]: 1) secrecy with regards to techniques - banks do not want people to know their techniques so the public can get around them, 2) limited availability of real fraud records. These challenges were overcome by accessing public information and using techniques gleaned from the author's work experience in the financial industry.

The authors of [27] comment that the major task in fraud detection is the construction of algorithms or models that can learn how to recognize a variety of patterns [27]. While collusion and repeated abuse can be viewed as two patterns, they cover a large behavioural space and generalize a lot of patterns. They are also not content specific and, within an adaptable system, can classify many types of behaviour under one umbrella.

Zhang and Zhou [27] emphasized that fraud detectors require systems that can detect rare events, outliers or noise. There is evidence that financial fraud is often executed by the same individual repeatedly via transactions in which he is making higher than normal profits [17]. Traders are aware that sophisticated algorithms exist to detect their transactions. Therefore, the risk of being detected is lower when trades are slightly more profitable than average trades as opposed to executing extremely profitable trades. Repeated abuse can be characterized by frequency and density. Plastic card fraud experiences this phenomenon where several transactions are executed rapidly and in nearby locations. A systems diagram of a typical industrial financial fraud detection system with inputs and outputs is shown in Figure 2-1.

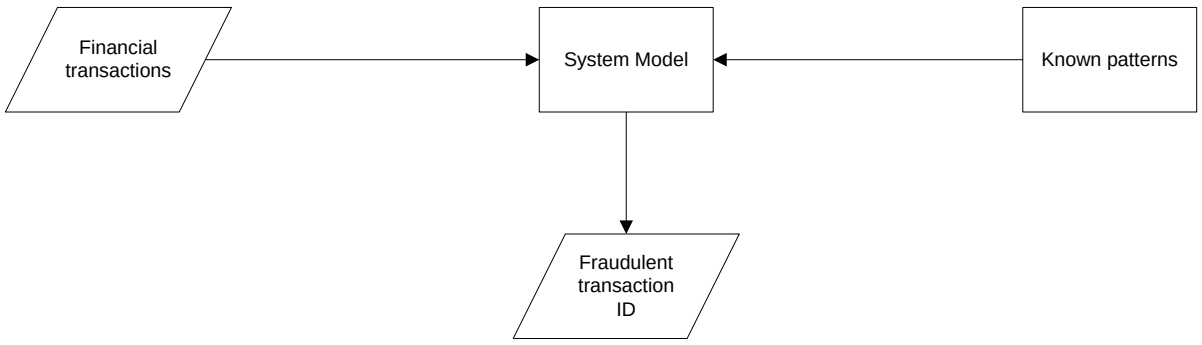

Figure 2-1: Financial fraud detection systems diagram

Financial transactions are the raw data that is preprocessed, grouped and classified based on the System Model used by the financial institution. Examples of the System Model are Linear Regression, Artificial Neural Networks, and Rule-based systems [7]. The resulting patterns are compared to known patterns

and those that either match known patterns of fraud or do not match regular patterns of behaviour are output as fraudulent transactions or "breaks" [28]. These breaks are not necessarily descriptive. The entire transaction is visible, but the reasons for which it was flagged are because it broke a rule or was anomalous to known behaviour. Since the fraud detection is conducted on a one-off basis, seemingly fair transactions go undetected. However, when several "fair" transactions are grouped together by an individual or asset, the behaviour can be illegal. Therefore, current industrial methods are not focused on detecting groups of fraudulent behaviour such as those modeled in this thesis.

In [19], Francis suggests that financial regulators must act like public health experts and constantly search for pathologies (e.g. criminal pathologies) that have the capacity to spread and cause severe crisis. The health analogy of pathology of financial fraud allowed us to consider general systems that demonstrate risky pathologies.

Financial systems are generalized as "risk-reward" systems because humans acting on the system absorb some level of risk for reward. This system representation can then be extended to other systems in which risk is exchanged for reward. An example of such a system is the human body. Humans make risky decisions with respect to their lifestyle choices in exchange for heart disease. Conversely, a non-risky (healthy) lifestyle is typically rewarded with long life. Repeated abuse is a characteristic of human systems where poor choices are made daily. Additionally, different choices can collude in the human body with deleterious results [4]. Dahlof observed that moderate risk levels in several risk factors is often more dangerous to health than large risk in one risk factor. Heart disease is commonly detected by searching clinical data where there are more cases of health than disease.

Diagnosis of heart disease has been approached from a risk perspective in the literature [29] [30]. The authors remark that in the emergency room, clinicians require a decision-making tool to assess risk and diagnosis. Liu et al [30] use a combination of data level sampling processed through an ensemble of support vector machine classifiers (SVM) from which a risk score is generated. They simulate the real-world medical setting where more than one opinion is sought before making final decisions.

Heart disease has been generalized in this research to a "risk-reward" system based on the literature and that it is related to "risky behaviour". Health is certainly a more complicated domain with underlying genetic and biological systems that we cannot necessarily control nor fully understand. In the interest of studying imbalanced databases we pursued heart disease with the understanding that a domain expert would be required in order to accomplish a more thorough investigation in the health care domain.

In the context of data mining, isolating cases of collusion and repeated abuse can be a classification problem: from a group of samples, a subset is classified as risky, and the rest of the samples are classified as non-risky. This can also be considered as an anomaly detection problem in which the model figures out or is instructed as to what is "normal" behaviour, and then uses a distance calculation to compare new

samples to “normal” samples. The main problem with an anomaly approach is that suspicious samples are dependent on described normal or average behaviour in the database, and anything else is labeled fraud.

Many different algorithms have been developed to solve the problem of fraud detection, and more generally, classification: linear regression (LR), artificial neural networks (NN), support vector machines (SVM), and K-nearest neighbor (KNN). Because classification is a broad area, the following literature review is performed only on some of the methods related to fraud detection and generally, imbalanced databases. Imbalanced databases have many more samples belonging to some classes than others whereas balanced databases have a consistent number of samples attributed to each class. A deeper discussion of imbalanced databases will follow later in Chapter 2. The literature review will also discuss Reinforcement Learning and Hidden Markov Models which were used to develop new methods for classification of imbalanced databases in the thesis. The sections in this chapter will present detailed background information on relevant concepts and mathematical formulations popular in the literature.

2.2 Important Reinforcement Learning concepts and techniques

Reinforcement Learning (RL) is a computational approach to learning from interaction between an agent and its environment [31]. It is a type of goal-directed learning in which the agent receives a numerical reward signal. RL is model-free; it does not fit the data to a function nor estimate function parameters. In practice, this means that it does not require a transition probability matrix as in a Markov Decision Process (MDP) [32]. It simply begins searching through the environment without any examples of desired behaviour by making decisions and assigning scores to each example of behaviour [33]. The study of behaviour dates back to 1911 in which Thorndike [34] studied animal psychology. He concluded that animals learn from experience and associations, not memory or imitation. In particular, repetition of an experience consecrates a reaction in the animal’s mind to reach a goal. RL applies the concept of repetition by frequently visiting states of the system that provide the highest reward.

RL is classically applied in the field of control theory where actions are repeated and modified until stability or an optimal policy is achieved [33] [13]. The applications range from manufacturing to playing games to optimal routing [35]. RL appeared in the data mining literature as part of a Learning Classifier System in 1978 but gained popularity in the early 1990s [36]. Since then, RL has been applied to classification [37] [38], clustering [39] [40], and outlier detection problems [41].

The application of RL in this thesis was motivated by financial fraud investigative techniques used by Lu [41]. Lu's exploration of fraud detection hinged off constructing a “fraud case builder”. The goal of the algorithm was not to find fraud; instead it was to gather the elements contributing to fraud, and build policies that dictate fraudulent behaviour. RL is an improvement to current systems by performing the

role of identifying suspicious data. Due to its "trial and error" approach, it eliminated the need for expert intervention because it built a body of evidence based on best rewards [42]. Sawh et al. [43] applied RL to two problems in finance: derivation of an optimal trading strategy and detection of insider trading. Lu [41] and Sawh et al. [43] demonstrated how operating in a model-free environment increased the agent's ability to detect new instances of anomalous data. RL does not require knowledge of the entire environment; calculations are based on a temporal differencing equation which only looks one step ahead. This is in contrast to learning models that examine an entire database or network [44]. RL examines data directly by linking records with large rewards; other learning methods often group records based on statistical features [45]. The main disadvantage of RL is that sometimes the problem can have a large state space; it takes a long time to converge and requires a lot of data to learn the problem [33] [46]. The ability to operate unconstrained can be a disadvantage because it might identify anomalies that do not characterize anomalous behaviour and draw incorrect conclusions. Improvement to RL techniques will be presented later in this thesis.

2.2.1 Reinforcement Learning components

The RL problem is described by three main components: an agent, an environment and an action. In engineering terms, the components are analogous to a controller, a controlled system/plant and a control signal. A schematic of the components is shown in Figure 2-2.

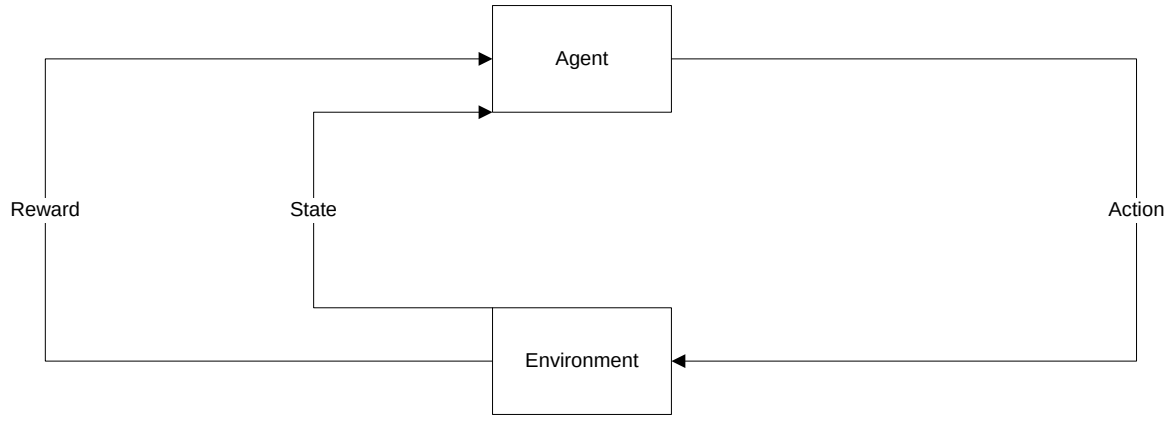

Figure 2-2: **Component diagram of agent-environment interaction**

The agent interacts with the environment by an action. Once the action is taken, the environment produces a numerical reward. The agent receives the information, evaluates it and selects an action based on the evaluation. The action takes the agent to the next state. In defining an environment, anything that cannot be changed arbitrarily by the agent is considered to be part of the environment.
Generally, states are the information available to the agent, actions are any decision it wants to learn how to make, and rewards represent desired outcomes. For example, if the system problem is a human

throwing a ball, then the states are the different positions of the human muscles in the act of throwing. The action is throwing the ball and the reward is the accuracy of the throw.

The difference between MDPs and RL is that RL does not require transition probabilities to move between states; rather, change of state is dictated by action selection. A reward function is also not required by RL. Instead, the reward is computed and collected as the agent traverses the environment. An important step in RL is that the agent must collect a reward before recognizing a new state as part of the feedback loop. In the example of throwing a ball, a human must assess the accuracy of his throw before throwing again. In doing so, he can modify his eye-hand coordination and muscle movement so that the next throw is more accurate. In the sport of baseball, athletes repeatedly throw balls to train their muscles during practice so that they do so automatically and accurately under the pressure of a game.

2.2.2 Bellman Action-Value equation

RL applies a temporal differencing method called the Bellman Action-Value equation (BAV) [31] for solving MDPs:

$$Q^t(i,a) = Q^t(i,a) + \alpha[r_{ij}^t(a) + \gamma Q^t(j,b) - Q^t(i,a)] \quad \textbf{(2-1)}$$

In the BAV equation (2-1), i, j are the states, a, b are actions, and $r_{ij}(a)$ is the reward obtained when the agent moves from state i to state j via action a. The equation also supplies a learning rate, α, which controls the rate of convergence of the model. RL methods can be episodic or continuous. For episodic applications, the discounting factor $\gamma = 1$ because the applications do not require discounting in time. All applications in this thesis were episodic. The symbol for time, t, represents the simulation number and if the RL algorithm is run over the input table 100 times, then $\mathrm{t} = 1, \ldots, 100$.

The agent begins at state i and takes action a. The agent's position is represented by a "state-action" pair, (i, a). The agent then obtains a new state-action pair, (j, b), from where the next iteration will begin. The resulting values output from equation (2-1) are called "Q-values". Q-values are stored in a matrix called a "Q-table" where each cell represents a value for $Q(i, a)$. The matrix is of size (number of states x number of actions). Q-values are considered to be "weightings" on each state-action pair.

Once the simulation has converged and the Q-table is complete, the optimal action (i.e. decision) is determined for each state. This translates into the optimal behaviour of the agent in each state. To do this, the heaviest weightings for each state are examined. The corresponding actions are the underlying behaviour to be extracted and studied. Therefore the maximum Q-value is selected for each state. The resulting column vector of states is called the "V-table".

The V-table represents the "optimal policy". A policy is a stochastic rule by which the agent selects actions as a function of states [31]. The optimal policy maps the best action for each state. The best action

represents the action that returns the highest reward in that state. The agent's objective is to maximize the amount of reward it achieves over time. The V-table gives the maximum return across all actions for each state.

Knowledge of the optimal policy is useful to decision-makers in three significant ways. It indicates what action to take, in the future, when in that state. It indicates the action resulting in the highest reward for each state. In a reverse engineering problem it indicates the optimal path of states and actions for the environment.

Temporal differencing is appropriate for data mining applications because it links records in a database. By differencing between two Q-values in equation (2-1), it finds a differential, or a distance in Q-values. The size of this difference contributes to the Q-value, and ultimately, the weighting of that state. For those states that are close in weights, the Q-value is smaller and thus the state-action pair is similar to the other state-action pairs. However, when the Q-values greatly differ, the larger Q-values represent a deviating state-action pair. Over many simulations one state-action pair is compared with several others and a distinct set of state-action pairs stand out as significantly weighted.

2.3 Learning algorithms

In Reinforcement Learning optimal policies are used to take actions. These policies are constructed by learning algorithms. The learning algorithms "learn" the problem by balancing between exploration and exploitation. Exploration of the database occurs when the learning agent varies the actions it takes in order to explore opportunities for finding higher rewards. This behaviour is equivalent to a non-risky human who varies his decisions according to the environment. Exploration also allows the agent to search for collusive agents in the database. Exploitation occurs when the learning agent has found the action that consistently returns higher rewards and repeatedly takes that action in order to achieve the highest expected value. This is the behaviour of a risky agent who repeatedly abuses a system.

The behaviour of the decision-makers must be learned based on the assumption that risky human behaviour is adaptive; in this sense the learning agent is "unsupervised" in that it is not told which state to go to and does not realize a reward until it reaches that state. RL is an unsupervised model when it is not trained on data containing the true classes (i.e. labeled data). When the agent is exposed to true class labels during the learning process it is "supervised" because it is trained on labeled data. The models in this thesis use RL in both its unsupervised and supervised forms.

There are four common policies in the literature: random, greedy, ε-greedy and Softmax [31]. The random policy is purely exploration; there is no preferred action. Therefore the probability of selecting an action is $\frac{1}{|A(i)|}$ where $|A(i)|$ represents the number of actions available at state i. The random policy is

called Sarsa in the literature [31]. Sarsa stands for: State, Action, Reward, State, Action. It tends to be exploratory and time-consuming.

The greedy policy is a purely exploitative learning algorithm, called Q-learning. Because action a has the highest Q-value for state i and is selected each time the state is visited, $Q(i,a)$ increases and therefore the same action is selected each time. Q-learning is an exploitative method that follows the greedy behavior. Q-learning is considered to be an "off-policy" method because it is always maximizing$(i,a)\ \forall\ a \in A(i)$. Therefore the policy for action selection is not "learned"; it is the same every time.

The ε-greedy policy balances exploration and exploitation based on the value of ε. Actions in the set of admissible actions are selected with probability $\frac{\epsilon}{|A(i)|}$. The Softmax policy, also known as Boltzmann Learning [47] is another method of balancing exploration with exploitation. The probability of selecting an action is given by the following formula:

$$P(a) = \frac{e^{Q(i,a)/\tau}}{\sum_{z\in|A(i)|} e^{Q(i,z)/\tau}} \qquad \textbf{(2-2)}$$

where $Q(i,a)$ is the action-values for state i when action a is taken. τ is a positive number called temperature which is the variable used for tuning the model to the problem. When the temperature is high the probabilities of taking all actions are the same; as the temperature decreases with the number of samples, the higher probabilities are assigned to actions with higher values for $Q(i,a)$. This results in an exponentially increasing probability for the most frequently selected action. Boltzmann learning has the advantage of not spending a lot of time on bad alternatives and works best when there are a small number of alternatives (e.g. less than 100) [47].

The third learning algorithm employed in this thesis is a statistical method outside of the RL literature; it is called "Adaptive learning" [47]. This is a novel approach to learning in the RL setting introduced by this thesis. In honour of Powell's text this learning algorithm will be called "optimal learning" going forward. Optimal learning is set in a Bayesian framework; the Bayesian perspective is primarily interested in estimating the mean and variance of μ which is the true mean of the random variable *W*. In the RL setting, the random variable *W* represents the reward associated with a RL action and is recast as W_a. The model assumes that a prior distribution of belief about the unknown parameter μ_a exists and that selection of one action over another is independent.

The action with the highest estimated mean reward is selected when the agent enters a state. The mean for the action selected is updated with the reward the agent collects.

It is assumed $W_a \sim N(\mu_a, \sigma^2_{W_a})$. θ^n_a is used to estimate μ_a after *n* observations for action *a*. β^n_a is used to estimate the precision of W_a as $\beta^W_a = \frac{1}{\sigma^2_{W_a}}$.

θ^n_a is iteratively updated using equation (2-3):

$$\theta_a^{n+1} = \frac{\beta_a^n \theta_a^n + \beta_a^W W^{n+1}}{\beta_a^n + \beta_a^W} \quad \textbf{(2-3)}$$

β_a^n is iteratively updated using equation (2-4):

$$\beta_a^{n+1} = \beta_a^n + \beta_a^W \quad \textbf{(2-4)}$$

Once the agent collects a reward W^{n+1} is set equal to the reward which represents the newest information in the model. The equations for θ_a^n and $\beta_a^n$are then updated for each action. Then the action producing the highest mean is selected: $\max_{a \in A} \theta_a^n$.

The learning algorithms presented in this section were selected based on their diversity; varying from simple to complex, and a mix of artificial intelligence with statistical methods. Moreover, they most similarly mimicked a risky agent in a risk-reward system by repeatedly abusing the system (i.e. exploitation) while colluding with other agents (i.e. exploration).

The RL algorithm finds the optimal path of a risky agent. The path outputs are Q-values and rewards. This research used Q-values to determine risk in the RL Search and RL Classifier models. To assess state riskiness using rewards, a Hidden Markov Model was applied.

2.4 Important Hidden Markov Model concepts and techniques

The intuition behind Markov models is to use the Markov property to simplify a Markov chain of events.

$$P[X_{t+1}|X_t, X_{t-1}, X_{t-2}, \dots, X_0] = P[X_{t+1}|X_t] \quad \textbf{(2-5)}$$

(2-5) says that the value of X_t in the next time step is only determined by the most recent value of X_t and not the entire set of values X_t has taken in the past [48].

In practice, this means treating a stochastic process $\{X_t | t \in T\}$ as a first-order Markov process. Typical Markov model analysis assumes that the state sequence of the process is known and observable. In a Hidden Markov Model (HMM), the state sequence through which the process passes is unobservable. The objective of the HMM is to use a sequence of observations of the dynamics of the process to uncover the underlying hidden states [32].

Formally, a HMM is applied when the problem is to recover a sequence of states $x(t)$ from observed data $y(t)$. The main assumption is that a Markov chain is the underlying process with internal states hidden from the observer [32]. HMM solves the conditional probability in equation (2-6).

$$P[y(t)|x(t)] \quad \textbf{(2-6)}$$

The observations are the output of the system, and are also referred to as "emissions". HMMs are applied to state-based systems where the system outputs an emission when it arrives in a state. There are a finite number of possible emissions and states.

Hidden Markov Models are doubly stochastic processes. According to Ibe [32], this means that an underlying stochastic process is unobservable and can only be observed through another stochastic process that produces a sequence of observations. Using one observed stochastic process to derive another stochastic process makes it a doubly stochastic process. Figure 2-3 shows the two processes and the link between the states that make them sequential.

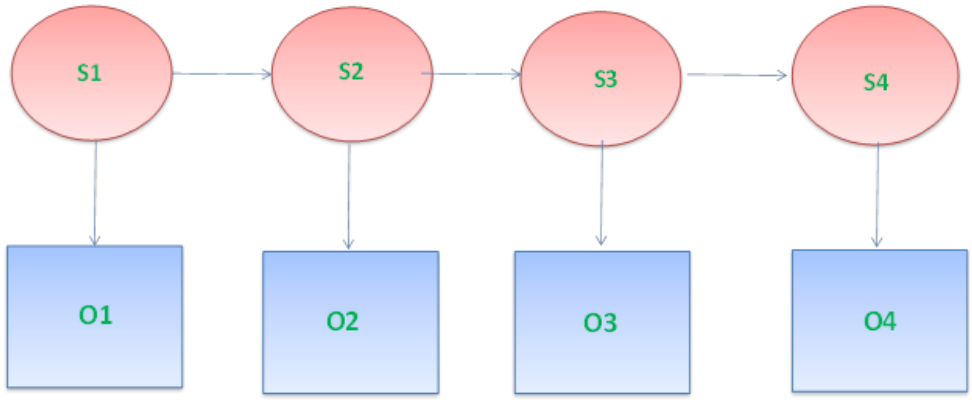

Figure 2-3: HMM representation

The square boxes represent the emissions O1, O2, O3, O4 that are some observable output from the system. These observations relate to hidden states S1, S2, S3, S4. The goal of a HMM is to find the states, in the correct order, that produced the emissions. The state sequence is found based on the information stored in two matrices: a transition matrix, $P[x(t+1)|x(t)]$ which describes the probability of an agent's transitions between states and an emissions matrix, $P[y(t)|x(t)]$, which describes the probability of the output given the current state.

HMMs have many applications, the most popular being in speech recognition [49]. The observed data in this case is sounds and the hidden states represent words. The application of HMM in this thesis was motivated by Eisler's application of HMM to uncover financial market volatility [50]. Volatility measures the uncertainty about returns of a stock [1] which is not directly observable from the stock prices themselves. Volatility is instrumental in the decisions made by financial market agents because it gives the perception of the risk associated with that stock [51]. Eisler et al. selected HMM so they could visualize the sample path taken by volatility over time. The research by this thesis is intended to uncover risky behaviour, a hidden process like volatility.

HMMs appear in the financial fraud literature as well. Wu et al [22] observed financial trades to uncover multi-variable states where the variables indicate whether a transaction is malicious and part of a collusion set. Khan et al [52] used HMMs to uncover spending behaviour via observed credit card transactions.

In the literature there are two types of problems solved using HMM:

1. To derive the sequence of states, indexed by time, from a set of observations.
2. To assign probabilities of classes in the test dataset.

The main disadvantage of HMM is that model training can take long and convergence is not guaranteed [32]. The model is constrained by probabilistic assumptions which can be too restrictive for some systems. The documented limitations were overcome in this research by training the model using RL and then applying the HMM probabilistic model assumptions to quickly derive risky states.

2.4.1 HMM components

In a HMM model with N states, γ emissions and $t = \{1, 2, 3, \dots, \mathrm{T}\}$ steps in a sequence, the fundamental variables are as follows:

S: finite set of N states $S = \{s_1, s_2, s_3, \dots, s_N\}$

Ω : finite set of γ types of observations $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \dots, \Omega_\gamma\}$

Q : hidden states $Q = \{q_1, q_2, q_3, \dots, q_\mathrm{T}\}$

O: path of observable emissions $O = \{o_1, o_2, o_3, \dots, o_\mathrm{T}\}$

Q is a subset of S

K : observation paths $K = \{k_1, \mathrm{k}_2, \mathrm{k}_3, \dots, k_K\}$

S is a hidden Markov process that is observed through Ω; therefore $\Omega = f(S)$ for some function f. S is the state process and Ω is the observation process. The model parameters calculated prior to solving the HMM are the transition probabilities ($P = \{p_{ij}\}$), emissions probabilities ($\Phi = \{\phi_i(o_T)\}$) and initial state probabilities ($\pi = \{\pi_i\}$); When describing an HMM it is customary to denote the triplet $\lambda = (P, \Phi, \pi)$ for the model parameters used to fully define the model.

The transition probabilities $P = \{p_{ij}\}$ reflect the probability of moving from state i to state j in a discrete first-order Markov chain. The Markov equation for transitioning between states is given by,

$$P(q_{t+1} = j | q_t = i, q_{t-1} = l, q_{t-2} = m, \dots, q_0 = n) = P(q_{t+1} = j | q_t = i) = p_{ij}, \quad \textbf{(2-7)}$$

describing the assumption that the future state depends only on the current state.

The probabilities satisfy conditions 1 and 2 [32]:

1. $0 \leq p_{ij} \leq 1$
2. $\sum_j p_{ij} = 1, i = 1, 2, \dots, N$, since the states are mutually exclusive and collectively exhaustive.

They are displayed in an [N x N] matrix P which will also be called a "transition probability matrix" (TM) in this thesis. It takes the form:

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{bmatrix}$$

The TM models the transitions between states thereby giving a fully-defined space from which to derive a state sequence.

A state transition diagram is a graphical representation of P. One is shown in Chapter 3 describing the fundamental transition assumptions underlying the algorithms developed during this research.

Another component of λ is $\Phi = \{\phi_i(o_T)\}$, the emissions matrix (EM). It is an [N x γ] matrix mapping the probability of an observation given that the system is in a specific state.

$$\phi_i = P(o_t|S_i) \quad \textbf{(2-8)}$$

The EM can be calculated using two methods:

1. Empirical method. The probabilities are derived from observations resulting from some dynamic process which transitions through states in sample data.
2. Theoretical method. The probabilities are derived from a probability distribution of expected observations when in each state.

Note that the rows of the EM sum to 1. This is because the total probability of state output emissions must equal 1.

The final component of λ are the initial state probabilities, $\pi = \{\pi_i\}$, otherwise known as the marginal probabilities. π_i describes the probability of a single event occurring. A joint probability is the probability that two events will occur simultaneously. p_{ij} is a joint probability because it gives the probability of an agent being in s_i and going to s_j. The marginal probability describes the probability that the system is in s_i, not accounting for where it transitioned to or from, and is used as the initial probability for state s_i of the system.

The system of equations represented by $Ax = b$ is used to solve for π. $A = P$ with the last row of P replaced with 1s to satisfy the sum of probability condition [53]. x is a vector of N marginal probabilities and $b = [0,0, \ldots 1]$.

The resulting probability output from the HMM is a posterior state probability. It is the conditional probability of being at a state given an observation and model λ.

$$P(Q|O, \lambda) \quad \textbf{(2-9)}$$

This calculation results in a [N x γ] matrix. Posterior probabilities condition on observations so the total probability for the observation is distributed among the states. The columns of a posterior probability matrix sum to 1.

2.4.2 HMM assumptions

There are three fundamental assumptions that need to be satisfied to use a HMM model [32]:

1. Markov Assumption
   The next state in a system depends only on the current state, and the transition probabilities are defined by equation (2-7). The HMM is first-order.

2. The stationarity assumption
   The state-transition probabilities are independent of the actual time the transitions take place. Thus, for any two times t_1 and t_2,

$$P\left(q_{t_1+1} = j \middle| q_{t_1} = i\right)\right) = P\left(q_{t_2+1} = j \middle| q_{t_2} = i,\right) = p_{ij} \quad \textbf{(2-10)}$$

3. The observation independence assumption
   The current observation or output is statistically independent of previous observations.

2.4.3 HMM fundamental problems

There are three fundamental problems that can be solved using HMM:

1. The learning problem. Given a set of observation sequences find a HMM that best explains the observation sequence. Find $\lambda^* = argmax_\lambda P(O|\lambda)$.
2. Evaluation problem. Given a model $\lambda = (P, \Phi, \pi)$, find the probability that observation sequence $O = \{o_1, o_2, o_3, \dots, o_T\}$ comes from that model. Find $P(O|\lambda)$.
3. Decoding problem. Given a model, $\lambda = (P, \Phi, \pi)$, find the most likely sequence of hidden states that could have generated a given observation sequence. Find $Q^* = argmax_Q P(Q, O|\lambda)$.

The learning problem estimates the HMM parameters P, Φ, π given a set of observations in a training set. The Baum-Welch algorithm is commonly used to solve this problem [32]. In this thesis, HMM parameters are calculated empirically from the observations collected by the RL agent, a procedure described in Chapter 3. The Baum-Welch algorithm is not used.

The goal of solving the evaluation problem is to find $P(O|\lambda)$. The evaluation problem estimates the probability of an emission o_t given the model $\lambda = (P, \Phi, \pi)$. The most straightforward way to find this probability is to enumerate all possible state sequences of length T and then sum their output probabilities [54]. The method forms a trellis of states and calculates the probability of each state occurring at a specific time in the sequence, t_i. [32]. For the calculation, compute a forward probability variable $\alpha_t(i)$ as defined in equation (2-11).

$$\alpha_t(i) = P(o_1, o_2, \dots., o_t, q_t = s_i|\lambda)\ t = 1, \dots, T;\ i = 1, \dots, N \quad \textbf{(2-11)}$$

$\alpha_t(i)$ represents the probability of being in state s_i at time t after having observed sequence $\{o_1, o_2, \dots, o_T\}$. It is calculated by summing probabilities for all incoming arcs in the trellis node. The forward algorithm is implemented by the following steps:

Initialization:

$$\alpha_1(i) = \pi_i \, \phi_i(o_1) \;\; 1 \leq i \leq N \tag{2-12}$$

In the initialization step, the forward probability variable for state *i* is calculated as the marginal probability for state *i* multiplied by the emission probability of observation 1 (o_1) coming from state *i*.

Induction:

$$\alpha_{t+1}(j) = \left\{ \sum_{i=1}^{N} p_{ij} \, \alpha_t(i) \right\} \phi_j(o_{t+1}) \;\; 1 \leq t \leq T-1; \; 1 \leq j \leq N \tag{2-13}$$

Iteratively the forward probability variable moves along a path of states using the transition probability, the forward probability from the previous step and the emission probability corresponding to the emission and state. The forward probability variable accumulates probability as it goes down a path so there is no need to save intermediate probability values.

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} P(O, q_T = s_i|\lambda) \tag{2-14}$$

The model terminates at the end of the observation sequence T. It sums over all terminal probabilities from all state paths. The forward algorithm is linear in time. The output is a [1xK] matrix with one probability assigned to each sequence of observations.

The Viterbi algorithm is commonly described in the literature to solve the decoding problem: find the most likely path of states specified by *P*, Φ, and O [55] [56] [54]. The output of the algorithm is a set of optimized sequential states. Optimality is defined by the state sequence that has the highest probability of producing the given observation sequence.

This research takes the perspective on risky human behaviour that it is independent of time. Because of this, a unique approach has been invented using posterior probabilities to solve the decoding problem independent of time as described in Chapter 3.

This section has provided an overview of the fundamental assumptions, problems and applications for HMM. The probabilistic assumptions provide a holistic approach to analysis of state-based systems.

2.5 Classification

Classification is a supervised learning method by which data is optimally grouped together based on similar characteristics. Supervised learning methods are trained on data with labels assigning the correct class to the sample. After training, the classifier is validated on the same dataset to tune the parameters

and then tested on a separate set of unlabeled data to evaluate its ability to identify the correct classes. In contrast, unsupervised methods such as clustering are trained on unlabeled data and draw conclusions based on specified conditions and metrics [57]. Classifiers are implemented using a variety of techniques; a short summary of statistical and machine learning classifiers are presented in this section.

2.5.1 Statistical Classification

The oldest classification methods are statistical and can be described in two phases; the "classical phase" is based on linear discrimination [58], whereas the "modern" phase is more flexible looking for joint distributions of features in each class. Statistical classification is characterized by having an explicit underlying probability model to describe each class [49] [57]. Examples of modern statistical classifiers are linear regression [7], rule-based methods [59], peer-group analysis [60], Bayesian methods [45], Hidden Markov Model classifier [49], and Gaussian mixture models [61].

In risk models, statistical tools are founded upon comparing observed data with expected values and generating "behaviour profiles" [62]. The output model is typically the same: the system generates "alerts" which are in the form of a "risk score" describing a probability of suspicion. According to [60], there are two approaches for using statistical methods for fraud detection, depending on whether the pattern of the fraud is known or not. If known, then pattern matching techniques are employed for classification. If unknown, anomaly detection methods are used. Anomaly detection methods can be considered as "one-class" problems where the normal behaviour is characterized and everything else is outliers.

The advantage of statistical methods is that they have been around for a long time and are therefore industrially acceptable in the current infrastructure. They are well understood, fast, quick to train, and achieve good accuracy for simple datasets. The biggest drawback is these models are usually constructed by statisticians and lack knowledge-based components. When new data arrives the parameters must be re-estimated to capture the dynamics of the new information. When used on imbalanced databases, they model the negative class and, as a result, have trouble finding the positive class.

Rule-based methods are considered as statistical classifiers because they are of a logical form: If {condition} then {outcome} and are implemented as a result of statistical analysis. The financial industry is heavily dependent on such methods because they are easy to implement, understand and investigate. Furthermore, a lot of banking fraud scenarios can be summarized into such conditional statements and has thus kept banking fraud to the low levels at which they currently exist [63].

Krivko [59] uses a hybrid model for plastic card fraud detection systems. His model first finds deviations of data from an account model of aggregated spending behaviour whereby he creates models for "groups"

then runs the deviating groups through a rule-based filter. Krivko's model is adaptive by modifying the degree of risk as new transactions enter the time window.

Current methods in the banking industry are rule-based ([59], [62], [64], [7]) which have proven to be successful but mostly used because they are easy to understand and implement. While rules have been sufficient, they are limited to fixed values and thus are not always dynamic. This constraint provides an opportunity for fraudsters to exploit the gaps in the rules.

Linear regression (LR) is a statistical model whereby one or more input variables x_i are used to predict one output variable y. When linear regression is used for binary classification it is called a logistic regression model and outputs a binary value. It is a very popular method in the fraud detection literature. Jha et al [65] apply a logistic regression model to credit card fraud; it is also used as a benchmark in [10] and is referred to 16 times as a financial fraud detection model in Ngai et al's literature review [7]. LR estimates parameters based on the dominating class in a binary problem and thus is insufficient for imbalanced problems.

HMM is considered to be a statistical classifier because it is fully described using probability. The properties of the HMM that make it so popular is the ease of implementation and a solid mathematical basis for classification problems. Khorasani et al use HMM to classify Parkinson's disease using human gait data [66]. They apply the Baum-Welch algorithm for training the model and estimating P,Φ and π. Their results show success on a balanced database. HMM is the most used classifier in emotion classification [49]. The drawbacks of HMM classifiers are that they require proper initialization for the model parameters before training and a long training time is often associated with them [49]. Other design issues of the HMM classifier include determining an optimal number of states, the type of observations and the optimal number of observation symbols.

Gaussian mixture models (GMM) have attracted considerable interest for data mining applications [61]. The underlying assumption of the GMM is that the dataset can be modeled using a Gaussian probability density function. In this research the dataset was purposely not modeled, or fitted to any distributions. Therefore GMMs were not applied.

The linear regression model is used as a benchmark for the results of this thesis. Rule-based models were reviewed due to their popularity in the financial industry and the Hidden Markov Model classifier was described because it is integrated in the invention proposed by this research.

2.5.2 Machine Learning

Machine Learning encompasses automatic computing procedures that learn a task from example. It endeavours to sufficiently mimic human perception, reasoning, and learning to make decisions from incomplete information [67]. Reinforcement learning is a machine learning technique. Examples of

machine learning classifiers are support vector machines (SVM) [68], and k-nearest neighbour classifiers [57]. Artificial neural networks (NN) are often considered under the general umbrella of "artificial intelligence" [58] but because of its learning capabilities to approximate human reasoning it will be considered here as a machine learning technique for classification.

Support vector machines (SVM) were developed in 1995 for binary classification problems [68] for linearly separable data. Based in pattern recognition, SVM maximizes a margin by creating a hyperplane which separates the classes. SVMs are often selected for classification because of their reliable performance and efficiency [30]. However, it is overwhelmed by the majority class instances in the case of imbalanced data sets [69].

K-nearest neighbour (KNN) is a machine learning algorithm which classifies samples according to a similarity measure, for example, Euclidean distance. It can be used in a supervised manner where it is first trained on labeled data and then the training samples are assigned to classes. It can also be used in an unsupervised manner by pre-selecting a number of clusters and then assigning samples using the distance measure to each cluster. When used in this manner it is called "K-means". The main advantage of KNN methods is that they are easy to use, intuitive and have had a lot of success on balanced databases. KNN was used to classify companies that violate accounting disclosure rules [70]. The positive class was 40% which made it a relatively balanced database. In the biomedical classification space, Majid et al balanced their database and then apply KNN to identify cancer [71]. The variable *k* indicates the number of neighbours to be located nearby the classified sample. Majid et al ran tests on both the imbalanced dataset and the preprocessed, balanced dataset; performance values increased tremendously when the data was balanced. After testing many values for *k*, the best results were when $k=1$.

Artificial neural networks (NN) consist of layers of interconnected nodes; these interdependencies incorporate nonlinearity allowing for very general functions to be modeled. NN combine the complexity of some of the statistical techniques with the machine learning objective of imitating human intelligence; in NN learning is done at an "unconscious" level by creating a non-linear equation and there is no accompanying ability to make learned concepts transparent to the user [58].

NNs are the most advanced fraud detection models implemented industrially. An example of an industrial leader is FICO [72] in their Falcon Fraud Manager for plastic card fraud detection [73]. Their systems provide alerts for plastic card fraud. Equifax [74] also uses a neural network based solution in their product called Equifax Gemini Verify Score [75] which is industrially known to be the best product on the market.

Ogwueleka et al [10] applied NN in their study of customer relationship management at an international retail bank. At the time of publication, they were the first to apply NN for predicting the behavior of customers in banks. The purpose was to classify customers as "normal loyal customers" and "abnormal

customers". They also compared their method to a logistic regression model which achieved 72% positive classification rate versus their NN positive classification rate of 94%.
Generally, the disadvantages of NNs are that they do not always converge and that empirical testing must be performed to find the number of nodes in the hidden layer. This can be a time-consuming process. Also, learning requires a large number of passes through the learning examples before converging. NNs are known to overfit (describe the noise instead of the underlying relationship in the data) and are difficult to interpret because of its black-box form. Also, adaptability is an issue because of the requirement to retrain the weightings on the nodes. Industrially, the main drawback is that the black-box nature of the NN makes it hard for users to understand what is going on and the results are largely ignored or incorporated with another model for fraud detection.

2.6 Feature representation

While the topic of this thesis is not feature selection, classification cannot be discussed without it. Feature selection reduces the dimension of multivariate data or extracts features from an original dataset [76]. Data mining tools hinge off the underlying data selected for mining.
A variety of feature selection methods exist in the literature as a method of dimensionality reduction [57]. The goal is to minimize the number of features prior to use of the classifier so as to only use variables important for understanding the underlying phenomena. A few popular feature selection methods were applied in this research (PCA [57], Jensen-Shannon divergence [77]) but the one sharing the same philosophical basis as the thesis was the Las Vegas Filter [78].
The fundamental basis of the Las Vegas Filter (LVF) is to find independent discrete features. The LVF evaluates the quality of its feature sets by calculating an inconsistency criterion (IC). Inconsistency means that the database rows match (without considering their class labels). By not including the class labels, it is an unsupervised learning method for feature selection [76]. It therefore finds independent features by selecting feature sets with as few matching rows as possible. The benefits of the LVF are that it is [78]:

1. Simple to implement
2. Calculates quickly
3. Presents many possible solutions
4. Independent of a learning algorithm

The main variables are: N = total number of features; S= current set of features; Sbest = best set of features; C=number of features; Cbest= best number of features; IC = inconsistency criterion; gamma=threshold for IC; db = dataset.

The LVF begins by considering a random subset of all features (S<=N). It modifies the database to only include feature set S and counts the number of rows that match other rows resulting in the inconsistency criterion , IC. Common risky behavior such as collusion and repeated behaviour is reflected in a database as rows possessing the same feature values for a set of features. This translates into a high value for IC which is represented by IC(count) in the algorithm. The difference between IC and the number of negative samples represent the anomalies in the database. These are the number of rows that do not match any other rows, and are not negative samples, represented by IC(final) in the algorithm. The purpose of the dimension reduction exercise was to minimize the anomalies in the database relative to all samples. The LVF also revealed that when there are less matching rows, more features are required for uniqueness. Since the goal of the exercise was to minimize the search space it required a minimal number of features that manifested into a minimal number of states. The algorithm is shown in Table 2-1.

Table 2-1: Las Vegas Filter

Las Vegas Filter – Main	
1	Define variables: N = total number of features, S= current set of features, Sbest = best set of features, C=number of features, Cbest= best number of features, IC=inconsistency criterion, gamma=threshold for IC, db = dataset
2	S=randomSet(N); C=numFeatures(S)
3	If C<=Cbest; check IC Function
4	If IC(final)<gamma; Sbest=S, Cbest=C
5	PrintCurrentBest(S)

IC Function	
1	For j = 1:num_rows
2	Samp=data(j,S)
3	Check if there are any other rows in the dataset that are the same as *samp* EXCEPT for the class label
4	Increase IC(count) by 1 everytime *samp* has similar rows
5	After the entire db is surveyed, calculate: IC(final) = (IC(count) – number of negative samples)/num_rows

2.6.1 Reward selection

The focus of the research was on risk-reward systems. The features in the database representing risky behaviour were the inputs for the search agent. The reward for the agent was a feature in the database that summarized the expected returns of the agent. In the financial fraud scenario, reward was the profit on the trade. In heart disease applications, the reward was "age" because an increased length of life is a reward for human health.

Two methods were used to find the reward best representative of the risk-reward systems studied:

1. Context-based reward. Intuitively determine the reward from knowledge of the system.
2. Correlation. The feature with the most positive correlation coefficient to the class vector is the reward.

The correlation coefficient represents the degree to which two variables movements are correlated [79]. It varies from -1 to +1 where +1 represents perfect correlation (two variables move together) and -1 represents negative correlation (two variables move opposite directions). For variables x and y it is calculated by $\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$.

2.7 Input data preprocessing

Inevitably in data mining some discretization of the data must occur [57] where the data is sliced into disjunct sub-intervals that are subsequently treated as discrete attribute values. A loss of information results from the process however it is necessary for summarizing data to output comprehensible knowledge by the system.

Two methods of discretization were applied in this thesis: hard discretization using statistical quantiles and soft discretization using a fuzzy inference model.

Hard discretization is defined by a threshold which generates boundaries between two crisp sets without overlapping. Quantiles are a statistical method where the k^{th} *quantile* of a set of values divides the data so that *k%* of the values are below *k* and *(100-k)%* of the values lie above. Quantiles are described using cumulative probabilities, from 0 to 1. For example, the lower quantile is denoted by *Q(0.25),* returning the value below which 25% of the values exist in the vector.

The Matlab [80] function "quantile" was used in this research. For vector x with N elements the method is:

1. Sort the values in vector x and attribute them to quantiles by calculating $(0.5/N), (1.5/N), \ldots, (N-0.5)/N)$.
2. Linear interpolate to compute quantiles between $(0.5/N)$ and $(N-0.5)/N)$.

For example, let vector $x = [1,2,3,4,5,6]$. $Q(0.5) = 3.5$. This means that 50% of the values in *x* lie below 3.5. $Q(0.33) = 2.5$ which means that one-third of the values lie below 2.5. In discretization, *Q(p)* are used to create adaptive intervals in which to bucket the continuous data.

Soft discretization is defined by a fuzzy set pair which forms a fuzzy partition between sets that can be overlapping. Zadeh [81] proposed fuzzy logic for the first time in 1965. It was invented to handle uncertain and imprecise knowledge in real world applications [67]. It is a powerful tool for decision making and is used as part of a Fuzzy Inference System (FIS). FIS is a form of classifier that fuzzifies input data, combines it based on rules and then produces an output. The 5 components of an FIS are:

1. Pre-processor to modify the raw data (e.g. normalize) to be put into the FIS.
2. Fuzzy input model which models each attribute in the raw data with membership functions. This is called "fuzzifying" the data.
3. A set of rules using logical operators to combine the attributes and map the data to an output membership function.
4. If a Mamdami model is used, then the output data must be operated on using a T-norm (e.g. min operator) to obtain the crisp output value (defuzzification).
5. If Sugeno model is selected, the output is already crisp and no further operations occur.

Bagheri et al [82] used an FIS as part of a forecasting tool for foreign exchange rates. Another financial forecasting application can be found in [83]. It was also applied in a breast cancer study [84] plus many other areas.

The advantages of a FIS are:

1. Does not require large memory storage.
2. Inference speed is very high.
3. It can handle incomplete information.
4. It can handle qualitative and quantitative data.

The disadvantages of a FIS are:

1. It is inflexible: rules are not adaptable since they cannot be inferred from the data as easily and are usually controlled by the designer.
2. Input and output data is fit to a membership function; the data is "modeled"

It is a particularly good way to combine data which is why it performs well as a classifier using if, and, or statements. In this thesis it was used only as a data preprocessor for input into the Hidden Markov Model using membership functions and rules. [82] and [83] both suggest that that the bellshaped membership

function is the one "usually adopted". The fuzzy model used in the thesis is a Mamdami type and described in Chapter 3.

2.8 Imbalanced databases

The classification literature has evolved into distinguishing between balanced and imbalanced problems [85], [15]. A balanced problem is when the classes are evenly distributed among the samples. In an imbalanced problem a small number of samples belong to one class (minority class) and this is the class that is identified as positive samples (e.g. fraud, heart disease). Imbalanced problems can arise in binary classification problems where a small sample of the population is a positive sample. Haibo He [86] described the fundamental problems facing learning algorithms in the imbalanced database space. The first is that there is a limited amount of theoretical understanding on the principles and consequences of the problem. Secondly, he addressed the need for publicly available benchmarks specially dedicated to imbalanced problems. Thirdly, practitioners should be using metrics for assessing the quality of imbalanced databases specifically. Until this delineation, accuracy (the number of correctly classified samples for all classes) was the benchmark for evaluation. [87] and [49] used accuracy to evaluate their classifiers and compare them. This is the simplest way to evaluate a classifier, and is the only reasonable choice for multiclass classifiers.

General solutions for handling imbalanced data at the data level are to undersample the majority class and oversample the minority class during training [30]. The SMOTE technique is popular in the literature [15] [88] for managing the imbalanced data by generating "synthetic" examples of the minority examples instead of oversampling with replacement [89] so that there is more balance in the number of minority and majority samples. At the algorithm level Kernel-based methods and active learning [86] are employed; ensemble methods are prominent in the literature [15] [85] whereby several classifiers are used to make decisions. Data and algorithm solutions are combined to create cost-sensitive methods to adjust the costs of classes to counter the class imbalance [88] [14].

The general systems structure for classifying imbalanced data is to have several data sampling methods produce sets that are processed by individual classifiers. The output of the individual classifiers are typically entered into a risk score calculation [88], a voting method [15] or cost structure [14] to determine the class for each sample. There is a lot of research into multiclass imbalanced models, the details of which can be found in [85] and [14].

2.9 Evaluation measures

Several measures have evolved for evaluating the ability of the classifier to find positive samples in an imbalanced database. The need for this was motivated by the fact that, if for example, 5% of the

population has heart disease and the classifier has a 95% classifier accuracy, then it is possible that the classifier identified all of the healthy samples correctly, but none of the unhealthy samples.
Binary classifier results can be presented in a confusion matrix [90] given in Table 2-2 . A confusion matrix is a result visualization tool typically used to measure the accuracy of classification models.

Table 2-2: Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. The values from the upper left of the matrix to the bottom right represent correctness, or "classified". This counts the positive samples predicted to be positive (TP) and negative samples predicted to be negative (TN). The opposite diagonal is considered "misclassified". These values represent the positive samples predicted as negative (FN) and negative samples predicted as positive (FP). The benchmark metric is accuracy which measures the overall correctness of a classifier:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad \textbf{(2-15)}$$

Accuracy provides a general evaluation of the classifier however in an imbalanced database, it does not accurately represent where the classifier fails.
Receiver operator characteristic (ROC) analysis is used to compare classification models or select possibly optimal models given different learning parameters [14]. The area under the ROC curve (AUC) measure has been found to exhibit the desirable property of describing balance in a model when compared to overall accuracy for machine learning algorithms [91]. It is commonly used in the case of imbalanced data where the class distributions are skewed [92]. The actual ROC is a plot of the false positive rate on the x-axis and the true positive rate on the y-axis. There are many ways to calculate the AUC [91]. However recent literature has determined that capturing a single point on the ROC is sufficient [93] and that it is appropriate for imbalanced data [94].
ROC analysis provides two measurements: "sensitivity" and "specificity". Sensitivity is the classification rate of positive samples. A positive sample in thesis examples is "fraud", "heart disease" and "employed". Sensitivity in the case of fraud is also called the "fraud detection rate". Specificity is the classification rate of negative samples; in these applications a negative sample is a fair trade, a healthy human, or unemployment. Using the variables in the confusion matrix,

$$Sensitivity = \frac{TP}{TP + FN} \quad \textbf{(2-16)}$$

$$Specificity = \frac{TN}{TN + FP} \quad \textbf{(2-17)}$$

$$AUC = \frac{Sensitivity + Specificity}{2} \quad \textbf{(2-18)}$$

One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another). The Type-I error is the misclassification of negative samples. It is also referred to as "false positive". The Type-II error is the misclassification of positive samples. It is also referred to as "false negative".

$$False\ Positive\ Rate = \frac{FP}{FP + TN} = 1 - Specificity \quad \textbf{(2-19)}$$

$$False\ Negative\ Rate = \frac{FN}{FN + TP} = 1 - Sensitivity \quad \textbf{(2-20)}$$

ROC analysis focuses on proportions of total positive samples and total negative samples to evaluate the classifier from a cost-benefit perspective of modeling. In contrast, accuracy is a proportion of correct positive and negative samples and is an evaluation of performance.

The value of using ROC analysis for imbalanced databases is best elucidated with an example. Let there be an imbalanced database with two classes in which 14% of the samples belong to the minority class and 86% of the samples belong to the majority class. The goal of the classifier is to identify as many of the minority class examples as possible (i.e. the benefit of the classifier) without misclassifying majority class samples as belonging to the minority class (i.e. the cost of the classifier). The example classifier results are shown in Table 2-3.

Table 2-3: Example confusion matrix

		Predicted	
		Positive	Negative
Actual	Positive	8	6
	Negative	6	80

Based on equation (2-15), the accuracy of the classifier is $\frac{88}{100} = 88\%$. This means it correctly classified 88% of the samples. The result is largely because of the classifier's ability to correctly identify the majority class samples and misleads one to think that the classifier is performing very well.

From ROC analysis we calculate sensitivity, the proportion of correctly classified positive samples. Note that the calculation uses values from the *row* of actual positive (top row) in the confusion matrix. In this example, $Sensitivity = \frac{8}{14} = 57\%$. The classifier correctly identified 57% of the minority class samples. The classifier specificity is the ability to correctly identify the samples belonging to the majority class. The calculations are derived from the *row* of actual negative (bottom row) in the confusion matrix. In this example, $Specificity = \frac{80}{86} = 93\%$. The classifier is very good at identifying the majority class.

AUC is therefore the average of sensitivity and specificity: $AUC = \frac{57\%+93\%}{2} = 75\%$. AUC is 13% lower than the accuracy; this is because sensitivity is specified as a quantity in the calculation. Because the classifier could identify 57% of the minority class correctly, it is not considered to be as "good" in terms of the AUC metric versus the accuracy metric. In fact, it is harder to get a "good" value in AUC versus accuracy in imbalanced databases. However, to use accuracy would be misrepresentative of the classifier's ability to correctly identify the minority class samples which is the entire point of the exercise.

From a cost-benefit perspective, the cost of the classifier is represented by false positive which is another metric to assess the classifier's ability to model the minority class. When false positive is high, it means that it cannot properly delineate between the minority and majority class. In this case, $FalsePositive = \frac{6}{86} = 7\%$ which is very good; the classifier does not spend a lot of time recognizing majority class samples as belonging to the minority class.

In order to apply these evaluation measures the researcher must know how many positive samples exist in the database. However, real-world settings do not always lend to full information about the system. But if someone is comparing methods they can all have the most expected frauds as the number of positive samples (in the denominator) and then all methods are treated fairly.

This thesis focuses on two measures: maximizing sensitivity and minimizing the false positive rate. The sensitivity and false positive rates can be viewed in a ROC graph [95]. It depicts the cost-benefit tradeoff for a discrete classifier. An example of an ROC graph from Fawcett [95] is in Figure 2-4. Generally, the most northwest points on the graph (e.g. point D) are the best classifiers because they have high sensitivity and low false positive. They are considered to be "conservative" classifiers because they make positive classification with strong evidence of few false positive errors.

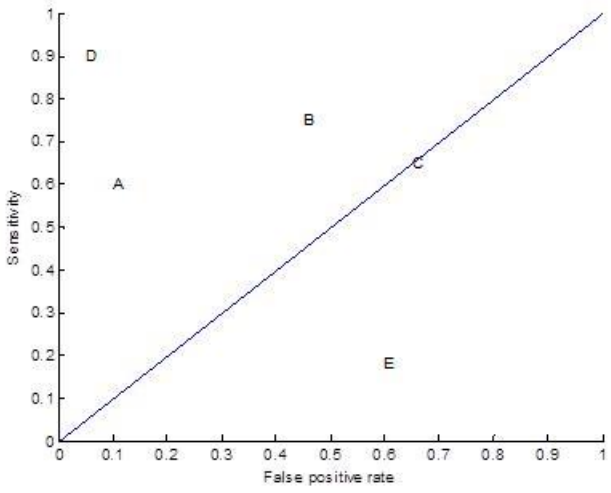

Figure 2-4: ROC curve for five discrete classifiers

Classifiers on the northeast side of an ROC graph (e.g. point B) are considered to be "liberal" because they make positive classifications with weak evidence and are often associated with high false positive rates. C lies on the diagonal line and represents random performance. Any points lying below the diagonal line perform worse than random guessing.

We now review and critic other measures accepted for evaluating imbalanced datasets in the literature ([14], [96]). The F-measure is composed of two concepts: *precision* and *recall*. Precision is the percentage of relevant objects identified for retrieval; recall is defined as the percentage of retrieved objects that are relevant.

$$Precision = \frac{TP}{TP + FP} \quad \textbf{(2-21)}$$

$$Recall = \frac{TP}{TP + FN} \quad \textbf{(2-22)}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad \textbf{(2-23)}$$

The F-measure is used when the learning objective is to achieve a balanced performance between identification rate (recall) and identification accuracy (precision). One drawback of the F-measure is that it neglects the correct classification of negative samples (specificity) and only reflects the importance of retrieval of positive examples (sensitivity) [93]. It does not account for misclassification whatsoever. Also, since it uses values from both rows in the confusion matrix, it is inherently sensitive to class skews [95].

Another metric for evaluation is G-mean. It is used when the learning objective is to balance identifications rates between positive class and negative class. It is calculated as:

$$G-mean=\sqrt{\frac{TN}{TN+FP}\times\frac{TP}{TP+FN}}=\sqrt{Sensitivity\times Specificity} \quad \textbf{(2-24)}$$

It also does not account for misclassification whatsoever.

2.10 Summary

In this chapter, relevant research articles, including a few different machine learning and statistical techniques applied to risky imbalanced databases, have been reviewed. Many similar methods are utilized to tackle both imbalanced and balanced databases; however they often require knowledge of the entire environment, only operate well when supervised, and rely on a lot of data pre-processing to balance the data. Moreover, most of these techniques cannot cope well in large-scale real-world problems. In the next chapter, we introduce improvements to RL techniques and a new Hidden Markov Model hybridization for imbalanced databases.

Chapter 3
State-space Machine Learning Algorithms for Classification

3.1 Introduction

In this chapter, the mathematical framework within which the database was modeled is introduced. The behavioural paradigm is described followed by a systems diagram of the three models presented by this research: RL Search, RL Classifier (RL) and RL-HMM hybrid. The RL section provides details on the steps used by the three models. Thereafter, three learning algorithms are presented along with the unique application of them within the RL framework to search for specific behaviours. Finally, the novel hybridization of a Hidden Markov Model (HMM) with RL and the classification method is proposed. This research takes an original approach to the definition of risk using two explicit behaviours: "collusion" and "repeated abuse" as defined in Chapter 1. Three behavioural patterns are modeled for the insider trading problem:

- Case 1: collusion - both the trader and the stock are fraudulent

  One trader executed many trades on one stock based on insider information.

- Case 2: repeated abuse - one trader, many stocks –the trader is fraudulent, not the stock

  One trader is executing many trades across several stocks and is repeatedly achieving higher than average profits. This is a case where the trader is risky, and is taking on risk for reward, independent of the stock.

- Case 3: repeated abuse - one stock, many traders - the stock is fraudulent, not the traders

Several traders are aware of insider information on one stock therefore they all trade that stock and achieve higher than average profits.

These three cases can be generalized. Consider a multi-variable system of $m = 1, \dots, M$ features in a database. Each feature can be expressed as δ_m discrete values (e.g. categorical survey data, binary signal, discretized risk levels) in the database. A combination of at least one value from feature m_1, and m_2 respectively can represent Case 1. Cases 2 and 3 can be represented by at least one value from any feature m appearing repeatedly in database records where a reward is higher than average.
In Section 2.8, three challenges to solving imbalanced databases were described: limited theoretical understanding, lack of publicly available benchmarks, and appropriate evaluation metrics. The challenge of understanding the problem was addressed in this thesis by specifying scenarios that focus on risk-reward relationships. Isolating imbalanced databases in this manner allowed for a system design that

targeted a theoretically distinct group of systems. The lack of publicly available benchmarks was addressed by selecting a diverse set of databases both public and private: financial fraud (private), heart disease (public), and female labour force participation (public). Finally, the classification algorithms were evaluated using metrics derived from ROC which assessed the accuracy of identifying both the minority and majority classes, namely sensitivity, and false positive.

3.2 Thesis models

Reinforcement learning (RL) was applied in two ways in this research. The RL agent always plays the role of the "risky" agent searching for higher than average rewards.

1. RL was used in its basic form as a search algorithm receiving informative feedback.
2. RL was used as a supervised classification algorithm that was trained on labeled data, validated to select a learning algorithm and tested on non-labeled data.

The former model will be called the "RL Search" algorithm hereafter and the latter will be called the Reinforcement Learning –Hidden Markov Model hybrid classifier ("RL-HMM hybrid"). A third model, the "RL Classifier" is a by-product of the RL-HMM hybrid model. The RL-HMM hybrid model has two forms of discretization, quantile (HyQ) and fuzzy (HyF) and will be referred to using these acronyms throughout the following sections

All models addressed the imbalance problem by using a learning agent to discover the patterns of the risky minority class and weight them higher than the majority class.

The RL Search algorithm appropriated the capability of "control" for which it is normally applied [97]. It revisited states with similar attributes and higher than average rewards. It was not trained; it was simply exposed to the entire database and received rewards as feedback to build the Q-table. Once the search was complete based on the number of records it visited, the Q-table was converted to a V-table. The V-table was sorted by weights and the top *c%* of states was considered to be positive. There were three search algorithms used for which results are shown: ε-greedy, optimal learning and Boltzmann learning.

The RL-HMM hybrid (HyQ, HyF) models began with training, validating and testing the RL agent. There were two outputs of the testing phase of RL: the first was the Q-values which were converted to a V-table and classified as in the RL Search algorithm. This model was the RL Classifier. The second output was several paths of rewards collected by the risky agent by taking optimal actions determined in the training phase. The risky agent's paths were subsequently modeled using the Markov property to

derive transition and emission matrices. The matrices were used to calculate posterior probabilities of risky states of which the top *c%* of states were considered to be positive. This model was the RL-HMM hybrid (HyQ, HyF).

The value for *c* was derived by testing several cutoff values and graphing sensitivity versus specificity. The value at which these two curves intersected was selected as the optimal cutoff value *c*.

The difference between the RL Classifier and RL Search models was that the former was trained on labeled data; the latter was not. Both were novel approaches to simulating the behaviour of a risky agent. Existing methods spend a lot of time manipulating the balance of majority and minority classes in the database. This requires a large amount of input data pre-processing which may be too slow for an industrial environment. Here the imbalanced problem was approached by using the learning agent to explore the data to find behaviour exhibited by minority class agents. A cutoff was then used to select the number of samples classified as positive.

A general representation of the three models is presented in Figure 3-1.

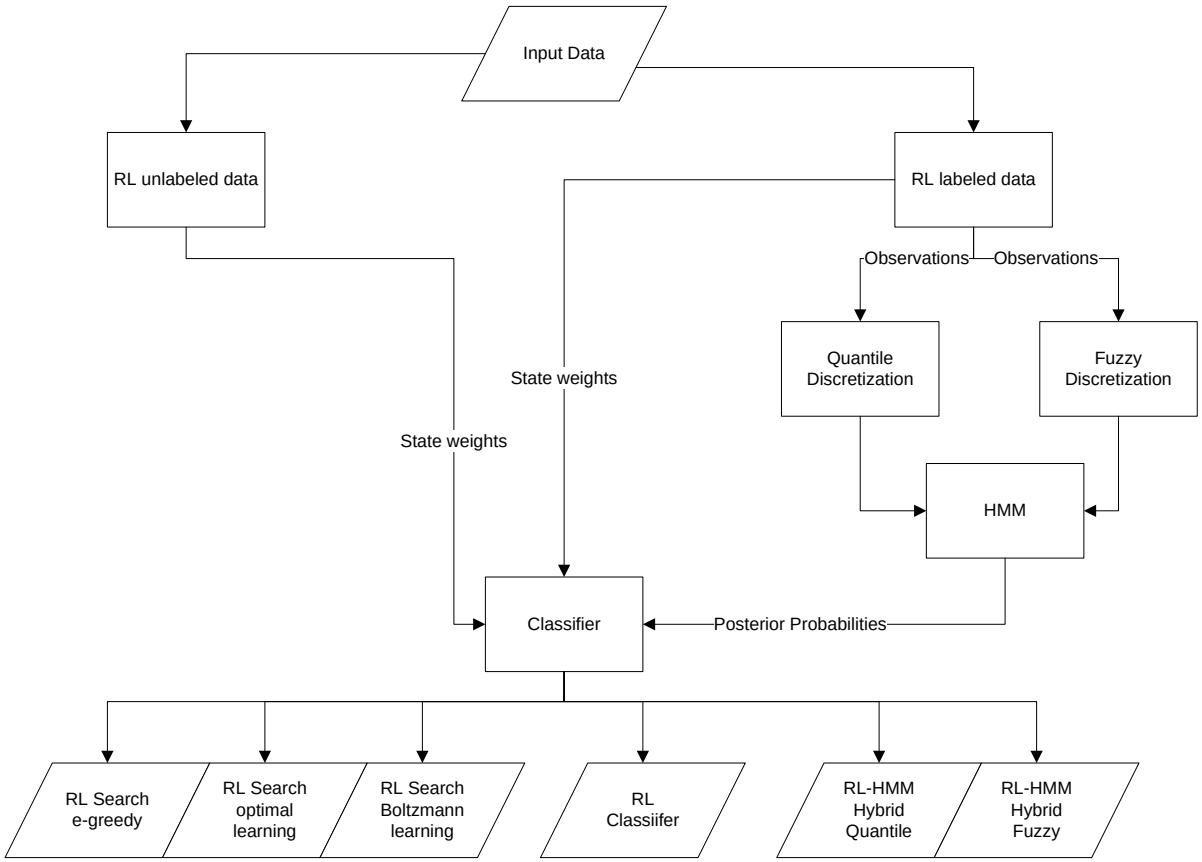

Figure 3-1: Classification algorithms

At the first stage the input data was cleaned and features were selected for class, reward and RL actions. RL then went through two separate processes. The path on the left of the diagram is the RL Search model. RL learned risky behaviour using three algorithms and Q-tables were calculated for each learning agent. The optimal Q-values (weightings on the states) were sent to the classifier to label states as positive and negative, and assigned labels to individual records. The output of the RL Search model is represented in Figure 3-1 as data modules. These are three vectors of class labels, one for each learning algorithm.

The path on the right begins with training RL on labeled data. After testing, the resultant optimal Q-values were sent to the classifier as above for the RL Classifier, the output of which was one vector containing class labels for each record.

The path on the right also outputted the rewards (observations) from the RL labeled data for the RL-HMM hybrid (HyQ, HyF) models. There were two separate discretization processes: quantiles and fuzzy. After the rewards were discretized they were sent separately to the Hidden Markov Model which assigned a probability of risk to each state. As with the previous two models the risk probabilities were sent to the classifier. The output of this stage was two vectors containing class labels for each record, one vector for each respective discretization model.

Overall there were six classification vectors output from the models in this thesis; three from the RL Search, one from RL Classifier and two from RL-HMM hybrid (HyQ, HyF).

3.3 Reinforcement Learning model

Reinforcement learning was selected to model human behaviour in risk-reward systems because:

1. It has been used in the past to model decision-making processes [41]
2. The decision-making agent is searching for a reward [31], an exact analogy to the systems modeled in this research
3. It is model-free; the lack of parameters allows for several behaviours to be modeled [41] across a diverse set of applications

3.3.1 State-space definition

Let us assume that the learning agent operated in a multi-variable state space. Each state was defined by the number of features in the database used in the search. For the insider trading problem the state was defined as $s_i = (trader_g, stock_l)$ for $i = 1,2,\dots,N$ states, $g = 1,2,\dots,\delta_1$ traders; and $l = 1,2,\dots,\delta_2$ stocks.

Case 1 (collusion) in the behavioural model was fully described by one state s_i, but many different cases of collusion can occur. In Case 2 (trader is fraudulent), the first variable in the state, $trader_g$, was

constant, but the second variable $stock_l$, could change. In Case 3 (stock is fraudulent) the second variable in the state, $stock_l$, was constant and the first variable $trader_g$ could change. Furthermore, many records were related to each state. This can be seen with a simple example in Table 3-1. The feature *trader* is assumed to take on two values (1,2) and the feature *stock* also takes on two values (1,2). The records are treated as "samples" in a probability space Ω; the table has 10 records. Profit represents the amount of money made or lost on a financial market trade. Class represents the binary classifier label. 1 represents a fair transaction, 2 represents a fraudulent transaction.

Table 3-1: Records in a database

Record#	trader	stock	profit	class
1	1	1	11.62	1
2	2	1	26.06	2
3	1	1	16.02	1
4	1	2	18.52	1
5	1	2	-12.34	1
6	1	2	11.89	1
7	2	2	21.56	1
8	2	2	23.83	1
9	2	1	29.59	2
10	1	1	-15.19	1

Each unique combination of feature values is abstracted to represent a "state". For the records in Table 3-1, the states are shown in Table 3-2. The states define the state space *S*.

Table 3-2: States

State#	trader	stock
1	1	1
2	1	2
3	2	1
4	2	2

State space *S* covers the entire sample (record) space. In this framework, information was collected from each sample in the database but was grouped by state. Operating in a state space had many advantages. It allowed for relating the samples through state transition and decreased the input data tracking space

because only states were monitored. It was efficient, concise and descriptive to use states. In a multi-variable state notation the insider trading problem of Table 3-2 is defined by 4 states: $s_1 = (1,1)$, $s_2 = (1,2)$, $s_3 = (2,1)$, $s_4 = (2,2)$.

In this example, both fraudulent records (record# 2,9 in Table 3-1) are defined by state $s_3 = (2,1)$ and is a fraudulent state representing Case 1, collusion (trader 2 and stock 1).

3.3.2 Action definition

The features of the database were used as "actions" in the RL algorithm. The algorithm could only arrive at a new state through one feature. Actions were the decision-variables moving the learning agent from state to state.

In a system with *M* features, a state is defined by multiple variables, e.g. $s_i = (m_1, m_2, \dots m_M)$ where m represents each distinct feature. There were *M* actions in action set *A* that could be taken in states $i = 1, \dots, N$ where action A_j corresponded to selecting feature m_j for $j = 1, \dots, M$ features.

The insider trading problem of Table 3-1 is a system with $N = 4$ states, $M = 2$ features $(trader, stock)$. There are two actions in action set *A*, $trader$ and $stock$.

3.3.3 Link analysis

The records in a database were linked by features and grouped by states. The table was searched by linking the same feature values occurring in two states. For example, in Table 3-2 States 1 and 3 can be linked via the state variable = *stock* because the feature value for both is defined by *stock* = 1. Therefore when in State 1, if action=*stock* then the agent moves to a record where the feature value for *stock* = 1. That record is then translated into a state for calculations. Incidentally, States 1 and 3 cannot be linked via the action= *trader* because they have different feature values for this state variable. At the extreme, an agent in state 1 can never get to state 4 directly. Figure 3-2 is a state transition diagram which describes possible agent moves for the insider trading example. State transitions to itself are always possible in the model but are not shown in the figure.

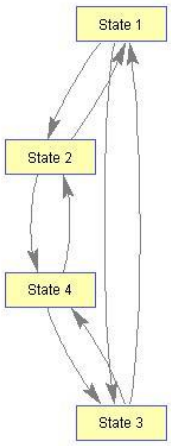

Figure 3-2: Possible state transitions in the insider trading problem

The concept of linking states together is not new to the fraud detection literature. Link analysis can relate known fraudsters to other individuals using record linkage and social network methods. These methods are popular in telecommunications fraud in which it is recognized that "fraudsters seldom work in isolation from each other" [98]. This theme was extended in this thesis by applying it to risky behaviour; fraudsters do not commit fraud once [17]. They are repeated abusers individually in addition to acting in collusion with other fraudsters and fraudulent assets.

To demonstrate how the link analysis works, examine the first 4 records for the insider trading problem in Table 3-1 and represent them in Table 3-3. The first column lists both the record number and state number as described in Table 3-1 and Table 3-2. Columns 2 through 4 are the same as in Table 3-1.

Table 3-3: Link analysis using 4 records

Record# /State#	trader	stock	Profit
1/1	1	1	11.62
2/3	2	1	26.06
3/1	1	1	16.02
4/2	1	2	18.52

When the current record# is 1, the current state# is 1. Let action = *trader*. State 1 has a value = 1 for action=*trader*. Therefore the system can move to another record where *trader* = 1. It therefore has the options to move to records 3 and 4. If record # 3 is selected, the next state is #1. If record #4 is selected, the next state is #2.

Lu's method of transitioning [41] and action definition is the same as described above. The difference between his paper and this thesis is that he considers records = states. He treats each record separately, which enlarges the calculation space. This research recasted and summarized the database records into states. The fundamental difference of defining states in this thesis allowed for the use of the Bellman Action-Value equation, BAV, to calculate Q-values which were interpreted as state weightings by the classifier.

3.3.4 Reward

Rewards were collected from each record and attributed to the state to which that record belonged. The reward was used to update the Q-value for that state through the BAV in equation (2-1). The approach was ideal for collusive behaviour because Q-values store sample information [99]. That is the difference between machine learning and statistical learning; machine learning employs BAV to summarize information while statistical methods require all of the data all of the time and uses basic summary statistics such as the mean of a random variable. Using the variables in equation (2-1) and the previous example, the following shows the BAV calculations:

- Current state is 1
- the model moves to record #4 (state 2)
- next action = *stock*
- initialize Q-value =0
- learning rate = 0.1,
- gamma =1 (episodic task)
- *t=1* for the first iteration through the database

$$Q^t(i,a) = Q^t(i,a) + \alpha[r_{ij}^t(a) + \gamma Q^t(j,b) - Q^t(i,a)]$$

$$Q^1(1,1) = Q^1(1,1) + \alpha\left[r_{1,2}^1(1) + \gamma Q^1(2,2) - Q^1(1,1)\right]$$

$$Q^1(1,1) = 0 + 0.1[11.62 + 1(0) - 0]$$

$$Q^1(1,1) = 1.162$$

For Table 3-2 the learning agent would move to state #2, the current action =*stock*, and calculations would be for $Q^1(2,2)$. The Q-table for the insider trading problem is described by the variables in Table 3-4.

Table 3-4: Q-table for insider trading problem

	Action 1 = trader	Action 2 = stock
State 1	$Q(1,1)$	$Q(1,2)$
State 2	$Q(2,1)$	$Q(2,2)$
State 3	$Q(3,1)$	$Q(3,2)$
State 4	$Q(4,1)$	$Q(4,2)$

The Q-table looks like Table 3-5 when the search is complete.

Table 3-5: Q-table for insider trading problem using ε-greedy learning

E-greedy	Action 1 = trader	Action 2 = stock
State 1	1.1620	0.5670
State 2	0	0.1200
State 3	8.0743	1.7487
State 4	0.6000	0

In all applications, the reward was weighted by the number of visits to that state. This was because if one state was repeatedly visited, then it was an example of collusion or repeated abuse and therefore must stand-out from the other states. When a state was found to have larger rewards than the others, it was considered "riskier".

Ultimately, a maximizing function was applied to the Q-table resulting in the V-table to obtain the maximum Q-values for each state, shown in Table 3-6.

Table 3-6: V-table for insider trading problem using ε-greedy learning

ε-greedy	Maximum Q-value
State 1	1.1620
State 2	0.1200
State 3	8.0743
State 4	0.6000

The values in the V-table correspond to the hidden behaviours generating each state. In State 3, the fraudulent state, the maximum Q-value corresponds to action= *trader*. This could be an example of Case

#1 and/or Case #2. Knowledge of the hidden behaviour that leads to the risky weighting on the state is a clue or starting point for human fraud investigators.

3.3.5 Learning algorithms

Action selection was the decision-making process in the model. The role of decision-maker was played by the agent searching the database which was driven by the learning model. Generally, the agent moved through the model using the following steps in Table 3-7:

Table 3-7: General search algorithm

General search algorithm	
1	current state = i, current action = a,
2	get reward $r_{i,j}(a)$ resulting from state-action pair (i, a)
3	move to state j
4	select action b using decision-making rules
5	calculate Q-value via the Bellman Action-Value equation using $Q(i, a)$ and $Q(j, b)$
6	convert new variables to current variables: $j \rightarrow i$, $b \rightarrow a$
7	go to 2

It is at Step 4 in Table 3-7 that each learning algorithm demonstrated its specific decision-making capability. The three learning algorithms used to uncover risky behaviour were: ε –greedy, optimal learning and Boltzmann learning.

All learning algorithms were initialized during training with the first record, the corresponding state to that record, and a random action.

The ε -greedy learning is a balance between Q-learning and Sarsa [31]. Q-learning selects the action for which the Q-value is largest in the current state. Sarsa randomly selects an action. The ε -greedy condition is invoked each time the model arrives at a state:

```
if rand< ε, Sarsa; else, Q-learning
```

For example, in Table 3-4, if the model is in State 1 and rand> ε, then the action a would be selected by the condition: $max_a(Q(1, a))$.

Each application had an optimal value for ε; the model explored (Sarsa) the database $\varepsilon\%$ of the time, and exploited (Q-learning) the database $(100-\varepsilon)\%$ of the time. When $\varepsilon=0$ the model never converged because it was ONLY exploiting the database. The ε -greedy algorithm is shown in Table 3-8.

Table 3-8: ε -greedy learning algorithm - general

ε -greedy learning algorithm – general	
1	current state = i, current action = a,
2	get reward $r_{i,j}(a)$ resulting from state-action pair (i, a)
3	move to state j
4	generate random number (rand)
5	if rand< ε, assign a random action b for that state; else, select the action b for which the maximum Q-value exists for that state
6	calculate Q-value via the Bellman Action-Value equation using $Q(i, a)$ and $Q(j, b)$
7	convert new variables to current variables: $j \rightarrow i$, $b \rightarrow a$
8	go to 2

The agent's behaviour is best demonstrated using a state transition diagram as in Figure 3-3. The diagram shows the direction of transitions between states, and indicates the agent's preferred states.

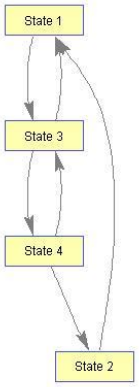

Figure 3-3: ε-greedy learning

The ε -greedy agent is a mix of exploration and exploitation. It explores the database but does not necessarily return to states. For example, it goes from State 2 to State 1 but does not return to State 2. The same behaviour is observed between State 2 and State 4. State 3 is visited from both possible states, 1 and 4, and has bi-directional arcs between those states. This shows that the agent navigates to State 3 frequently which is the only fraudulent state in this scenario.

Optimal learning is a statistical method implemented to take advantage of the adaptive statistical method. The algorithm estimated the mean of the reward received by the agent when it selected an action. In theory it was looking for the optimal action to take at each step to achieve the highest

reward. Testing showed that optimal learning selected the same action repeatedly, likely because it was originally designed for estimating one random variable. It was applied in this research to estimate multiple random variables, therefore it was modified by a random component to balance exploitation and exploration in a manner similar to ε -greedy learning. Consequently the agent exploited the states using an optimal learner by selecting the action that had the highest mean reward; it also explored the states randomly based on ε. This method of decision-making for RL was novel and a contribution to knowledge from this research. The algorithm is shown in Table 3-9; the state transition diagram is in Figure 3-4.

Table 3-9: Optimal learning algorithm

Optimal learning algorithm	
1	current state = i, current action = a,
2	get reward $r_{i,j}(a)$ resulting from state-action pair (i, a)
3	move to state j
4	update theta and beta based on $r_{i,j}(a)$ and current action = a
	if rand< ε, assign a random action b for that state; else, select the action b for which theta is the highest ($max_a(\theta^a)$)
6	calculate Q-value via the Bellman Action-Value equation using $Q(i, a)$ and $Q(j, b)$
7	convert new variables to current variables: $j \rightarrow i$, $b \rightarrow a$
8	go to 2

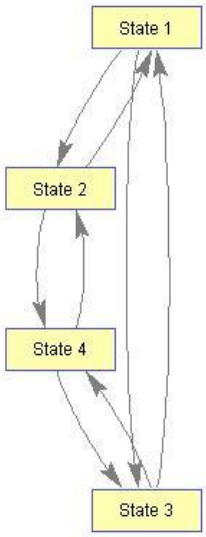

Figure 3-4: Optimal learning

The optimal learning agent visits all states from all other states; it has a bi-directional relationship with all possible states. It is the most explorative and exploitative learning algorithm in the suite of algorithms applied in this thesis. The difference between it and the ε-greedy algorithm is the value that is being maximized for the action-selection; optimal learning selects the action with the highest mean average reward whereas ε-greedy selects the action with the highest value in the current state.

Boltzmann learning is a process of estimating action probabilities. They were all initialized to be the same, $p(a) = \frac{1}{A}$. Because the initial probabilities are the same and actions were selected using a *max* function, the model was biased to action 1. This was because the probability for the current action (not all actions) was updated so when the *max* function was applied, the current action was selected again. This resulted in an exponential probability of action behaviour. As a consequence, the learning agent became stuck in local optimums. However, it had the potential to perform well in an imbalanced database because of this same property.

The most important parameter in the Boltzmann exploration learning algorithm is the temperature, τ. A genetic algorithm was applied to optimize this parameter for the application. Unfortunately it took too long and was rendered inefficient for this model. Therefore the parameters for training and validation were fixed to τ= number of records*2, and τ was decreased by $\rho = 2$ on each iteration after testing of several fixed parameters. The Boltzmann learning algorithm is shown in Table 3-10.

Table 3-10: Boltzmann learning algorithm

Boltzmann learning algorithm	
	initialize τ = number of records*2, $\rho = 2$
1	current state = i, current action = a,
2	get reward $r_{i,j}(a)$ resulting from state-action pair (i, a)
3	move to state j
	calculate $e^{Q(i,a)/\tau}$
	calculate $\sum_{z \in \lvert A(i) \rvert} e^{Q(i,z)/\tau}$
4	calculate $P(a) = \frac{e^{Q(i,a)/\tau}}{\sum_{z \in \lvert A(i) \rvert} e^{Q(i,z)/\tau}}$ for action a
	select action b for which the probability is the highest $(max_a(P(a))$
6	calculate Q-value via the Bellman Action-Value equation using $Q(i, a)$ and $Q(j, b)$
7	convert new variables to current variables: $j \rightarrow i$, $b \rightarrow a$
8	go to 2

In Figure 3-5 the state transition diagram for the Boltzmann agent is shown. It visits fewer states (notice the absence of State 4?) because it is exploitative.

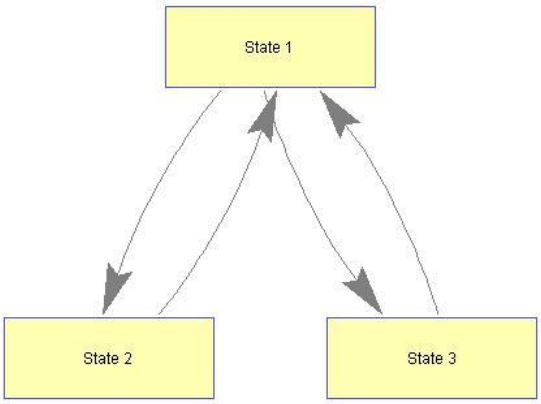

Figure 3-5: Boltzmann learning

The three learning algorithms and their observed properties are in Table 3-11.

Table 3-11: Comparing learning algorithms

Learning algorithm	Decision-making condition	General Observations	Q-table
ε -greedy	If rand< ε, select a random action, else, select the action with the highest Q-value	ε -greedy struggles to converge on small datasets; it remains in the local optimal states and does not explore further. It is challenging to converge to an optimal policy because the Q-values for all other states = 0.	Balanced
Optimal learning	If rand< ε, select a random action, else, select the action with the highest mean reward	Optimal learning balances between exploitation and exploration and visits more states.	Least sparse

Boltzmann	$P(a) = \frac{e^{Q(i,a)/\tau}}{\sum_{b \in \lvert A(i) \rvert} e^{Q(i,b)/\tau}}$	Boltzmann is exploitative; it uses the same action repeatedly.	Most sparse

In some datasets there were states that did not share feature values with any other states. This resulted in a local optimal state which was not necessarily the state for which the agent was searching.
The type of learning agent selected depended on the behaviour most suitable to the problem. Generally, in a database where there were few positive samples, ε-greedy and optimal learning were better to use because they explored the database searching for the rare samples. When the problem was to search for one positive sample Boltzmann quickly found and exploited it.

3.3.6 Deriving an optimal policy

Optimality in RL hinges off of Bellman's "Principle of Optimality" [100]. In an optimal policy, the sequence of decisions is optimal no matter the state in which the agent begins. An optimal policy was derived for each of the three learning algorithms via a training/validation/testing paradigm for the RL Classifier and RL-HMM hybrid (HyQ, HyF) model.
Learning occurred during the *training* phase and used a training dataset to obtain the optimal decisions (actions) to be taken at every state. These optimal actions demonstrated the behaviour used to construct a predictive relationship in the optimal policy. The reward in the training phase was the true class assignment (column 5 in Table 3-1); this was the supervised learning phase of the algorithm.
All of the learning algorithms collected rewards and processed them through the Bellman action value equation to calculate Q-values, $Q(S^n)$ (Table 3-5). The optimal value for each state was given when the Q-values were maximized for each state resulting in a V-table (Table 3-6), an example of which is shown in the second column of Table 3-12.

Table 3-12: V-table for the insider trading problem ε-greedy learning

State	V-table = max(Q-table)	Corresponding optimal action
1	1.1620	1
2	0.1200	2
3	8.0743	1
4	0.6000	1

On each iteration through the database, the resulting optimal values $V_{\pi'}$ were compared to the optimal values generated by the previous learning iteration through the training dataset, V_{π}. Optimality was calculated based on ζ:

$$\zeta = V_{\pi'}(S^t) > V_{\pi}(S^t) \quad \textbf{(3-1)}$$

for all states after t observations.

The optimal stopping criterion was based on a convergence value, *conv*.

$$\zeta \geq conv * N \quad \textbf{(3-2)}$$

When *conv*=0.7, it meant that at least 70% of the states were visited and assigned a weight. If ζ did not meet the optimal stopping criterion, the learning agent went through another episode of searching. The convergence constraint was implemented to speed up the processing time. In the process of filling the Q-table, the weights on the significant states increased faster than the others and converged. However, it did not mean that the search space had been covered. Therefore the search had to be constrained to ensure a sufficient number of states were visited to produce good results.

An optimal policy was the set of actions associated with the maximal values for each state once the algorithm converges. An example of an optimal policy is in column 3 of Table 3-12.

In the case of the insider trading problem, the weights represent the stocks and traders involved in frequent high returns relative to other stocks and traders. This deviation is calculated by the temporal differencing calculation of the BAV. The deviation becomes apparent in the differencing of state in the equation: $Q(j,b) - Q(i,a)$. This is also where the distinction in action emerges, that is, the action resulting in the highest reward. After training, the optimal actions for each algorithm are shown in Table 3-13.

Table 3-13: Optimal policy for all learning algorithms

State	ε-greedy	Optimal learning	Boltzmann learning
1	1	2	1
2	2	1	1
3	1	2	1
4	1	1	1

Boltzmann was the most exploitative learning algorithm in contrast to optimal learning which was the most balanced decision-maker.

The next step was *validation* to select the optimal learning agent. In this stage, the optimal policy was assessed on the training set using the rewards in column 4 of Table 3-1. The policy was then evaluated as a classifier using a confusion matrix. *c*% of heavily weighted states were labeled as positive samples.

The remaining states were considered to be negative samples. Since the ground truth vector used in the classification scheme assigned class by record, the assumption was that if a state is fraudulent, then all records belonging to that state are fraudulent too.

The best learning algorithm was selected using "correct rate" which was calculated as the sum of the diagonal of the confusion matrix. Values on the diagonal are correctly predicted samples. The correct rate was selected during the validation phase to identify the agent with the best ability to differentiate between positive and negative samples. In contrast, if the learning agent was selected based on sensitivity, then it was possible to have a high false positive rate as well. Therefore the best all-around learner was promoted to the testing phase along with its optimal policy.

A *testing* data set was independent of the training data; it was used to evaluate the quality of the classifier and its optimal policy. It was an off-policy method because the actions were not variable; they were fixed by the optimal policy derived from the most correct learner during the validation phase. These optimal actions were decisions made in each state on a test database and Q-values were calculated.

It was possible that states existed in the testing data that did not exist in the training data. Therefore no optimal policy existed for those states. In this scenario, the model defaulted to the original learning algorithm decision making process. If this scenario arose during initialization, the model selected a random action because there was no information collected yet to inform the decision maker.

This completed the learning agent's work. It was trained, validated and tested.

For the RL Classifier, the output was a V-table sorted by Q-values in which *c*% of states and their corresponding samples were labeled as positive.

For the RL-HMM hybrid (HyQ, HyF) model, the output of the testing phase was a path of rewards collected by the agent. These rewards were considered to be "observations" made by the agent as it conducted an optimal search through the database. The next step was to uncover the states that generated the optimal rewards. These states were then flagged as fraudulent. The fraudulent states were uncovered by a Hidden Markov Model.

3.4 Hidden Markov Model

The decision to include a Hidden Markov Model (HMM) component was guided by the following observations:

1. The RL Classifier took several hours to achieve acceptable results for an industrial setting.
2. The RL Classifier required several thousand records to achieve acceptable results.
3. The RL Classifier resulted in one sequence of state transitions, or paths. Further research was conducted to find a method of generating multiple paths, possibly from different learning agents, to add variety and increase the probability of identifying the correct path.

The purpose of the HMM in the model was to accept the sequence of rewards collected by the RL learning agent and derive state probabilities, independent of time. The method assigned a probability to each state conditional on observations, *O*, and model $= (P, \Phi, \pi)$, the state transition probabilities, emissions matrix and initial state probabilities. States with higher probabilities were considered to represent positive samples.

One important design issue addressed in the HMM was that of a left-to-right topology or a fully connected topology [49]. As shown in Chapter 2, the left-to-right topology is time dependent and the model advances through states in time. Wu's study of collusion [99], for example, uses HMM. They used the Viterbi algorithm to find the most likely state sequence which were time-dependent. For risk applications, this research assumed that "when" the risky state was visited was less important than the "frequency" of visits to that state. Because of this condition, the HMM was redesigned to calculate state probabilities independent of time while still using a left-to-right topology.

Other design issues brought forth by [49] are optimal number of states, type of observations and optimal number of observations. These were addressed by assuming:

1. Use only those unique rows as states in the training and testing databases.
2. Discretize the reward vector (the observations) using quantiles or a fuzzy logic model.
3. Limit the observation symbols to three so as to describe risk in terms of three levels: low, medium and high.

Each of these assumptions ensured an efficient model covering the entire database with a concise result.

3.4.1 Observation vector discretization

The rewards, which are now considered to be "observations" in the HMM context, were continuous values. Using the values as-is would retain the information however it created a large and sparse emission matrix due to a large number of observations with disparate values. Therefore the observation path data was discretized prior to input into HMM. Discretization occurred twice: the first was before calculating the emissions matrix. The second time was for calculating the path probabilities because they were based on the emissions matrix. As described in Chapter 2, two methods were used for discretization: quantiles (hard) and fuzzy logic model (soft). The mechanics of each follows.

The model used quantile discretization by bucketing data into three risk buckets representing, low (1), medium (2) and high (3) risk. The selected quantiles were *Q(0.5)* and *Q(0.9)*. The low bucket contained all values that fell below or were equal to the 50^{th} quartile, in other words, below the mean of the data. Medium risk was between *Q(0.5)* and *Q(0.9)*. The high risk bucket contained values above and including the 90^{th} quartile.

In the trading data in Table 3-1, the third column represents the observations of the system which are trading profits and losses. The observations before and after discretization are shown in Figure 3-6. The upper graph is continuous. The bottom graph shows the data after discretization: 5 observations are in the low risk bucket, 4 are medium risk and 1 observation is high risk.

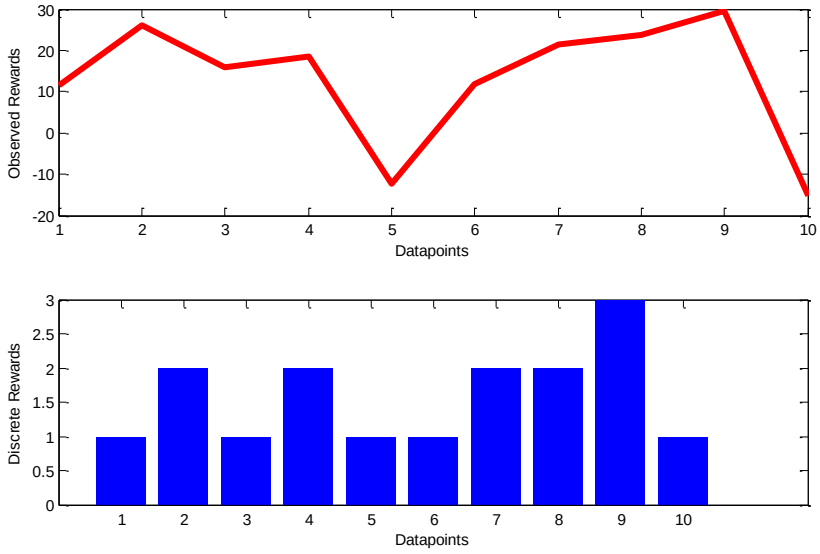

Figure 3-6: Quantile discretization

The fuzzy data model was a one input – one output Mamdani model [101]. A Mamdani model was selected so that both the input and output of the FIS could be modeled by membership functions.
Let X={x} be a set of points. In this example X represents the observations collected by the learning agent. Let A be the fuzzy set representing the input data. Let B represent the fuzzy set representing the output data. Then fuzzy set A in X is characterized by membership function $f_A(x)$ which maps each point in X to the interval [0 1]. After testing many membership functions, those resulting in the highest standard deviation in output values were selected for discretization: Gaussian membership function for $f_A(x)$, and triangular for $f_B(x)$. Because it is only a single-input, single-output model, a T-norm operator was not required. To defuzzify the data from the triangular function to the crisp output value, the centroid method was used and the values were rounded to [1,2,3] to represent each risk level.
One of the constraints on the FIS is that the parameters need to be modified for every database. The main practice in the literature is to train the parameters when they are entered into the system which takes a lot of time and is not the focus of the thesis. Therefore, partially adaptable constraints were set for $f_A(x)$.
The input range for $f_A(x)$ is [min(x), max(x)].

```
r1=[min(vect(:,i)) max(vect(:,i))];%for entire range
r2=[0.15*max(vect(:,i)) -10];% lo
```

```
r3=[0.15*max(vect(:,i)) 0.5*max(vect(:,i))]; %med
r4=[0.15*max(vect(:,i)) max(vect(:,i))]; %high
```

The output parameters could remain the same each time for $f_B(x)$ because the data is purposely restricted to 3 categories. Therefore the output range is [0,3].

```
a=addvar(a,'output','risk',[0 3]);
a=addmf(a,'output',1,'lo','trimf',[-1.2 0 1.2]);
a=addmf(a,'output',1,'med','trimf',[0.3 1.5 2.7]);
a=addmf(a,'output',1,'high','trimf',[1.8 3 4.2]);
```

The three corresponding rules in this FIS were:

```
If A is low then B is low
else
if A is medium then B is medium
else
If A is high then B is high
```

The two membership functions are shown in Figure 3-7.

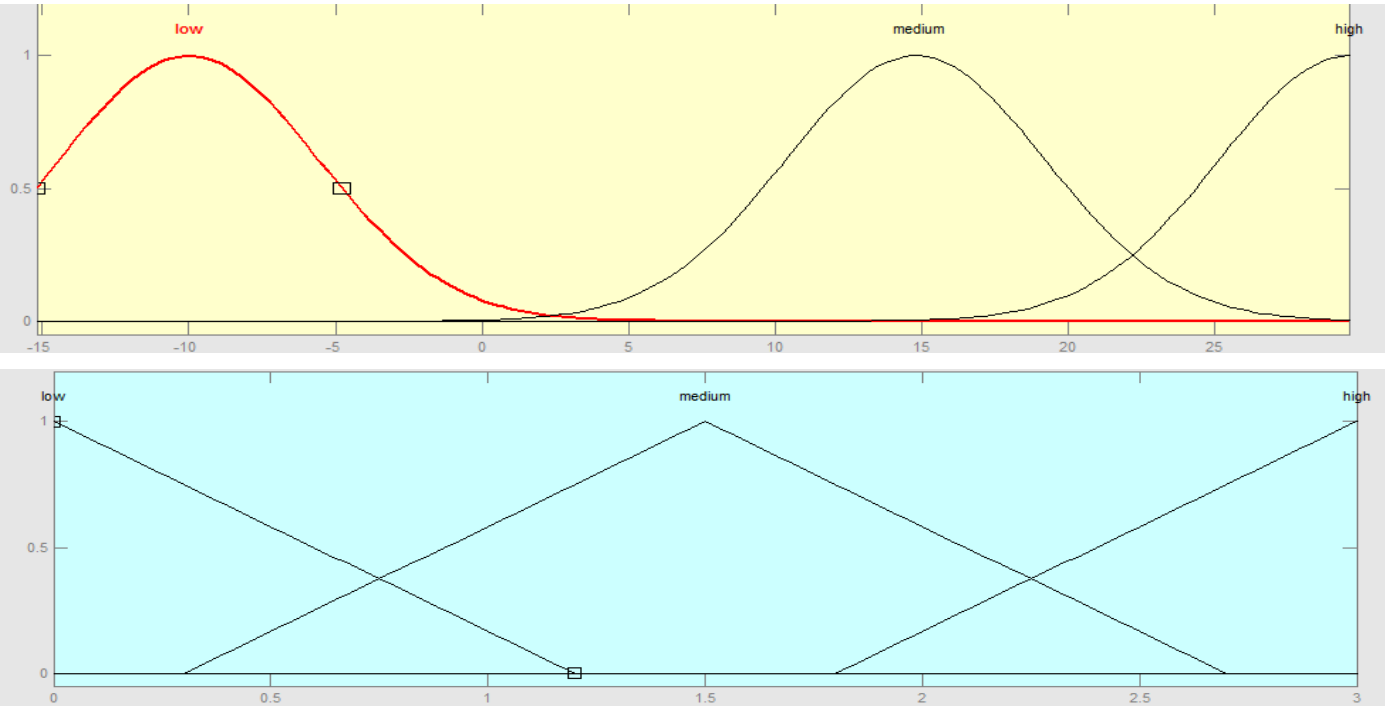

Figure 3-7: Single-input Mamdani fuzzy model membership functions and input-output curve

The "high" input space is the most narrow to compensate for the imbalanced nature of the research problems. The resulting output curve is shown in Figure 3-8.

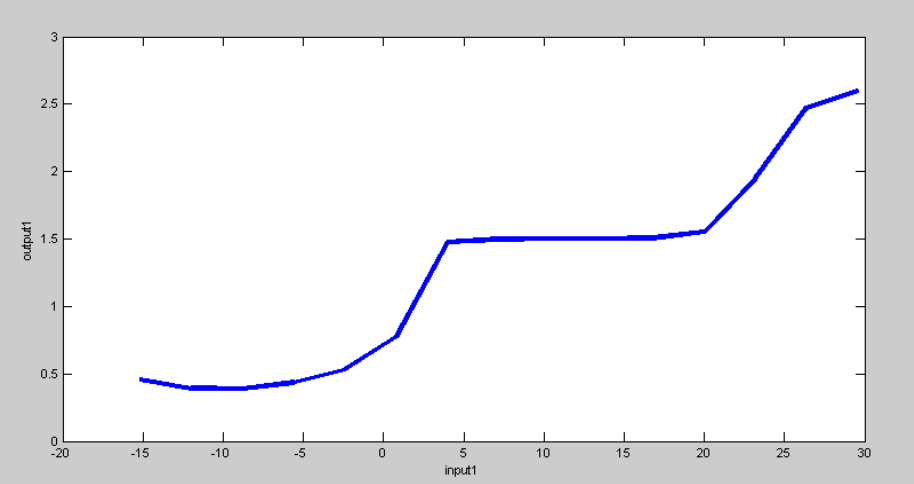

Figure 3-8: Output curve for Mamdani FIS

The nature of imbalance is also reflected in the output curve in where most samples are mapped to the majority classes, low and medium risk.

Figure 3-9 shows the entire sample dataset before and after fuzzy discretization. The upper graph shows the raw data. In the lower graph, the green line represents the crisp fuzzy output. It was thereafter rounded to obtain [1, 2, 3] and is represented by the blue bars.

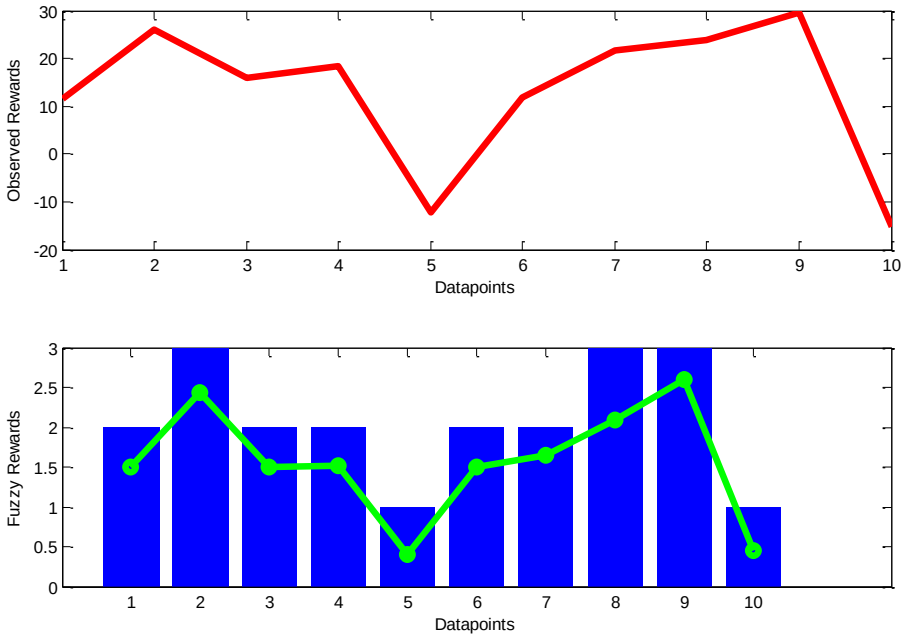

Figure 3-9: Fuzzy discretization

3.4.2 HMM Assumptions

The three main assumptions for HMM were described in Chapter 2. They were satisfied in this research as follows:

1. Markov Assumption

   The RL agent moved to a next state based on the current state during the learning process.

2. Stationarity assumption.

   The RL agent's search was independent of time. The learning algorithm operated within a simulation environment therefore transitions could occur between states at any step in the process.

3. Observation independence assumption

   Statistical independence of observations was assumed in this research based on two characteristics of the system:

   i. The data used in the thesis was generated empirically by human behaviour so there were no underlying statistical assumptions about the data.

   ii. The learning agent used "action-selection" to select observations within the machine learning paradigm versus drawing samples from a distribution. Therefore observations were considered to be independent insofar as the selection of an observation in one step did not influence the observation collected in the next step.

3.4.3 Mathematical formulation

The final output of the RL algorithm for RL-HMM hybrid (HyQ, HyF) was a path of rewards collected from an agent moving through states making optimal decisions. Optimality in this case refers to maximizing risk and rewards. *K* reward paths were generated by the learning agent. The observations were used to estimate the transition probability matrix (TM), Emissions matrix (EM), and the marginal probabilities (π) in the HMM. The following section describes the probabilistic framework used to derive TM, EM and π.

The probabilistic framework is described using the data in Table 3-1, referring to feature = trader as "C" and feature=stock as "T".

There are 20 points in the probability space representing observable input variables collected from a system. In this example there are 4 events; feature C=1 (C_1), feature C=2 (C_2), feature T=1 (T_1), feature T=2 (T_2). The probability space with 4 events is shown in a Venn diagram in Figure 3-10.

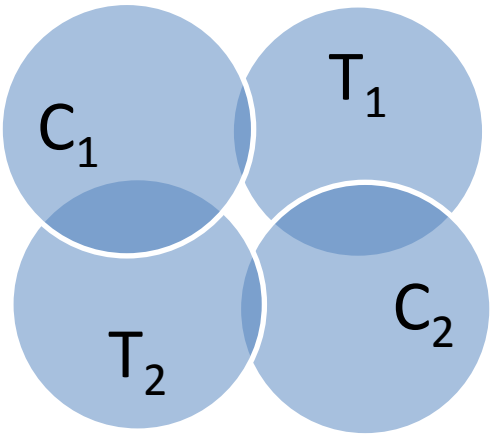

Figure 3-10: Sample Venn diagram

Based on the values in Table 3-1, the event probabilities are calculated in Table 3-14.

Table 3-14: Probability of feature value occurring

Feature and Feature value	Probability
$P(C_1)$	6/20
$P(C_2)$	4/20
$P(T_1)$	5/20
$P(T_2)$	5/20

The overlap in the Venn diagram represents the rows in the database, each of which is an entire description of the sample. The probability for each pair is derived by counting the rows in Table 3-1 and shown in Table 3-15.

Table 3-15: Probability of pairs of values occurring in the table

Feature and Feature value	Probability
$P(C_1)P(T_1)$	3/10
$P(C_1)\ P(T_2)$	3/10
$P(C_2)P(T_1)$	2/10
$P(C_2)\ P(T_2)$	2/10

Each unique combination of feature values is abstracted to represent a "state" through which the intelligent agent can search to uncover hidden information. For the records in Table 3-1, the states are shown in Table 3-2.

The probability of each state occurring can be derived from the Venn diagram and the probabilities in Table 3-16. For a 2-dimensional state-space definition there are $i = 1,2,\dots,N$ states; feature values $g = 1,2,\dots,\delta_1$ for feature 1; and feature values $l = 1,2,\dots,\delta_2$ for feature 2. The probability of a state is an OR combination because the state is arrived at through one feature value at a time.

$$P(S_i) = P(C_g \cup T_l) = P(C_g) + P(T_l) - P(C_g)P(T_l) \quad \textbf{(3-3)}$$

Plugging in the values from Table 3-14 and Table 3-15 into equation (3-3), the state probabilities are shown in Table 3-16.

Table 3-16: State probabilities

Feature value	Probability
$P(S_1)$	(6/20)+(5/20)-(3/10) = 0.25
$P(S_2)$	(6/20)+(5/20)-(3/10) = 0.25
$P(S_3)$	(4/20)+(5/20)-(2/10) = 0.25
$P(S_4)$	(4/20)+(5/20)-(2/10) = 0.25

Feature selection is translated as "actions" to the search agent. For two features, there are two actions, A_1 and A_2. A_1 corresponds to selecting feature C, A_2 corresponds to selecting feature T.

When an agent is selecting a state to go into, it is determined by an action. If the agent is in record 1 it is in state 1. When the action is A_1, the agent is looking for another record where C=1 (records 3, 4, 5, 6, 10). When the agent arrives at a new record, it is translated into the state-space described in Table 3-2. For this example, the agent is in state 1 and can move to state 1 or state 2. If A_2 is selected, then the agent is looking for another record where T=1 (records 1,2,3,9,10) translating into State 1 or 3.

The state transition diagram is described in Figure 3-11. The large ovals represent the states and contain the state names and their feature values. The arrows show which states the agent can transition to from the current state. The smaller ovals near the arrows show the action that can lead to the new state.

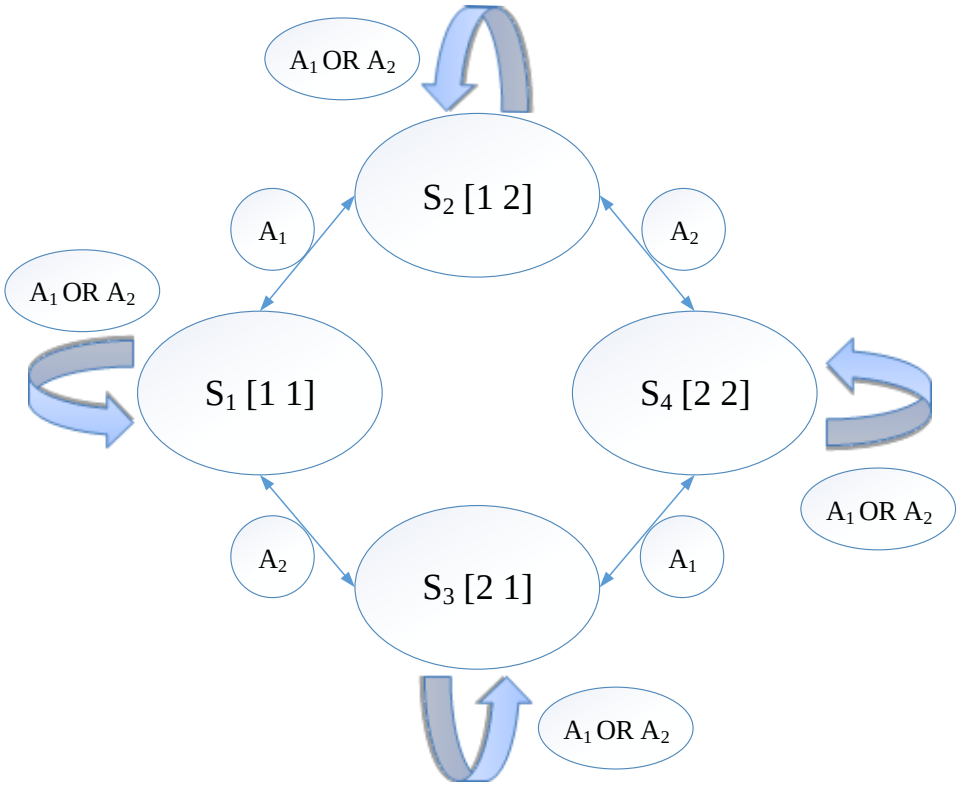

Figure 3-11: State transition diagram

The probability of selecting an action is determined from the learning algorithms converging to an optimal policy. The optimal policy defines the best action for each state that will return the highest reward.

The probability of selecting feature C is defined by $P(A_1)$; the probability of selecting feature T is defined by $P(A_2)$. These probabilities are determined empirically by observing the search agent. In this example, for explanation purposes, assume action probabilities: $P(A_1) = 0.4$; $P(A_2) = 0.6$. This means that the agent selects A_1 for 40% of the states, and A_2 for 60% of the states. Actions are independent; the agent can only select one at a time, and thus explore the database based on one feature at a time. Actions play a fundamental role in learning how to search the state space and must be included in the conditional probability calculations for congruency with the algorithm search engine.

The rule of Total Probability and Bayes' Theorem [32] [53] is invoked to incorporate action selection into the probabilistic calculations and calculate the TM.

The theorem states that a partition of a set A is a set $\{A_1, A_2, \ldots, A_M\}$ with three properties:

1. Set of subsets (probability space made up of subsets)
2. Subsets are mutually disjointed (independence)
3. All subsets are collectively exhaustive (they cover the entire probability space)

$$P(A_k|A) = \frac{P(A|A_k)P(A_k)}{\sum_{i=1}^{n} P(A|A_i)P(A_i)} \quad \textbf{(3-4)}$$

Define S as the set S made up of states $\{S_1, S_2, S_3, S_4\}$. They are independent, and cover the entire probability space (i.e. the space through which the agent searches). The purpose of this exercise is to derive the state transition probabilities e.g. $P(S_2|S_1)$; the probability of reaching state 2 given that the agent is in state 1. Because moving between states is based on the action taken, the conditional probability can be restated in terms of actions e.g. $P(S_2|S_1) = P(S_2|A_1)$. As shown in Figure 3-11, the agent can only reach state 2 from state 1 by selecting action 1. This theory is applied to update the probability of event S_i given the information A_k.

The definition of total probability can be applied to solve for arriving at a state based on an action.

$$P(S_i|A_k) = \frac{P(A_k|S_i)P(S_i)}{\sum_{i=1}^{n} P(A_k|S_i)P(S_i)} \quad \textbf{(3-5)}$$

Equation (3-5) can be expanded to obtain a more explicit version in equation (3-6).

$$P(S_2|A_1) = \frac{P(A_1|S_2)P(S_2)}{P(A_1|S_1)P(S_1) + P(A_1|S_2)P(S_2) + P(A_1|S_3)P(S_3) + P(A_1|S_4)P(S_4)} \quad \textbf{(3-6)}$$

To solve for (3-6) find $P(A_1|S_2)$ in equation (3-7).

$$P(A_1|S_2) = \frac{P(A_1 \cap S_2)}{P(S_2)} \quad \textbf{(3-7)}$$

Actions and states are independent because the probability of selecting one does not influence the probability of the other. Therefore equation (3-7) becomes:

$$P(A_1)$$

Because of independence the simplification of equation (3-7) works for all occurrences of where the probability of an action is conditional on a state. Equation (3-6) is simplified to:

$$P(S_2|A_1) = \frac{P(A_1)P(S_2)}{P(A_1)P(S_1) + P(A_1)P(S_2) + P(A_1)P(S_3) + P(A_1)P(S_4)} = \frac{0.4(0.25)}{4(0.4)(0.25)} = \frac{1}{4}$$

4 in the denominator represent the 4 states which are of equal probability in this example.

Because all of the states have equal probability the conditional probabilities of moving between states are all the same: ¼.

For self-same probability (the probability of remaining in the same state) the equation is as follows based on equation (3-5):

$$P(S_2|S_2) = P(S_2|A_2) + P(S_2|A_1) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

The calculations for all of the transition probabilities are in Table 3-17. The final transition matrix is in Table 3-18.

Table 3-17: Transition probability calculations

Transition state probability equation	Equivalent based on equation 3	Probability
$\boldsymbol{P(S_2|S_1)}$	$P(S_2|A_1)$	1/4
$\boldsymbol{P(S_2|S_2)}$	$P(S_2)$	1/2
$\boldsymbol{P(S_2|S_3)}$	0	0
$\boldsymbol{P(S_2|S_4)}$	$P(S_2|A_2)$	1/4
$\boldsymbol{P(S_3|S_1)}$	$P(S_3|A_1)$	1/4
$\boldsymbol{P(S_3|S_2)}$	0	0
$\boldsymbol{P(S_3|S_3)}$	$P(S_3)$	1/2
$\boldsymbol{P(S_3|S_4)}$	$P(S_3|A_1)$	1/4
$\boldsymbol{P(S_4|S_1)}$	0	0
$\boldsymbol{P(S_4|S_2)}$	$P(S_4|A_2)$	1/4
$\boldsymbol{P(S_4|S_3)}$	$P(S_4|A_1)$	1/4
$\boldsymbol{P(S_4|S_4)}$	$P(S_4)$	1/2
$\boldsymbol{P(S_1|S_1)}$	$P(S_1)$	1/2
$\boldsymbol{P(S_1|S_2)}$	$P(S_1|A_1)$	1/4
$\boldsymbol{P(S_1|S_3)}$	$P(S_1|A_2)$	1/4
$\boldsymbol{P(S_1|S_4)}$	0	0

The transition probabilities are expressed in a matrix in Table 3-18.

Table 3-18: Transition probability matrix

	S_1	S_2	S_3	S_4
S_1	1/2	1/4	1/4	0
S_2	1/4	1/2	0	1/4
S_3	1/4	0	1/2	1/4
S_4	0	1/4	1/4	1/2

The resulting TM is a matrix of size [N x N]. The transition matrix is different for every path generated by the RL model because the probability of actions $P(A_i) \forall i \in A$ changes with every RL learning agent. The emissions matrix is calculated empirically from the database from the reward path and states. The algorithm is in Table 3-19.

Table 3-19: Empirical calculation of emissions matrix

1	Discretize all possible emissions (hard and soft discretization method)
2	For each state, calculate the number of times each emission occurs
3	Calculate the percentage of times each emission occurs for each state as a percentage of total emissions from that state

The resulting EM is a matrix of size [N x T]. It is static within one application because it is calculated from all possible rewards, not just the rewards collected by the learning agent.
The marginal probabilities are calculated from the TM. The TM does not usually begin with full rank; there are sometimes dependent rows in the TM, for example, a row of zeros for unvisited states. The resulting matrix is of full rank after the marginal probability calculations and the system has a solution. Sometimes the TM is close to singular (noninvertible) due to its sparsity. In these cases it cannot calculate marginal probabilities and a new set of training data is selected for the problem.

3.4.4 HMM fundamental problems revisited

As discussed in Chapter 2.4.3 there are three fundamental problems that can be solved using HMM. All three problems are solved sequentially in this research to derive the state probabilities. Table 3-20 describes the solution methods applied in comparison to common solutions in the literature.

Table 3-20: Solving HMM fundamental problems

Type of problem	Common solution	Thesis
Learning	Baum-Welch	Reinforcement learning

Evaluation	Forward algorithm	Forward algorithm
Decoding	Viterbi algorithm	Posterior probabilities

The purpose of the learning problem is to train model parameters which can be time-consuming and not necessarily converge [32]. The RL algorithm was selected to overcome these constraints and take advantage of the properties of a learning agent in a model-free environment. In comparison to the commonly used, highly constrained Baum-Welch algorithm [32], [102], [66], [103], the only optimization condition on the RL learning agent is that it had to visit at least 70% of all states during training. The model parameters, $\lambda = (P, \Phi, \pi)$, are then calculated empirically from the observation path.

The evaluation problem was solved using the forward algorithm as described in Chapter 2. It is applied to each observation path generated by the RL algorithm. The solution of the evaluation problem, $P(O_k|\lambda)$, is a matrix of size [*K* x *1*]. This provides the probability of each path belonging to model λ. The most probable observation path of the *K* paths is then selected by solving $max_k\ P(O_k|\lambda)$.

The final step is the decoding problem. The goal is to find the optimal state sequence Q corresponding to the observation sequence O. It is commonly solved using the Viterbi algorithm [32] [54] which is dependent on time. Abusive and collusive behaviour can occur with frequency in small time frames, but are not necessarily sequential from an overall system point of view. A unique method of manipulating the observation probabilities to uncover state probabilities $P(Q|O, \lambda)$ independent of time was derived in this research.

Step 1: calculate the posterior probabilities, $P(Q_i|o_t, \lambda)$,

Step 2: isolate $P(Q_i)$ by

a) solving $P(O) = P(O_k|\lambda)\ x\ O_k$
b) solving $P(Q_i) = P(Q_i|o_t, \lambda) \cdot P(O)$

where $P(O)$ is the probability of an observation occurring at any step in the process.

The Viterbi algorithm hinges off of assigning probabilities to arcs in the trellis of states. The method used in this thesis considered the path insofar as it was required for posterior probability calculation. It then separated the posterior probabilities from time by multiplying the conditional posterior probability by the observation probability in step 2a.

The algorithm for the HMM procedure is shown in Table 3-21.

Table 3-21: Hidden Markov Model procedure

Step #	**Algorithm 2** Uncovering risky states in risk-reward system
1	Generate K paths of T observations using the RL agent
2	For the k^{th} observation path, $\forall$ N states in the testing database:
2-1	Calculate transition matrix (TM) based on the Markov Assumption
2-2	Calculate emissions matrix (EM) (only once)
2-3	Calculate marginal probabilities (π)
2-4	Preprocess observation path using statistical quantiles and fuzzy membership function
2-5	Assign a probability of belonging to model λ
3	Select the path with the highest probability
4	Calculate posterior probabilities
5	Calculate probability of each observation occurring at any step t
6	Calculate probability of each state occurring at any step t

3.5 Classification

Class labels were assigned by ordering the states by their probabilities from highest to lowest and labeling the top c% states as positive samples. The cutoff value was important to the quality of results; when c is large, then more states are considered to be positive samples. While this may "widen the net" cast for positive samples, it may also include many false positives, particularly in an imbalanced database where there are few positive samples. However, if c was too small, it missed some of the positive samples resulting in a lot of false negative results but worst of all, missing some risky states. Several cutoff values were tested and the resulting sensitivity was graphed versus specificity. The value at which these two curves intersected was selected as the optimal cutoff value.

The ground truth vector used in the classification scheme assigned class by record. The assumption is that if a state is fraudulent, then all records belonging to that state are fraudulent too. It is possible for a variable combination to be sometimes fair, and sometimes fraudulent in the real-world. However, assuming that every record described by a state is assigned the same class is congruent with the sought after behaviours of collusion and repeated abuse.

For evaluation, sensitivity, false positive and AUC are used because the purpose of the learning agent is to learn behaviours so it can correctly identify fraudulent samples. Misclassifying a sample as fraudulent means that it has not properly learned the behaviours. This is why we need to look at false positives: to

understand misclassification and hence, an in-depth understanding of the classifier's ability to model the minority class. A similar argument could be made about false negatives but since we already discuss sensitivity the quantity itself does not bring much value to the argument. Additionally, we use ROC graphs to discuss the results. ROC is referred to as a "cost-benefit tradeoff for a discrete classifier" graph [95]. "Cost" is referring to the cost of the classifier (e.g. time, energy), not the cost in economic terms. Notably, we do not calculate the *accuracy* metric because it does not appropriately reflect the classifier's effectiveness on imbalanced database.

The algorithm that processed the results from Table 3-21 for classification is described in Table 3-22.

Table 3-22: Classification steps

Step #	Algorithm 3 Classifying risky states in risk-reward system
1	The resultant vector is all of the states with a probability assigned to each one. We order the vector in ascending order and select a cutoff, *c*. The top *c%* samples are considered to be true positive, *(100-c)%* are considered to be fair.
2	Assign labels on each state to the corresponding records and evaluate.

3.6 Benchmark models

The K-means model was used to benchmark against the RL Search model because it is unsupervised. Four models were selected from the literature as benchmarks for the RL and RL-HMM hybrid (HyQ, HyF) classifiers: Logistic regression (LR), Artificial Neural Network (NN), Support Vector Machine (SVM) and K-nearest neighbour (KNN). All models were implemented using built-in Matlab functions [80]

LR and NN are probabilistic classifiers [95] and therefore output numeric values that represent the degree to which an instance is a member of a class. SVM and KNN are discrete classifiers and output only a class label. The outputs of the probabilistic classifiers were rounded to 0 and 1to match the output of the classifiers developed by this research.

Other modeling considerations were that the NN used for benchmarking had 10 hidden layers and is applied through the built-in Matlab function *nn_routine* [80]. For the KNN classification model, $k = 1$ which has shown best results in the literature [71].

3.7 Summary

In this chapter, we demonstrated application of RL techniques for data mining and hybridized them with a probability-constrained HMM to lower the samples required to speed up the learning process of RL techniques. An adapted statistical learning model, optimal learning, was introduced in a novel decision-making application to RL. We also presented a new methodology to find the state probabilities from a HMM model independent of time.

Machine learning models have been developed for imbalanced database applications to uncover risky behaviour. They were applied in both unsupervised and supervised environments.

In the next chapter, these techniques will be tested on various case studies.

Chapter 4
Experimental Results

4.1 Introduction

In the following sections, the RL Search, RL Classifier and RL-HMM hybrid (HyQ, HyF) Classifier are applied to four databases. They are as follows:

1. Insider trading: A synthetic database of equity trades generated from real-world scenarios.
2. Debit card fraud database (DCFD): A database generated from real-world numerical distributions of fraudulent and non-fraudulent debit card banking transaction data.
3. Canadian heart health database (CHHD): A Statistics Canada survey of Canadians to find the rate of heart disease.
4. Female labour force study (LFS) 2013: A Statistics Canada employment survey. The goal of this exercise is to detect female employment indirectly based on survey data. It is a balanced database.

Testing on the four databases will be shown for all algorithms devised in this thesis in four separate sections. In each section, we will initially present details about the database and the feature selection process. Next, a discussion about the RL parameters will be presented. Then the application of the unsupervised model, RL Search, will be presented on varying proportions of the data (10%-100%). The results for the RL Search algorithm will be evaluated in terms of sensitivity (i.e. the fraud detection rate) and false positive (i.e. the misclassification of the classifier) against the benchmark algorithm, K-means. We will also show the standard deviation of these metrics when RL Search is run over several iterations to demonstrate the robustness of the unsupervised learning algorithms. Standard deviation is calculated after many trials to evaluate consistency of the model. If standard deviation is high it means that the algorithm is highly dependent on the data in the training set. Otherwise, any deviation in the results is because of the proportionally smaller test set.

A summary of the database specifications is in Table 4-1.

Table 4-1: Summary of database specifications

Table name	Balance	Number of features	Positive sample	Number of samples in database	Rate of positive samples in database
Insider trading	Imbalanced	2	Insider trading fraud	5000	4.2%
Debit card fraud database	Imbalanced	2	Worthless Deposit Fraud	5001	7.2%
CHHD	Imbalanced	4	Heart disease	9035	11.52%
Labour Force Survey February 2013 (female samples only)	Balanced	3	Female labour force participation	24597	64%

Next, we will show the results for the supervised algorithms. The RL Classifier results will be shown calculated in tandem with the results of the RL-HMM hybrid (HyQ, HyF) classifier for crisp and fuzzy discretization. The RL Classifier and RL-HMM hybrid (HyQ, HyF) performance was evaluated using train/test sets of 90%/10% and 80%/20%. The results are presented in an ROC graph and compared to LR, NN, SVM, and KNN classifiers. Each section will conclude with a comparison of AUC for all unsupervised and supervised algorithms. AUC is the average of sensitivity and specificity, used to measure the overall effectiveness of the classifiers as discussed in Section 2.9. After the financial fraud applications we will explain why the supervised algorithms used a lower number of records than the unsupervised and were yet able to achieve nearly the same results. The chapter will conclude with a summary of the behaviour of the learning algorithms across the databases, and the AUC of proposed thesis algorithms for all four databases. A summary of models and notation is shown in Table 4-2.

Table 4-2: Model summary and notation for thesis algorithms results.

Type of learning	Model	Learning agent	Notation
Unsupervised	RL Search	e-greedy	e-greedy
Unsupervised	RL Search	Optimal learning	OptLearn
Unsupervised	RL Search	Boltzmann learning	Boltzmann
Unsupervised	K-means		Kmeans

Supervised	RL Classifier	Best learning algorithm	RL
Supervised	RL-HMM Hybrid	Best learning algorithm with quantile discretization	HyQ
Supervised	RL-HMM Hybrid	Best learning algorithm with fuzzy discretization	HyF
Supervised	Linear regression		LR
Supervised	Artificial neural network		NN
Supervised	Support vector machine		SVM
Supervised	KNN		KNN

A simulated insider trading problem is presented first to reveal the learning algorithms' ability to mimic described risky behaviours and demonstrate the mechanics of the classifiers. For this application intermediate calculation and results will be shown for the reader's understanding.

4.2 Insider trading

The problem of insider trading was described by three cases in Chapter 3. There were two steps to generating the data:

1. Generate random data
2. Generate patterns

The first step was performed using a random assignment of values. Random values from 1-250 were assigned to the first feature (trader); random values from 251-500 were assigned to the second feature (stock). There existed 500 input values in the data. The reward was generated by taking the absolute value of the "randn" function in Matlab which takes samples from the normal distribution.

There were 212 records and 16 states in the fraudulent pattern, distributed into three cases. The number of records and states are in brackets beside each case (records; states). For example, the overall (records; states) for fraudulent patterns is (212; 16).

When Attribute A = "trader" and Attribute B = "stock" then there are 250 traders booking *N* trades on 250 stocks. The three fraudulent patterns are:

- Case 1: collusion - both the trader and the stock are fraudulent (81; 6)
- Case 2: one trader, many stocks –the trader is fraudulent, not the stock (56; 4)
- Case 3: one stock, many traders - the stock is fraudulent, not the traders (75; 6)

Case 1 has 3 pairs of collusion; there are three scenarios where one trader executed many trades on one stock based on insider information. Altogether, 81 trades were executed by 3 traders on 2 stocks which

are summarized into 6 states. Case 2 has one trader executing 56 trades on 4 stocks, summarized into 4 states. Case 3 has 6 traders executing 75 trades on 1 stock, summarized into 6 states.
Algorithmically, the data was generated by:

1) Random data was produced for a total of *N*-212 records
2) The 212 pattern records were randomly added to the dataset
3) The data was divided into training and testing sets

As the number *N* records increased, the number of fraudulent cases was held constant. The fraudulent data was effectively "buried" deeper in the data with each increase in records. The corresponding good (fair trades) versus bad (fraudulent trades) percentage values are shown in Table 4-3.

Table 4-3: Insider trading characteristics of the data sets

Records	# Good	# Bad	%Good - %Bad
1000 records	788	212	78.8-21.2
2000 records	1788	212	89.4-10.6
3000 records	2788	212	92.9-7.1
4000 records	3788	212	94.7-5.3
5000 records	4788	212	95.8-4.2

4.2.1 Database

For *N*=1000 records, the data can be seen in Figure 4-1. The larger, red circles represent the fraudulent records. The smaller, blue circles represent the non-fraudulent records. 21.2% of the records are fraudulent in this case.

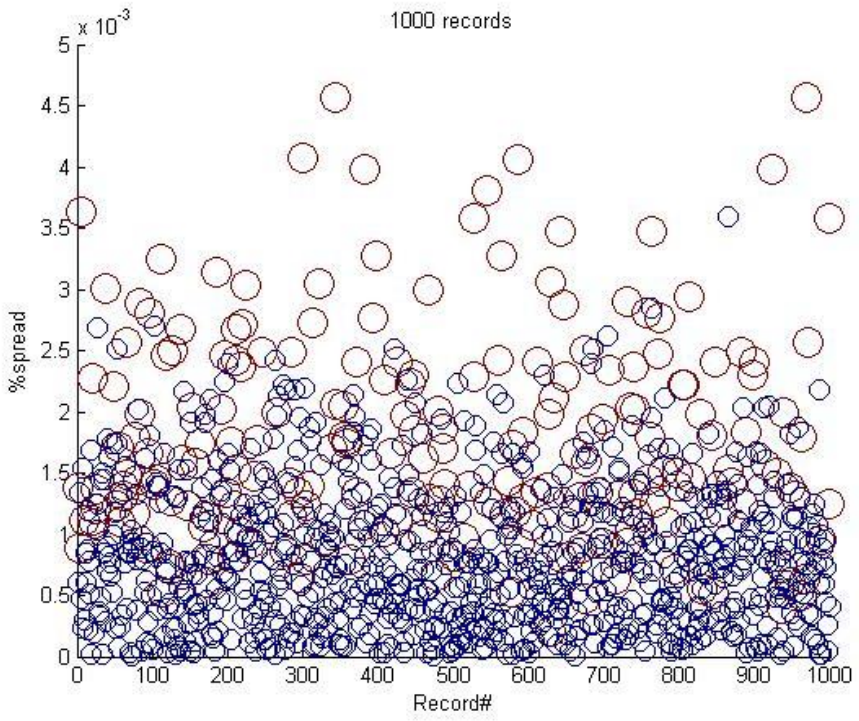

Figure 4-1: Insider trading reward distribution for a database of 1000 records.

4.2.2 Reinforcement Learning parameters

The RL algorithm parameters are:

1. $\{S\}$: $\{Trader\ 1-250, Stock\ 251-500)\}$
2. $\{A\}$: $\{trader, stock\}$
3. $R(S,A)$: $profit\ or\ loss\ on\ trade\ as\ a\ percentage\ of\ total\ profit\ in\ the\ database$

There are 62,500 possible states in the database (i.e every trader and every stock = 250^2). However, all stocks were not traded by all traders and vice-versa therefore the state-space is much smaller than 62,500. For example, in the database of 1000 insider trading records there were 883 states altogether.

4.2.3 RL Search Experimental Results

The unsupervised learning results are presented first to demonstrate the behaviour of a learning agent permitted to explore and exploit the database without any prior knowledge. This procedure is analogous to running a learning machine in the background during a business day to collect information about trading behaviour. Because this database was generated, results are shown for 5 databases with 1000, 2000, 3000, 4000, 5000 trades.

- The number of records to be sampled = x% of N*2 (2 times the number of records in the database) where x is the proportion of the database to which the learning agent is exposed (10-100).
- The cutoff, c, is 20%. That means 20% of all states were classified as fraudulent
- For the exploratory component of the learning algorithms, ε= 0.8 which means that the learning agent explores 80% of the time.
- $\alpha = 0.1$. This means the agent learns slowly and only takes 1% of the reward value each iteration

The RL Search algorithm is an unsupervised search through the database where decisions are made based on the learning algorithms' conditions. Sample maximum Q-values are shown in Figure 4-2.

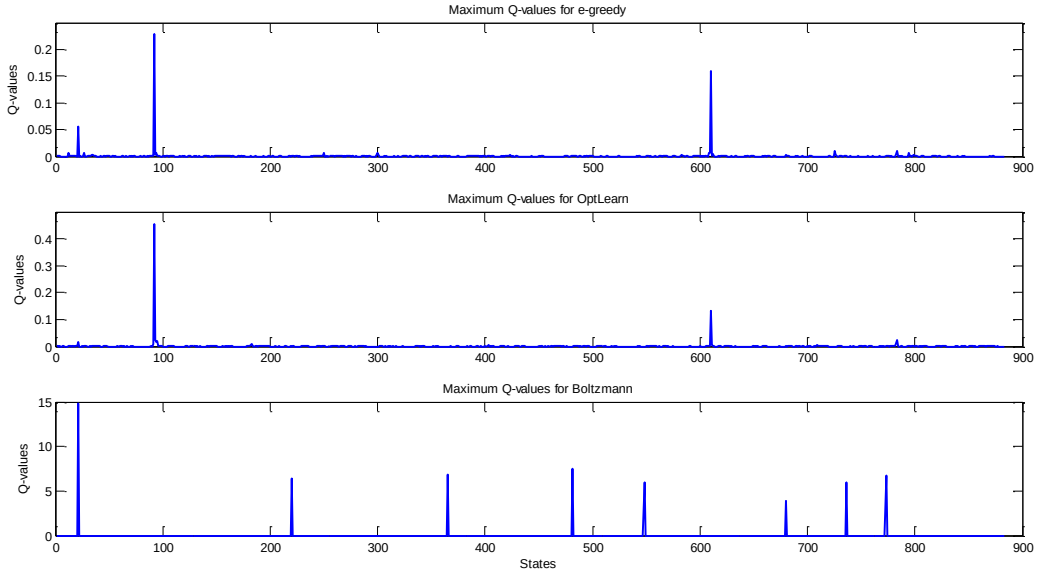

Figure 4-2: Insider trading RL Search paths

State 92 and State 610 are fraudulent collusive transactions found by ε-greedy and OptLearn agents. Boltzmann visits fewer states but has much higher Q-values. One of the fraudulent collusive states it finds is at state 21 (the Q-value = 15) which Boltzmann revisits frequently. To observe the gradual performance of the RL Search algorithm the ROC is presented in Figure 4-3.

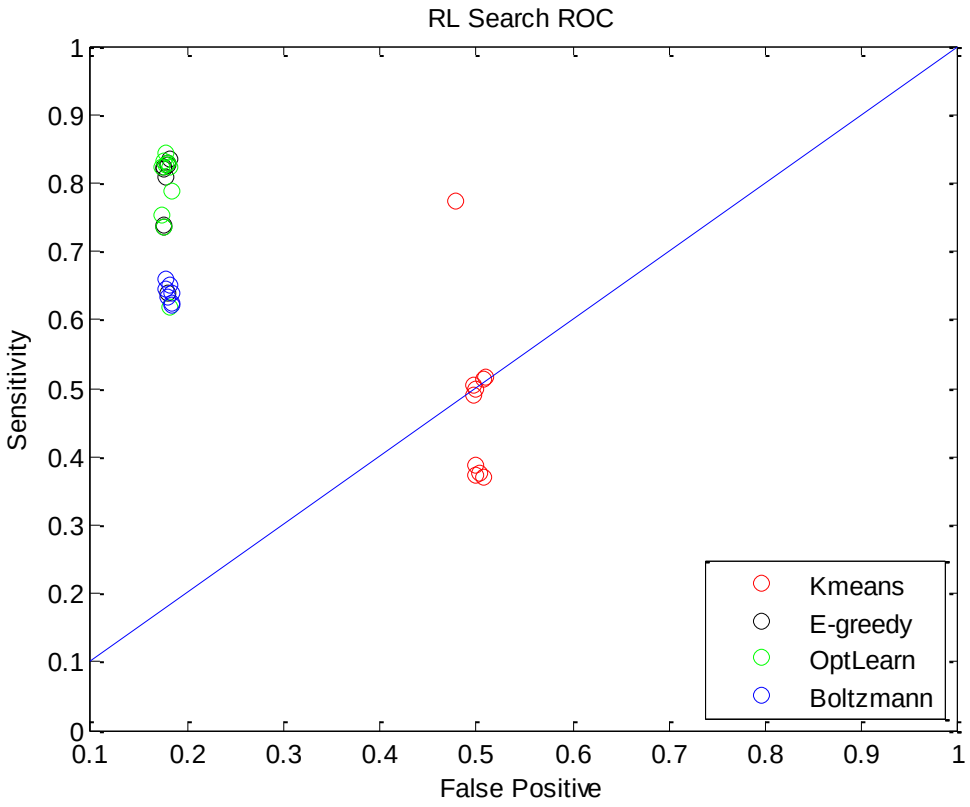

Figure 4-3: Insider trading ROC for RL Search

The ROC diagram shows values for testing percentages from 10-100%. Each circle represents the sensitivity vs. false positive value for each testing percentage; there are 10 circles for each algorithm with the 10% values being the most southern circles on the ROC graph for each respective algorithm. Southern circles represent lower sensitivity because, for an imbalanced database, it is likely that the smaller portions of the data do not contain any positive samples. All algorithms begin to migrate to the northern part of the ROC as the testing percentages increase, reflecting the ability to better identify fraud with more samples. For example, the green circles are migrating northward in the graph to reflect the increasing sensitivity of the optimal learning algorithm. In contrast, the red circles for Kmeans begin in the southern half of the graph, migrate to the "random" point in the graph (as described in Section 2.6 as being on the diagonal line) and then one point, 100% exposure shows a high sensitivity (~75%) but maintains a higher false positive rate (~50%). The increased exposure does not appear to improve results for K-means until it is exposed to 100% of the database. Overall, results become increasingly better for RL Search as it sees more of the database.

The following graphs summarize the sensitivity (Figure 4-4), false positive (Figure 4-5) and AUC (Figure 4-6) rates for runs on 1000, 2000, 3000, 4000, 5000 insider trading records over 10%-100% of the

database. On the x-axis the record size is represented by (_#) beside the algorithm name where # is the number of thousands of records. The different colours on the bars represent the testing percentages. A discussion of the results will follow the figures.

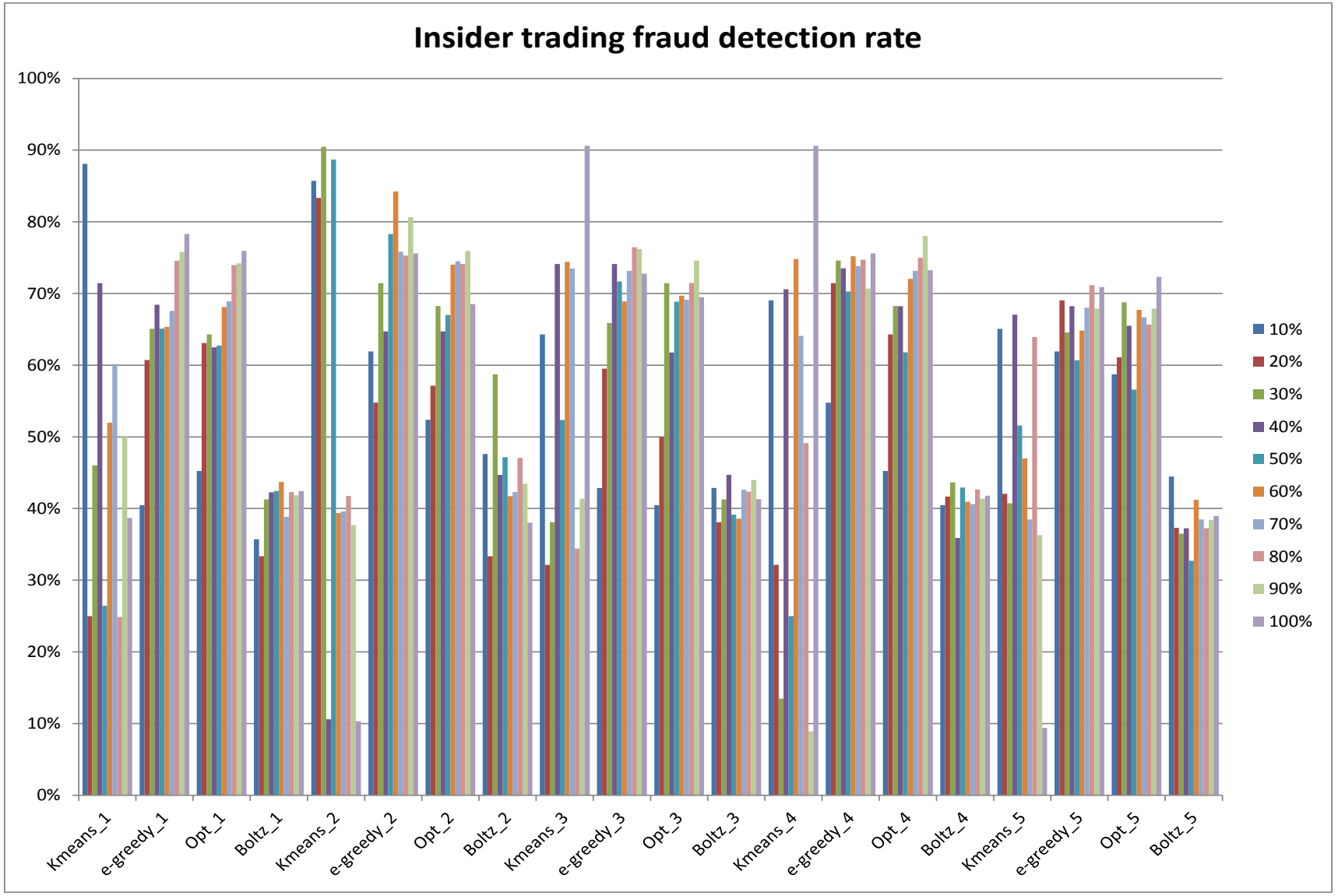

Figure 4-4: Insider trading sensitivity rate for unsupervised algorithms across all testing percentages

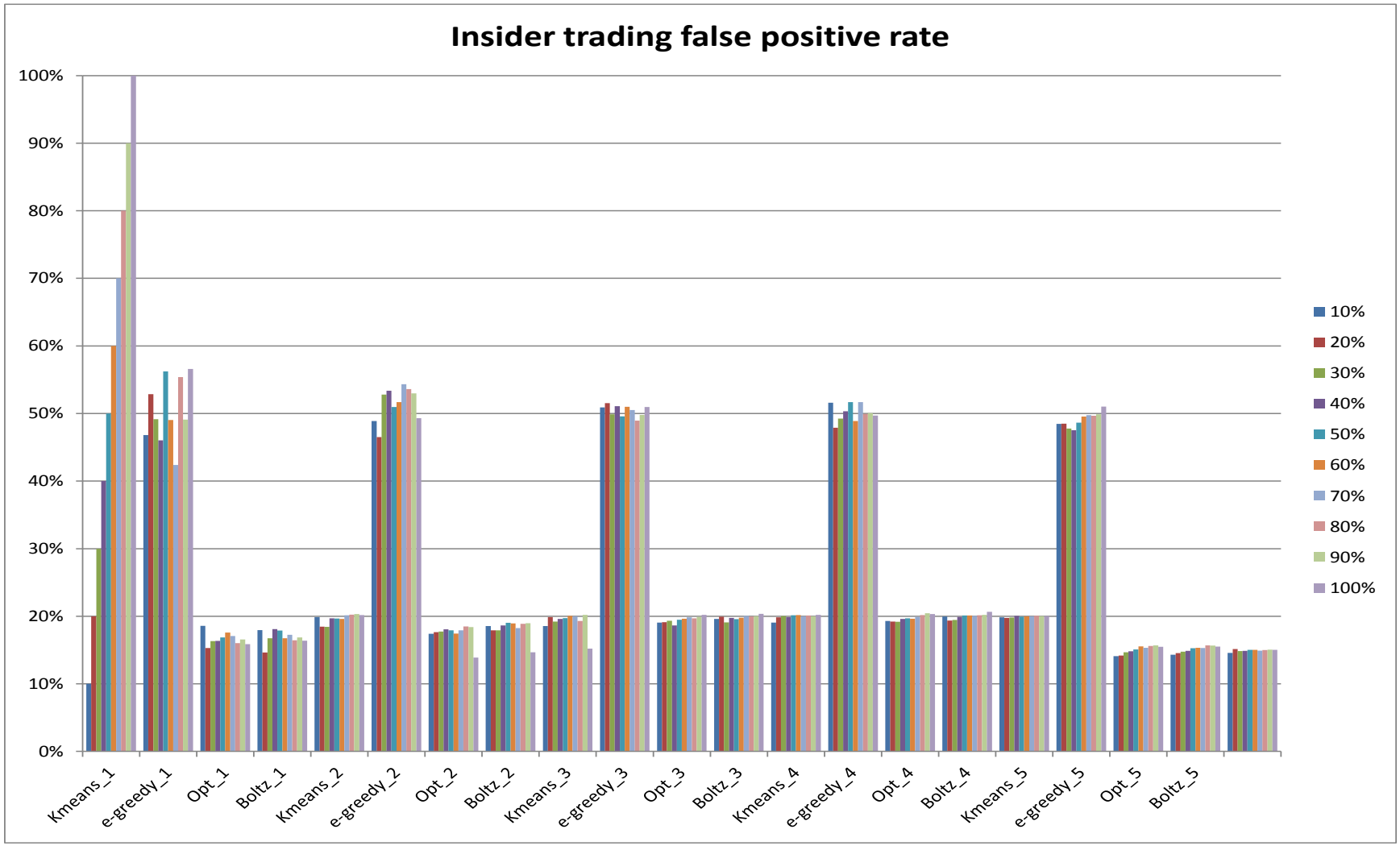

Figure 4-5: Insider trading false positive rate for unsupervised algorithms across all testing percentages

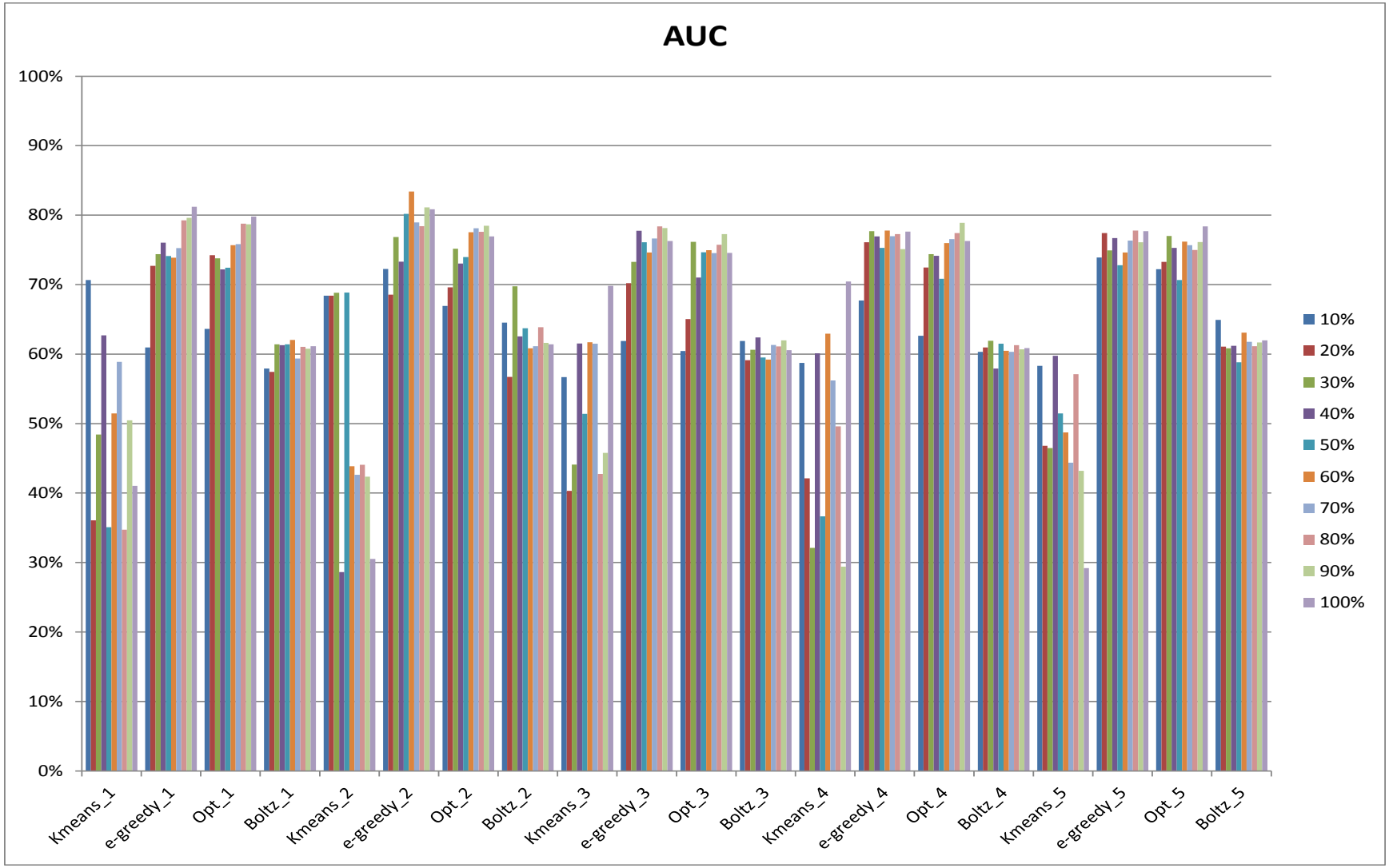

Figure 4-6: Insider trading AUC for unsupervised algorithms across all testing percentages

The RL Search ε-greedy learning agent is the best performing fraud detector. The AUC is near and excessive of 80% for varying percentages and sizes of the database. Kmeans appears to have the highest fraud detection rates but it also has the highest false positive rate. The results highlight the possibility that Kmeans cannot truly recognize the fraudulent trades. The Kmeans AUC is commensurate with this assessment by being less than 70% for all percentages of the database, for all sizes of the database.
As the number of records increase, so does the ability of ε-greedy and OptLearn to identify positive samples. These results support the use of RL for large databases. The combination of RL improvement when exposed to more data, and its ability to find fraud which is being buried deeper into the data proves the industrial capabilities of RL Search for financial fraud.
Interestingly, the lowest false positive rate for thesis algorithms is for the largest database, 5000 records. This means that even though the fraud is buried deeper into the data, the RL Search mechanisms can identify the related samples, the risky behavioural scenarios, and not mislabel them. This is the power of the link analysis and decision-making mechanisms in this research. The learning agents focus on the fraudulent scenarios and revisit them because of larger than average rewards and common feature values. This is the situation in true financial fraud databases therefore RL Search could work tremendously well for automatic fraud detection screening.
Consistency is necessary for any industrial application. The standard deviation of sensitivity and false positive rates are shown in Table 4-4. RL Search provides consistently low false positive rates meaning that the learning agents know the difference between fraudulent and fair trades. Kmeans has the highest variability in sensitivity which further proves the assertion that the performance of Kmeans is random and does not seem to have any systematic relationship with the number of records.

Table 4-4: Unsupervised learning insider trading standard deviation (%)

	Kmeans	ε-greedy	Optlearn	Boltzmann
Sensitivity	23.75	7.95	8.38	3.51
False Positive	2.09	0.75	0.70	0.55

4.2.4 RL Classifier and RL-HMM hybrid (HyQ, HyF)

The ROC is used to demonstrate the performance of thesis models in comparison to benchmark models for supervised learning. All model parameters are the same as for unsupervised learning except for the number of records which is only 500. The results are shown for datasets of 90%-training, 10%-testing (represented by 0.1) and 80%-training, 20%-testing (represented by 0.2). For the insider trading example, graphs exemplifying every step of the process will be shown for the reader's understanding.

During supervised training the reward is equal to the class. Figure 4-7 shows the optimal Q-values for each state for each learning algorithm for 80%-training, 20%-testing. Note that the state numbers are not necessarily the same as in the unsupervised graphs due to the random sample selection in the testing paradigm. ε-greedy and optimal learning emphasize a state just after the 0th state, and then around the 500th state. They both revisit several states during their searches. Boltzmann appears to visit a few states repeatedly around the 150th state.

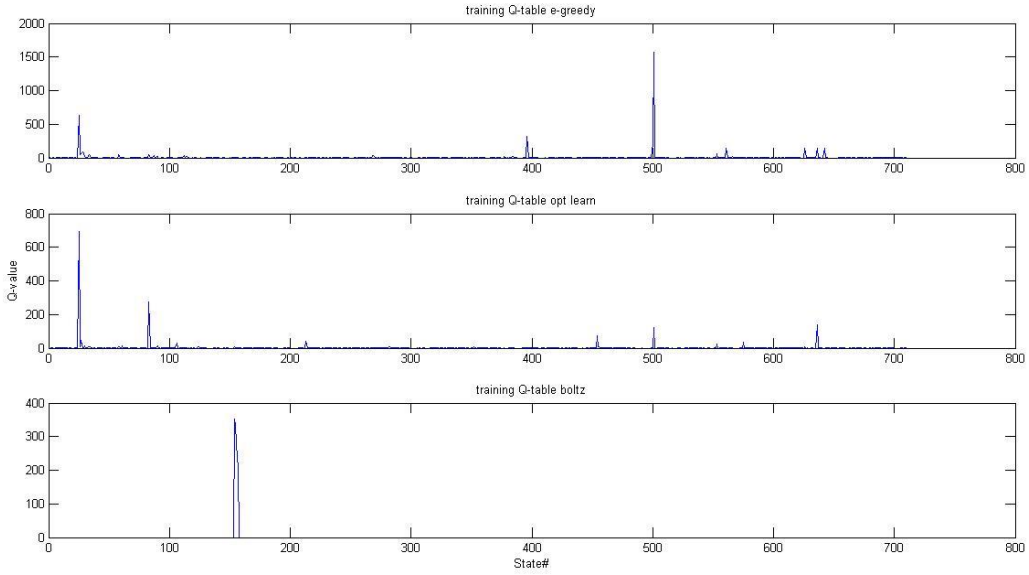

Figure 4-7: Q-tables for training data

Figure 4-8 shows the Q-values by state after applying the optimal policy to the training data during the validation stage. At the validation stage the reward is the reward profit or loss as a percentage of total profit of all trades in the database. In Figure 4-8 ε-greedy places particular emphasis on the 500th state plus some states around the 100th state. Optimal learning revisits a state just after 0 repeatedly, and a few others but not as much. The Boltzmann learning agent continues to revisit a few states around the 150th state.

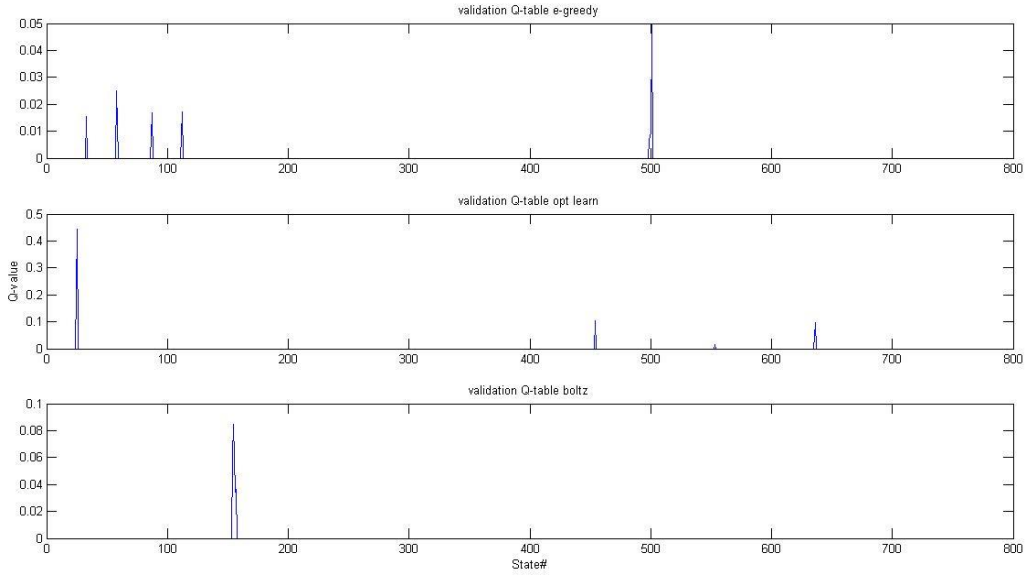

Figure 4-8: Q-table for validation on training data

The ε-greedy learner is selected in this example because it has the most correctly identified samples. Figure 4-9 shows the Q-values when ε-greedy is applied to the testing database. It revisits a few of the states between 130 and 140. The state numbers do not correspond between training and testing datasets due to the size of the datasets. This is the end of the RL Classifier procedure; the Q-values in Figure 4-9 are sent to the RL Classifier for ranking and labeling.

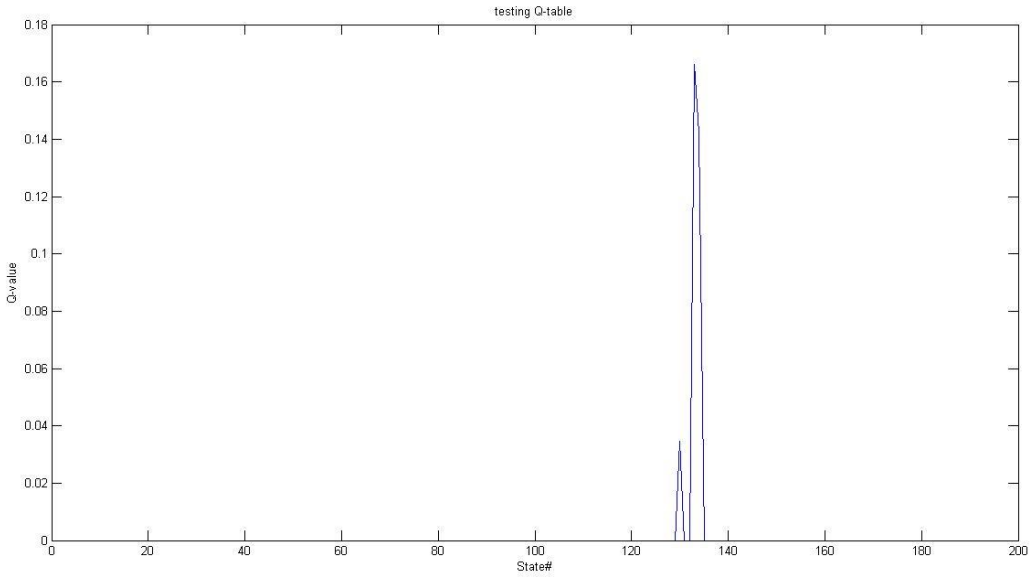

Figure 4-9: Q-values for testing data.

The rewards associated with the optimal path (derived in the V-table discussed in Section 2.2.2) during testing are sent to the RL-HMM hybrid (HyQ, HyF) model. This path of rewards is converted to discrete emissions for the HMM using soft and hard discretization. A sample set of paths after discretization is shown in Figure 4-10.

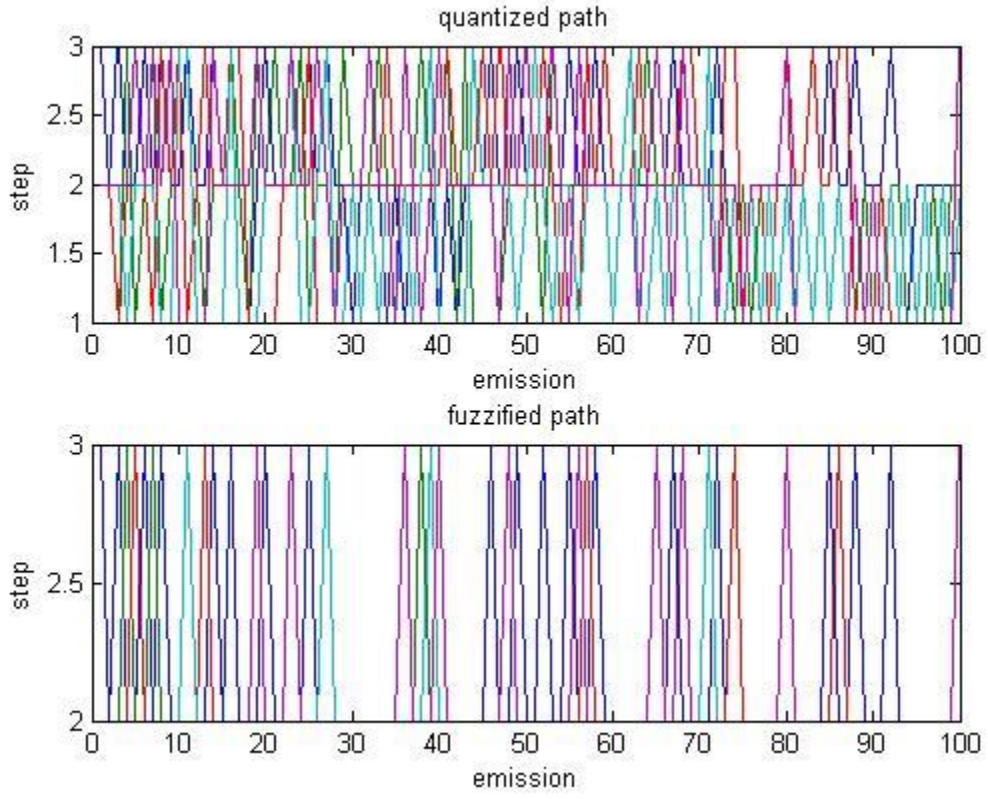

Figure 4-10: Paths after discretization

The fuzzified paths are only medium and high risk whereas the quantized path has all three levels of risk with an average level of medium. Note that these are observations for 100 steps.

The next step in the procedure is to use these reward paths to empirically solve for λ, and then calculate the posterior probabilities to solve the HMM decoding problem. A graph of posterior probabilities for each state is shown in Figure 4-11.

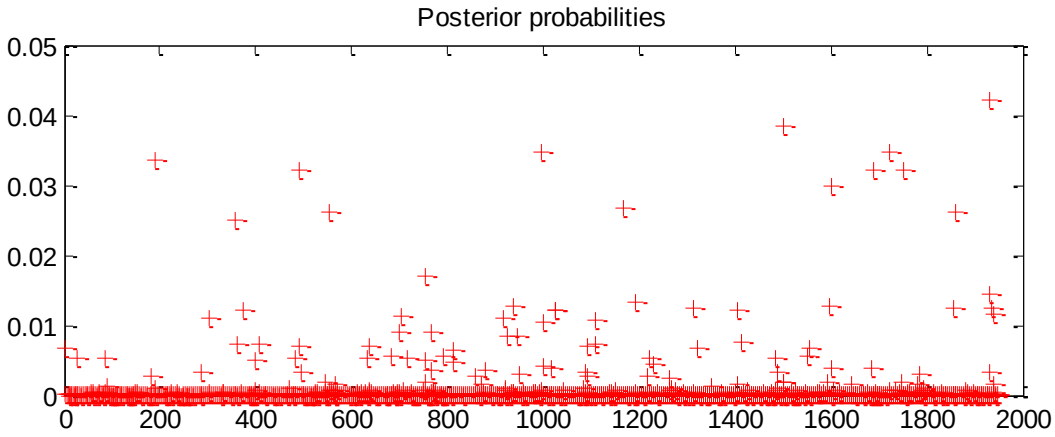

Figure 4-11: Posterior probabilities for insider trading problem

These are the posterior probabilities by state conditional on the observation. This shows the probability of a state occurring given observation and model. It is dependent on time. In order to decouple it from time, it must be multiplied by the probability of an observation occurring at any step. Once this multiplication occurs, the final state probability of risk is found, shown in Figure 4-12.

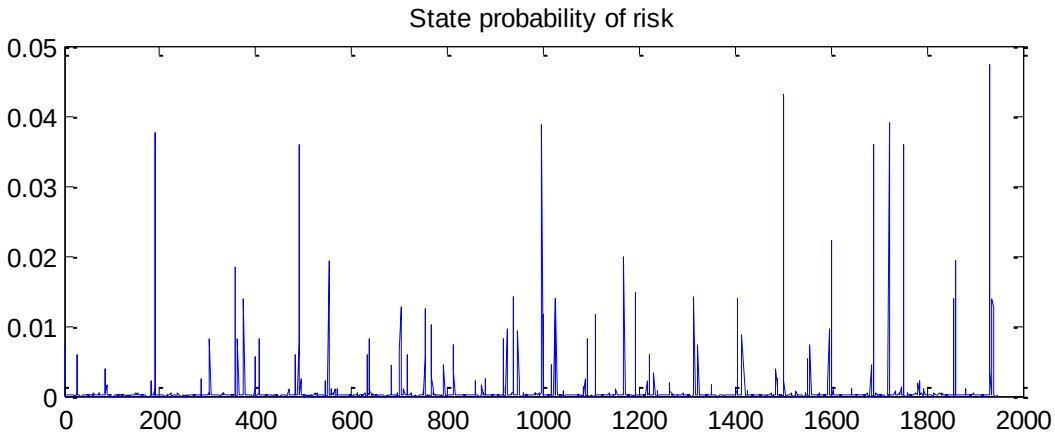

Figure 4-12: State probability of risk for insider trading problem

The resulting ROC for the supervised models is shown in Figure 4-13 on a database of 5000 records of which 100 are sampled. Note that the ROC x-axis has been modified for easier reading. All other parameters are the same as in the RL Search model.

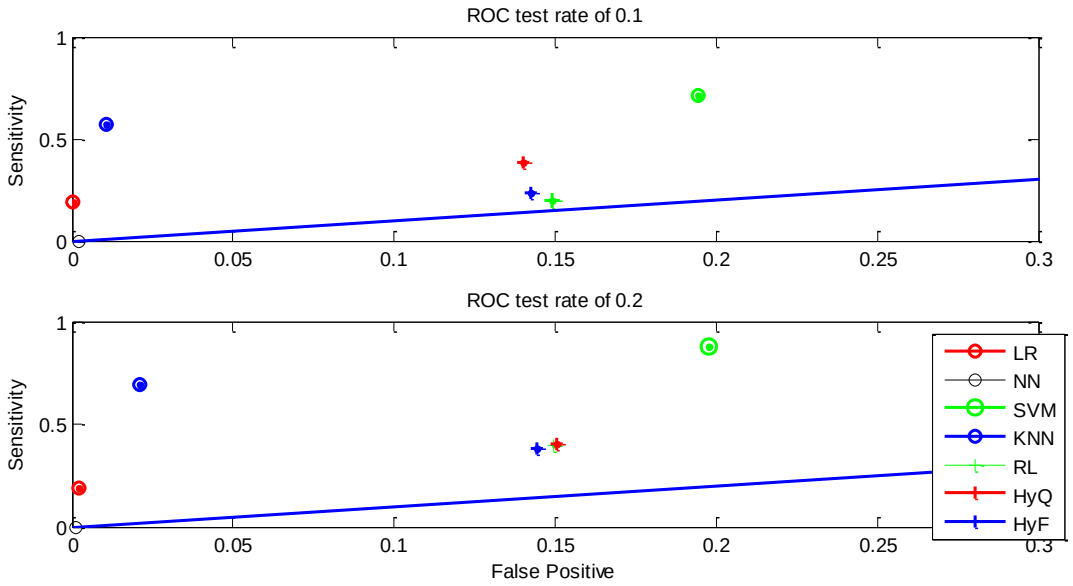

Figure 4-13: ROC for supervised models insider trading

SVM and KNN outperform thesis algorithms by 15% in terms of balance overall when tested on 10% of the database. When tested on 20% of the database the outperformance rate drops to 9%.

The overall balance of both supervised and unsupervised algorithms is expressed by AUC in Figure 4-14. The overall effectiveness of supervised models is not as good in comparison to unsupervised models for algorithms proposed in this thesis. Thesis models perform quite similarly and have a lower misclassification rate than the benchmark, SVM.

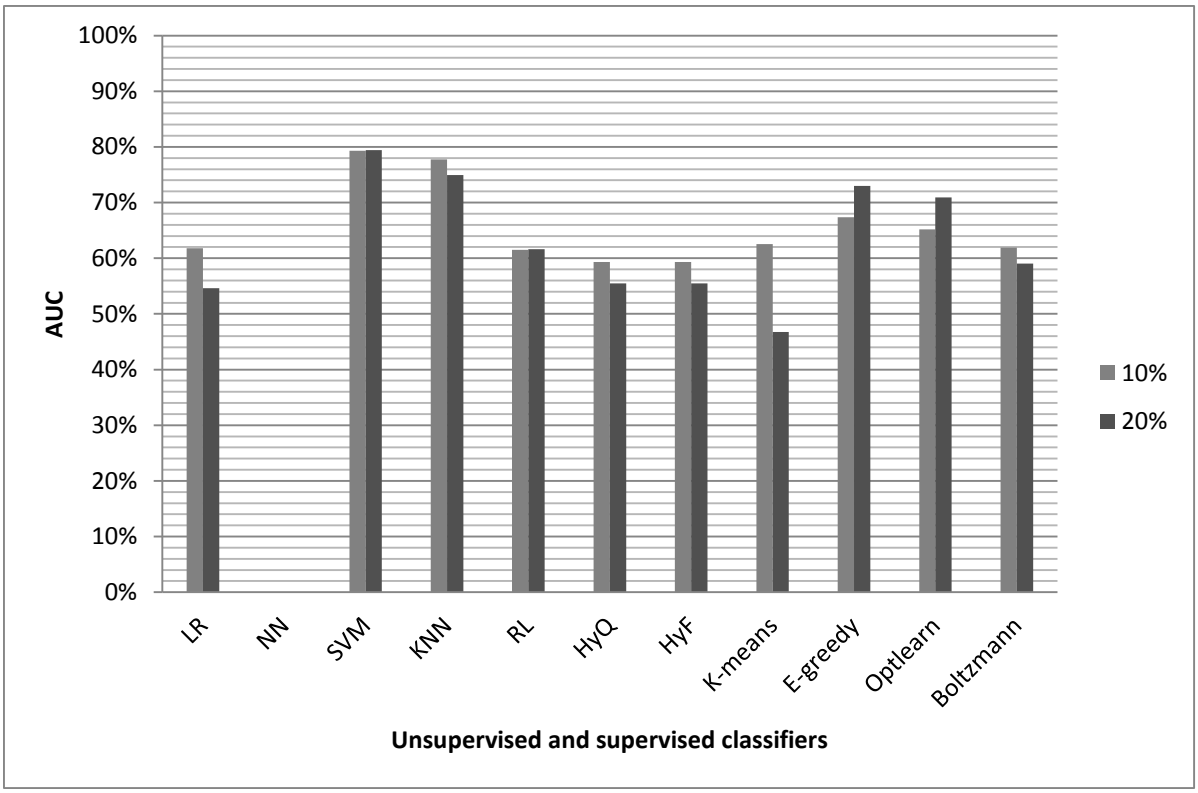

Figure 4-14: Insider trading AUC

This section demonstrated that reinforcement learning is an appropriate tool for solving the problem of financial fraud. It maintains high fraud detection rates while mislabeling a marginal number of fair trades no matter how deep, and rare, the fraudulent trades are buried in the database. The unsupervised RL Search is a suitable algorithm for industrial applications to detect financial fraud.

The unsupervised testing scenario when the learning agent saw 100% of the data represents the real-world procedure of running an automatic fraud detection screening tool after the trading day to detect fraudulent behaviour retroactively. This process could also run throughout the day, and then the intra-day detected states and associated actions could be compared with the end-of-day process for further substantiation of fraudulence.

4.3 Debit card transaction fraud

While the public's attention has focused on corporate financial fraud since the financial crisis of 2008, retail banking fraud has continued to be impacted by adaptive methods of fraudsters. The purpose of this study was to classify worthless deposit fraud (WDF) at ATM machines. This type of fraud occurs when an account holder deposits a blank piece of paper into an ATM machine but says he is depositing $100 (the amount is arbitrary). $100 is then withdrawn within seconds thereafter [104].

The main characteristic of WDF is frequent transactions in a short period of time. In banking terms, this type of transactional behavior is referred to as “aggressive behavior”. In terms of this thesis, the behaviour is translated to be "repeated abuse". A secondary fundamental characteristic is that account tenure is short; a customer with a long history with the bank does not suddenly begin defrauding the bank.

4.3.1 Database

The database in this study was constructed based on similar characteristics of true datasets found in Canadian financial institutions where this author worked for decades and gained considerable knowledge and expertise. The behaviour of the variables and fraudulent scenarios were derived from true transaction database statistics. The database was constructed by maintaining realistic proportions of fraudulent transactions versus non-fraudulent transactions. Realistic parameter values for mean and standard deviation were maintained in the underlying data.

Financial raw data falls into 3 categories: data describing characteristics of the user (e.g. name, address), the account (e.g. account number, portfolio name, tenure), and the actual transaction (e.g. time, price, amount). For WDF the requisite data is a subset of account and transaction data: 4 raw variables (account number, transaction type, transaction amount, tenure) and 4 derived variables (time difference between transaction variables).

The time difference variables represent the amount of time elapsed between two transactions on one account. There are 4 combinations of transactions possible at an ATM: "deposit, deposit" (dep_dep), "deposit, withdrawal" (dep_wd), "withdrawal, withdrawal" (wd_wd), "withdrawal, deposit" (wd_dep). The time difference variables were modeled based on true database statistics using the unit of time = hours. The distributions over the database commonly look like an exponentially decaying function. Since they were also bounded from the top and bottom the double-bounded probability density function (DB-PDF), the Kumaraswamy distribution [105], was selected for modeling. All values are strictly positive. The parameters for fraudulent and non-fraudulent transactions are shown in Table 4-5.

Table 4-5: DB-PDF parameters for time difference model.

	dep_dep	dep_wd	wd_wd	wd_dep
FRAUD				
x_{min}	0	0.004	0.002	0.004
x_{max}	168	167	168	168
F_0	0.1	0.1	0.1	0.1
a	0.4	0.1	0.1	0.215
b	1.9	1.7	1.285	1.15

Non-Fraud				
x_{min}	1	1.0	1	1.0
x_{max}	168	168	168	168
F_0	0.2	0.45	0.24	0.15
a	0.8	0.82	0.9	0.95
b	1.0	1.0	1.0	1.0

In all cases, a<1 and b>=1 which corresponds to graph "C" in [105] where the graph is asymptotic with the x-curve, peaking at x_{min} and dampening towards x_{max}.

Fraudsters execute their transactions at an ATM machine faster than "regular" customers of the bank so as not to be caught. This behaviour is exemplified in the histograms of Figure 4-15 where transaction behaviour between a deposit and withdrawal for fraudulent and non-fraudulent transactions is shown. The x-axis of the graphs is time, in hours, between transactions and the y-axis is the number of transactions. The time elapsed between a deposit and a withdrawal for most fraudulent transactions is less than 20 hours.

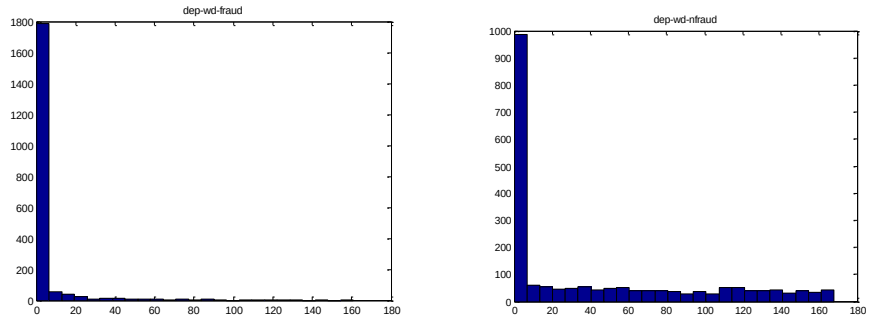

Figure 4-15: Time difference variables distribution for deposit and withdrawal

In engineering, velocity is the change in position (of a body) divided by the change in time [106]. A concept called "transaction velocity" is defined in finance as the change in financial position divided by the change in time. A deposit or withdrawal changes the financial position in an account and transaction velocity is calculated in equation (4-1).

$$transaction_{velocity} = \frac{transaction\_amount}{transaction\_timedifference} \quad \textbf{(4-1)}$$

This calculation was performed for all 4 pairs of transaction types.

The database, called the "debit card fraud database (DCFD)" was constructed with several types of fraudulent scenarios based on real-world events. There are two sets of scenarios: ATM fraud scenarios and organized crime scenarios. All transactions were randomly inserted into the final database while maintaining real-world statistical integrity. The final database statistics are shown in Table 4-6.

Table 4-6: DCFD statistics

Total transactions	#fair	#fraud	%fraud
5000	4637	363	7.2%
Total account #s	#fair	#fraud	%fraud
2036	1958	78	3.8%

These proportions make DCFD an imbalanced database problem.

4.3.2 Reinforcement Learning parameters

This problem is cast as an episodic task. The algorithm begins its search at the first record in the transaction table. The episodic search is complete after a specified number of records are visited.
The system parameters are as follows:

1. $\{S\}$: $\{Account\ 1-2036, Terminal\ 251-500\}$
2. $\{A\}$: $\{account\ number, terminal\ number\}$
3. $R(S,A)$: $max(transaction\ velocity)/tenure$

For the DCFD database there are 509,000 possible states. Not all states exist; for example, one account transacting only at one terminal eliminates 249 states.
The reward for every record is the largest transaction velocity divided by the tenure. The reward is therefore a function of the transaction amount, account tenure and the time elapsed between the previous and current transactions.
The reward changed after every action, with every state, with every transaction. The experiment was to detect suspicious transactional patterns based on the ratio of transaction velocity to tenure. The calculation of transaction velocity was with respect to two transactions so rewards for one account were thus implicitly linked. The learning agent searched for larger than average transaction velocity. When the tenure was short, the reward increased.

4.3.3 RL Search experimental results

The unsupervised learning results are presented first to demonstrate the behaviour of a learning agent permitted to explore and exploit the database without any prior knowledge. This procedure is analogous

to running a learning machine in the background during a business day to collect information about customer transactional behaviour.

- All learning algorithm parameters are as described in Chapter 3.
- The number of records to be sampled = x% of 5,000 (2 times the number of records in the database) where x is the proportion of the database to which the learning agent was exposed.
- The cutoff is 20%. That means 20% of all states were classified as fraudulent
- For the exploratory component of the learning algorithms, ε= 0.8 which means that the learning agent explores 80% of the time.
- $\alpha = 0.1$. This means the agent learns slowly and only takes 1% of the reward value each iteration

The first set of results show how all three RL Search algorithms perform relative to the benchmark, K-means in Figure 4-16 and Figure 4-17. The models were shown gradually more of the database, from 10% to 100%. The results for RL Search become increasingly better as it sees more of the database.

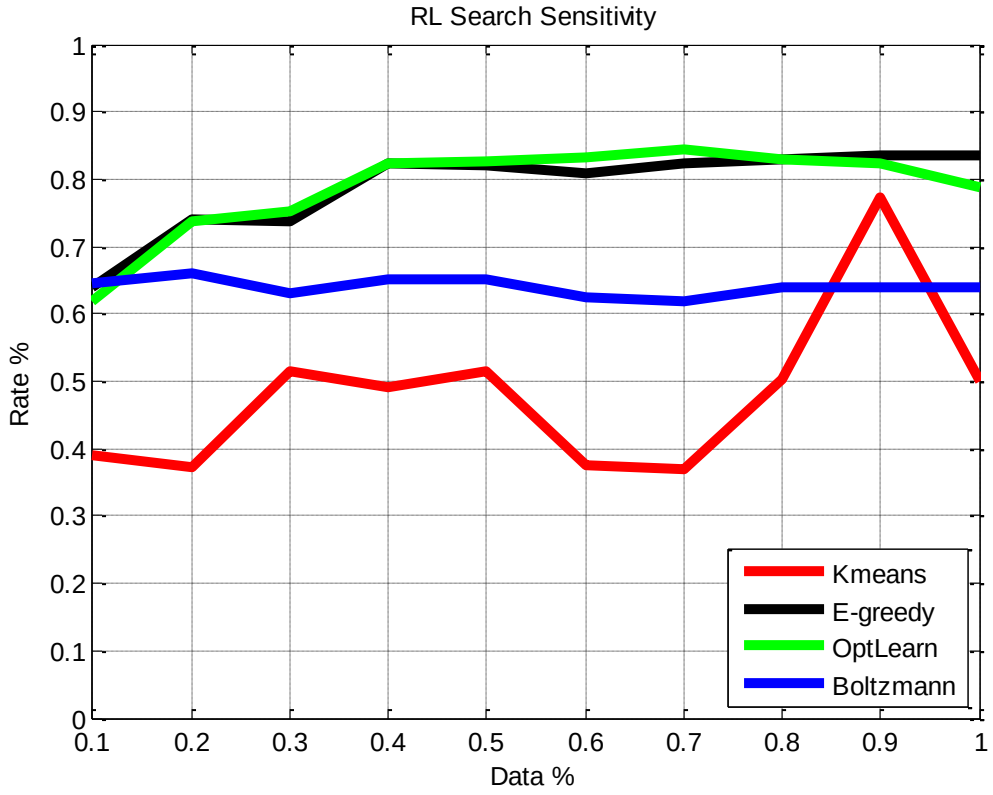

Figure 4-16: DCFD RL Search Sensitivity

RL Search outperforms K-means for every dataset except for at 90% when K-means outperforms Boltzmann. ε-greedy and optimal learning achieve 80% sensitivity consistently after seeing 40% of the

database and on. This result exhibits the ability of RL Search to use link analysis to consistently locate fraudulent activity and focus on it as the dataset increases. It is likely that, for an imbalanced database, the smaller proportions of the data did not contain any positive samples. However as the exposure increases, so does the sensitivity of RL Search. This increased exposure does not appear to improve results for K-means. Figure 4-17 shows the false positive rates for the unsupervised algorithms.

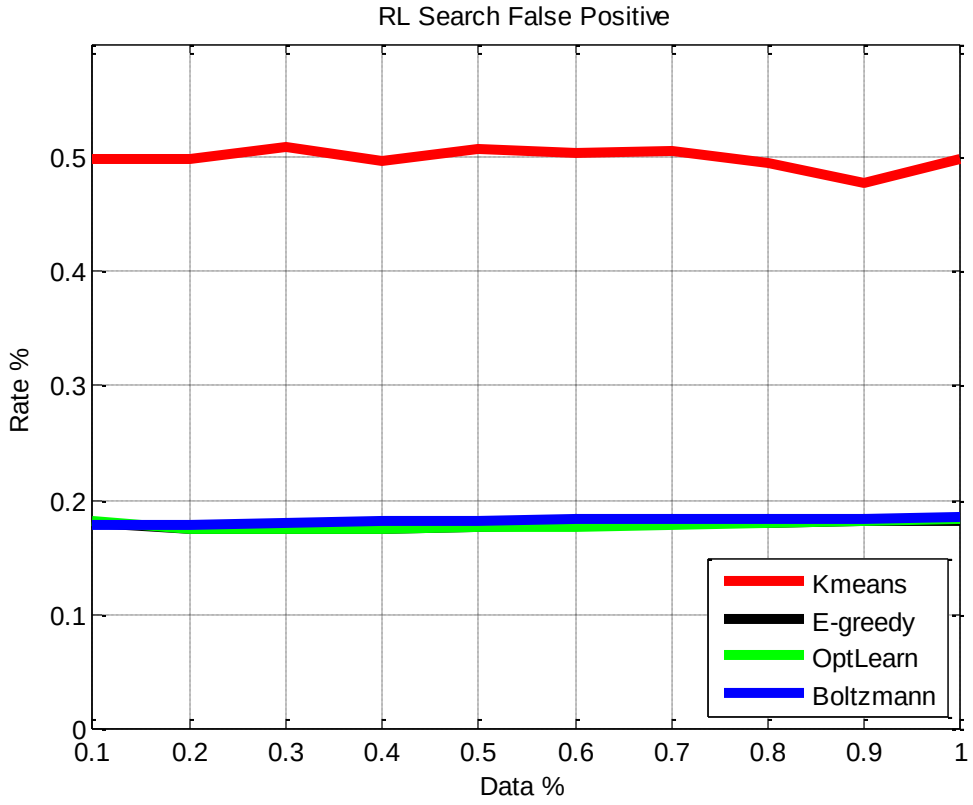

Figure 4-17: DCFD RL Search False Positive

The false positive rate is the classification of regular customers classified as committing WDF. It is less than 20% for all RL Search algorithms. K-means is almost at 50% which means it misclassifies non-fraud samples frequently, and far too often for industrial implementation. One of the goals of this thesis is to construct a machine learning algorithm that can achieve high sensitivity and low false positive. In the DCFD test, a real-world based database, we have achieved that goal.

Over several runs of the data the standard deviations are shown in Table 4-7 demonstrating the consistency of RL Search algorithms. Once again, Boltzmann has the lowest standard deviation for both metrics while K-means has the highest variability.

Table 4-7: Unsupervised learning DCFD standard deviation (%).

	Kmeans	ε-greedy	Optlearn	Boltzmann
Sensitivity	12.15	6.46	6.95	1.25
False Positive	0.87	0.27	0.34	0.25

Supervised learning results will be presented next to demonstrate the power of RL and its hybrid as classifiers.

4.3.4 RL Classifier and RL-HMM hybrid (HyQ, HyF)

The ROC is used to demonstrate the performance of thesis models in comparison to benchmark models for supervised learning. All model parameters are the same as for unsupervised learning except for the number of records which is only 500. The results are shown for datasets of 90%-training, 10%-testing (represented by 0.1) and 80%-training, 20%-testing (represented by 0.2).

The performance of supervised algorithms can be seen in Figure 4-18. Thesis algorithms perform better than LR and NN. LR identifies all of the non-fraudulent transactions as demonstrated by a 0% false positive rate. Neither algorithm finds many fraudulent transactions.

KNN performs the best out of the benchmark algorithms. The thesis algorithms have higher sensitivity and false positive rates. RL-HMM Classifier (HyF) and RL Classifier are competitive with SVM in terms of sensitivity. However, both SVM and KNN strike a balance between high sensitivity and low false positive. The DCFD database shows the same results for both types of data discretization for the RL-HMM hybrid model, HyF and HyQ in the graph. Thesis algorithms are more liberal than all benchmark algorithms in the language of Section 2.9 . Despite the fact that they are liberal, they also have low false positive rates which are the goal of this thesis. Notice that when the testing set increased to 20%, the thesis algorithms' sensitivity rate became closer to the KNN sensitivity rate. False positive rates remain consistent over the two testing sets for all algorithms.

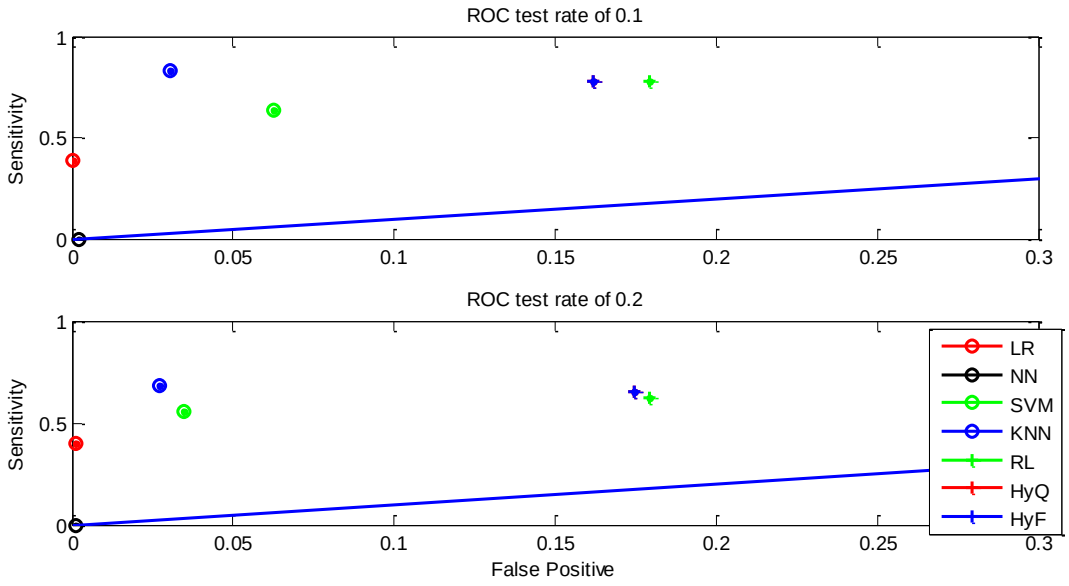

Figure 4-18: DCFD ROC for supervised models

In Figure 4-19 the AUC is compared across all classifiers. AUC reflects the ability of the model to balance true positive and true negative detection rates. RL-HMM hybrid (HyQ, HyF) models are the best performers of thesis algorithms at ~80%. KNN is the best performer overall while the Artificial Neural Network did not detect anything.

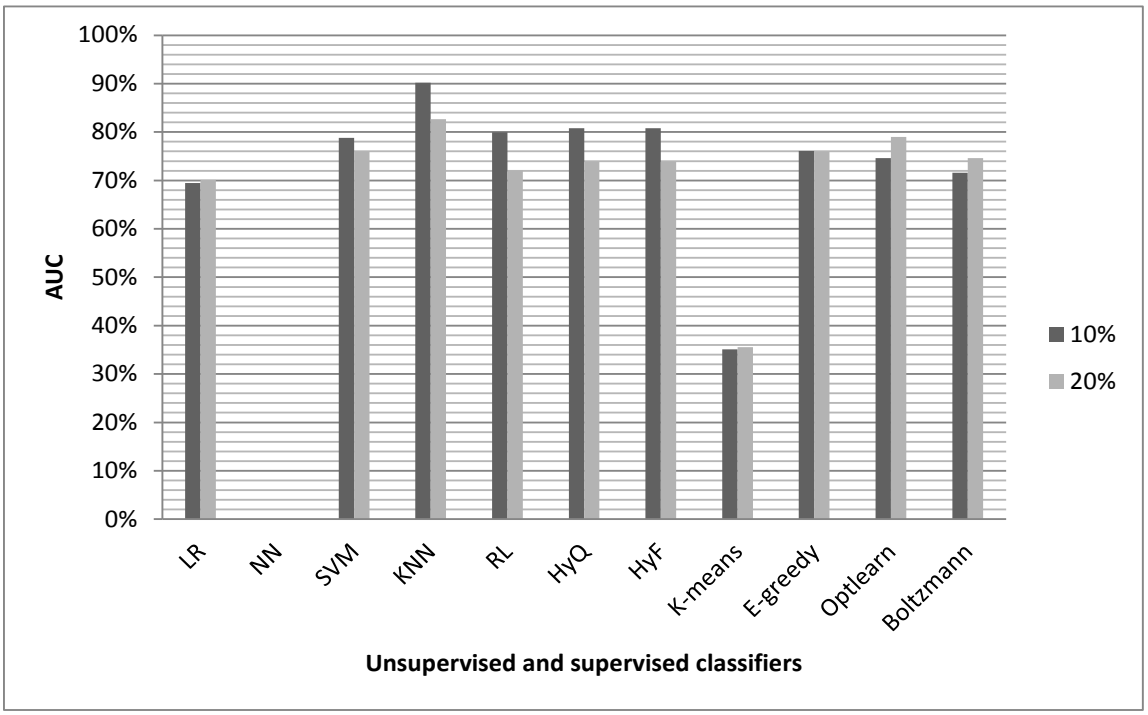

Figure 4-19: DCFD AUC

From these results it can be concluded that the KNN model is best for WDF if there exists time and samples to train the model. However, if the industrial application requires an automated fraud detection screening tool for continuous monitoring with frequent flags emitted from the model, any of the RL Search learning agents will be able to detect the fraud with a low false positive rate. Therefore, for the purpose of financial fraud detection (e.g. insider trading, debit card transaction fraud) the unsupervised thesis algorithms perform better than supervised thesis algorithms and KNN type algorithms in terms of the key metrics used in the thesis, sensitivity and false positive.
Overall, the RL learners and the machine learning framework invented in this thesis are superior tools for financial fraud detection.

4.3.5 Analysis of database size in financial fraud

One of the motivations for creating a supervised learning algorithm was the potential of decreasing the number of records the learner had to search to learn the risky behaviours. We discovered that was in fact true. We did not have to rely solely on the agent for information because we modeled the agent's paths within the succinct probabilistic framework of HMM. Sample values for testing 10% of the database for RL Classifier and RL-HMM hybrid (HyQ, HyF) are shown in Table 4-8 to demonstrate this phenomenon.

Table 4-8: AUC for large databases in supervised fraud detection algorithms

Number of records	**100**	**100**	**100**	**1,000**	**1,000**	**1,000**	**10,000**	**10,000**	**10,000**
Supervised algorithm type	RL	HyQ	HyF	RL	HyQ	HyF	RL	HyQ	HyF
Insider trading	54%	65%	65%	63%	59%	59%	60%	59%	59%
DCFD	75%	76%	76%	72%	73%	73%	73%	73%	73%

As the number of records increased for insider trading, the overall effectiveness of the classifier, represented by AUC, did not improve drastically. We presented results for number of records = 500 for [RL, HyQ, HyF] which had an average AUC of [62%, 57%, 57%] respectively. RL-HMM hybrid (HyF) gets up to 60% AUC for 10,000 records but that value was not enough to justify increasing the computational time for all supervised learners.

In the case of DCFD, AUC decreases after 100 records in Table 4-8, and from our testing 500 records was optimal with AUC of [76%, 75%, 72%] respectively. Ultimately, in both cases the unsupervised learners were superior to any of the supervised learning algorithms, whether the database was large or small.

4.4 Canadian Heart Health Database

The Canadian Heart Health Database (CHHD) is a survey of Canadians from 1986-1992 examining their lifestyles with respect to factors that are known to impact heart health. [1] The purpose of this experiment was to search for relationships between features in the database to discover causes of cardiovascular disease (CVD) by classifying CHHD samples as "CVD" or "healthy". It was a binary classification problem in which "CVD" was a positive response, and "healthy" was a negative response.

Risk factors tend to cluster and interact to increase CVD [4], an observation which opens up an area for significant research using data mining of hidden information. Specific blood pressure or blood cholesterol levels, for example, have little clinical relevance when considered in isolation from other risk factors. [4]. The work in this paper is based on a Master of Science thesis in Health Studies and Gerontology in 2012 [107]. It draws many of its initial propositions with respect to preprocessing and feature selection from the thesis. In contrast to Liu et al's work [88], this thesis shows how to use non-biometric data in a behaviour based model to diagnose CVD.

4.4.1 Database

The database was preprocessed in a similar process to [107]. After preprocessing, the database had 9035 samples and 151 features. The sample CHHD database had an 11.52% rate of positive samples making CHHD an imbalanced database problem.

Features were selected based on a combination of context-based and LVF, shown in Table 4-9. The bracketed variable is the name used in the CHHD database. The significant variables selected by the LVF were non-biometric and focused on behavioural attitudes and actions. The feature types are categorical so that the RL agent can link states using common category values between samples.

The most significant variable in the study was [SEX]. According to [108] and conversations with a researcher in heart health [109], CVD research in the past has failed when applied to women because studies were focused on males. Once [SEX] was used as a feature in the research, researchers were able to detect female CVD faster and design customized treatment to suit their needs.

The feature [GPAGE2] represents age and was an intuitive selection for reward because age is typically a reward for the risk of lifestyle habits.

[1] The database was obtained from Statistics Canada via the Ontario Data Documentation, Extraction Service and Infrastructure (ODESI) under DLI license giving free access for academic purposes. The identification number is: chhd_E_198-1992

Table 4-9: CHHD features

Feature	Variable type	Feature selection method	Feature type	#of categories
[SEX] SEX	Socio-demographic	Context	Binary	2
[CIGCAT] CIGARETTES SMOKED CATEGORIES	Lifestyle	LVF	Categorical	6
[BPPORK] HBP RELATED TO: PORK?	Awareness	LVF	Categorical	3
[SALTFOOD] HOW OFTEN IS SALT ADDED AT THE TABLE?	Lifestyle	LVF	Categorical	5
[GPAGE2] AGE GROUPED IN 10 YEARS	Socio-demographic	Context	Categorical	6

The class variable for each record is designed in this thesis as a binary indicator of heart disease. The "CVD" classification is a combination of responses to three questions: Have you ever had a heart attack? Have you ever had a stroke? Do you have any other heart disease? It is assumed that if a patient answered any of these three questions with yes, that patient is classified as "CVD". If not, then it is classified as "healthy".

4.4.2 Reinforcement Learning parameters

The system parameters are as follows:

4. $\{S\}$: $\{[SEX]1,2; [CIGCAT]1-6; [BPPORK]1-3; [SALTFOOD]; 1-5\}$
5. $\{A\}$: $\{sex, cigcat, bppork, saltfood\}$
6. $R(S,A)$: $[GPAGE2]$

Dahlof, [4] suggested non-biometric data considered with other risk factors should be examined in the study of heart disease. Because RL is designed for collusion and repeated abuse, it is applying his theory of grouping together non-biometric risk factors

4.4.3 RL Search experimental results

- The number of records to be sampled = x% of 9035 where x is the proportion of the database to which the learning agent was exposed.

- The cutoff is 15%. That means 15% of all states (and then corresponding records) were classified as CVD.
- For the exploratory component of the learning algorithms, ε= 0.5 which means that the learning agent explores 50% of the time.
- $\alpha = 0.1$. This means the agent learns slowly and only takes 1% of the reward value each iteration

The sensitivity of RL Search algorithms and Kmeans to heart disease is shown in Figure 4-20.

Figure 4-20: CHHD RL Search Sensitivity

Optimal learning was the most sensitive model and K-means was the least consistent. The LVF tests revealed that there exists a lot of anomalies in the data which is evident from the Boltzmann results; it had the hardest time finding anything, likely because it focused on a few states.

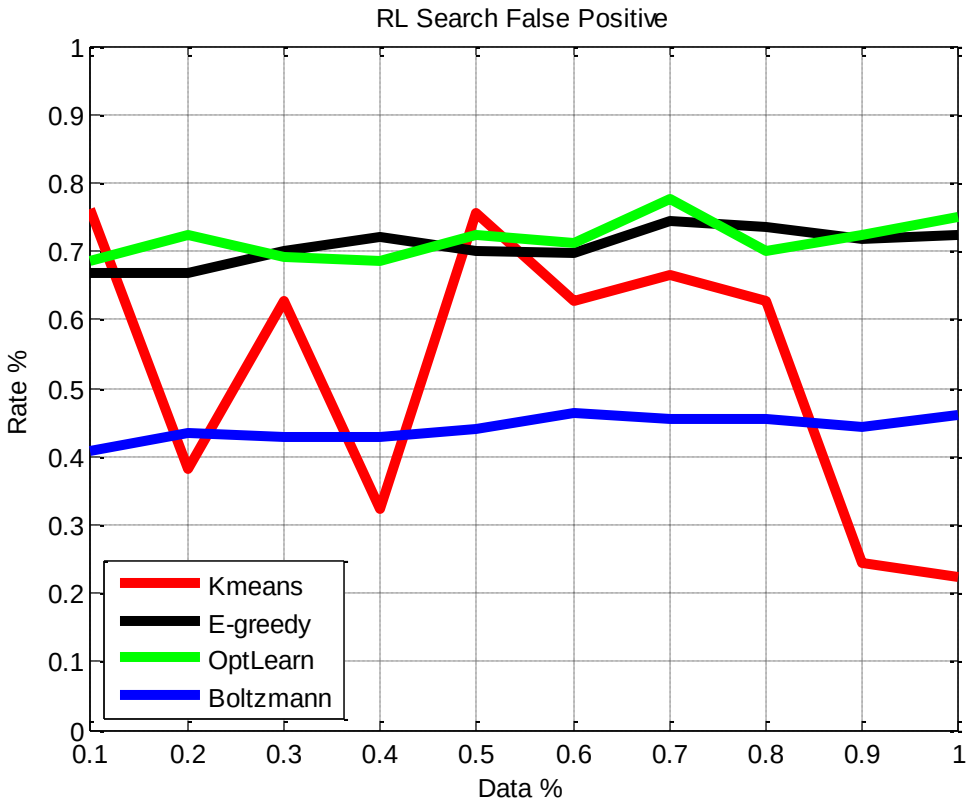

Figure 4-21: CHHD RL Search False Positive.

All false positive graphs in Figure 4-21 mirror the sensitivity graph. This means that the models mislabel as many samples as they label correctly. The unsupervised learners cannot tell the difference between CVD and health.

Over several runs of the data the standard deviations are shown in Table 4-10, once again demonstrating the consistency of RL Search algorithms.

Table 4-10: Unsupervised learning CHHD standard deviation (%).

	Kmeans	e-greedy	Optlearn	Boltzmann
Sensitivity	22.56	4.63	3.77	2.41
False Positive	19.11	3.06	2.88	1.80

4.4.4 RL Classifier and RL-HMM hybrid (HyQ, HyF)

All model parameters are the same as for unsupervised learning except for the number of records to be sampled which was 100.

The sensitivity of all algorithms is quite low. Thesis algorithms are almost random since they are on the line of Figure 4-22.

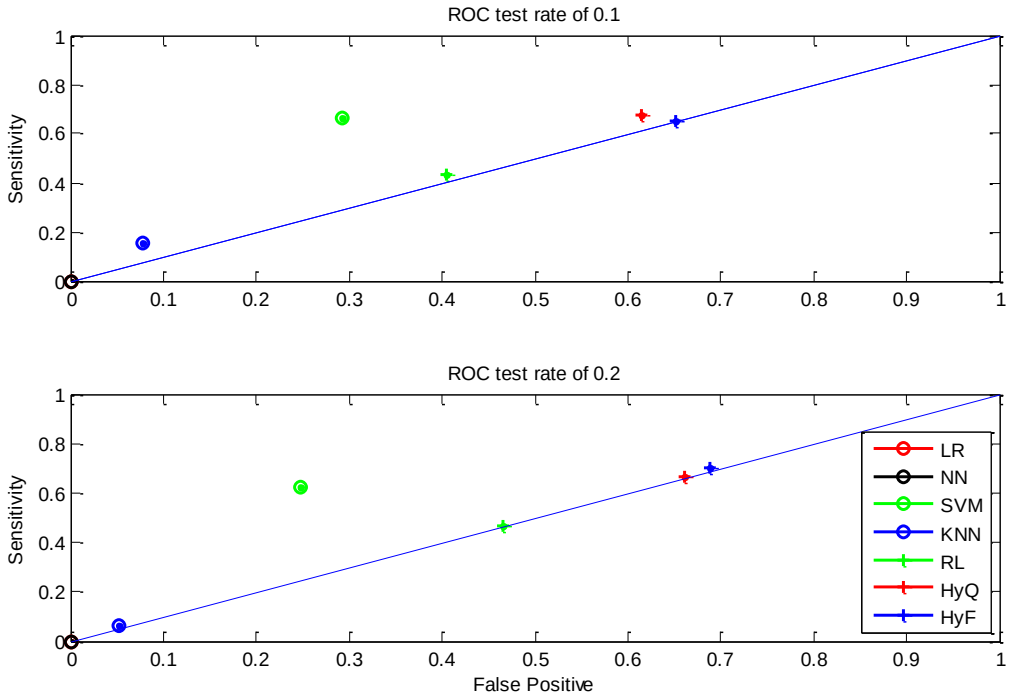

Figure 4-22: CHHD ROC for supervised models

State probabilities in Figure 4-23 are also low which shows the lack of confidence of the RL-HMM hybrid (HyQ, HyF) model and hence the liberal classification results.

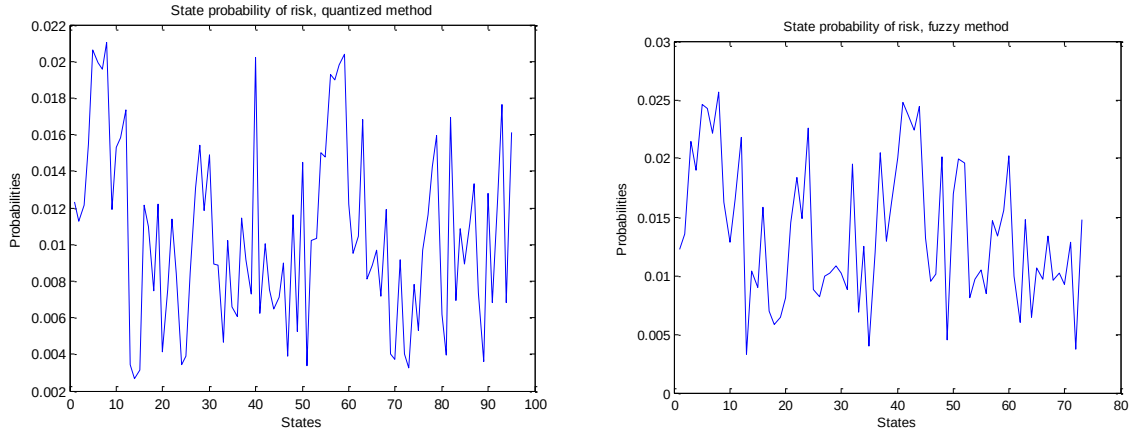

Figure 4-23: CHHD state probabilities for supervised models

The most likely explanation is that when the classifier was exposed to just 10-20% of the data, it could not locate matching states from the training set because there were so many states and so little data. Also,

the optimal action for one state did not seem to apply for all records corresponding to that state because of the amount of dissimilarity in positive samples over the same feature values.

The behavioural analysis has shown promise as a tool but because the same symptom in two different samples can result in different outcomes, the model cannot draw conclusions about collusion of risky behaviour. Interestingly, RL-HMM hybrid (HyQ) was the most sensitive of the algorithms and shows the most promise for future research in CHHD.

In Figure 4-24 the AUC is compared across classifiers. Model performance is similar across classifiers except for neural network which was unable to find anything correctly and SVM which is marginally better than the rest.

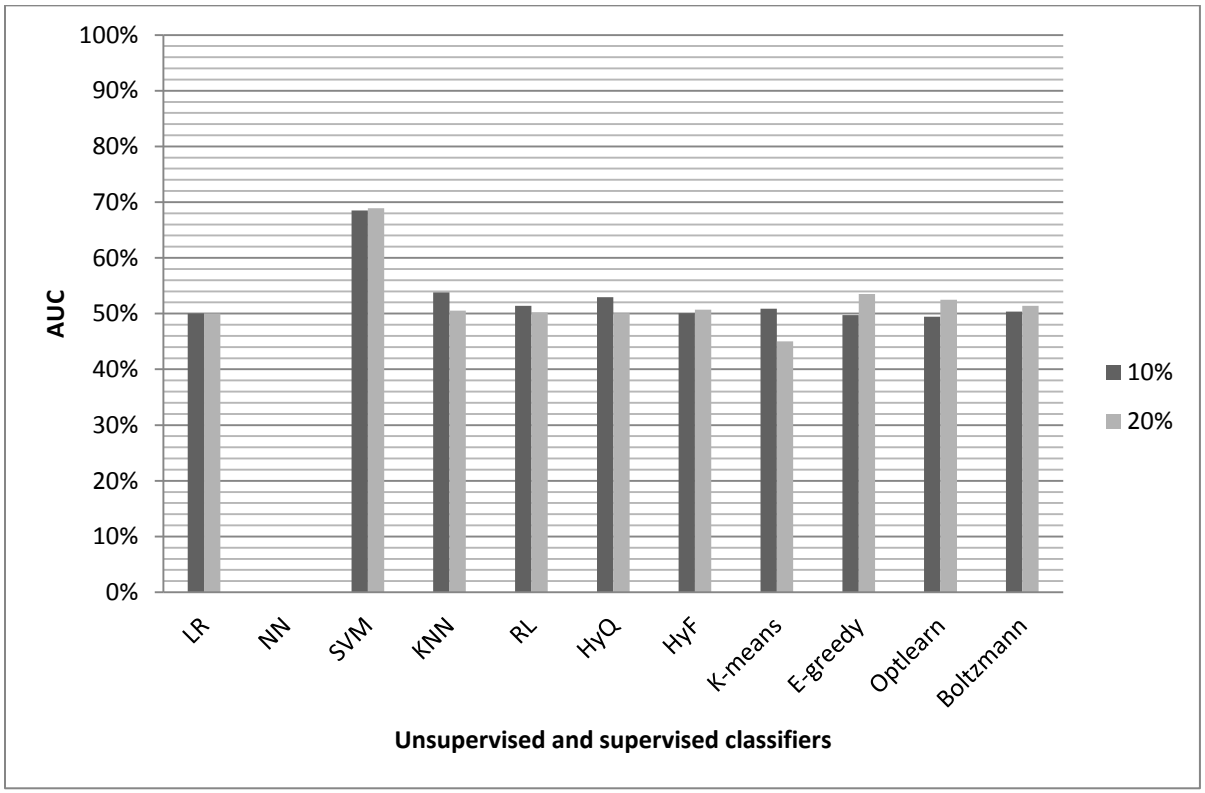

Figure 4-24: CHHD AUC

Sex is a key feature in analysis of the CHHD. Results improved significantly when this variable was added to the analysis. Non-biometric data provided the most information to the RL models. Awareness and lifestyle features appeared often as significant. For example, the [SALTFOOD] variable gave better information than blood pressure. This proves that the RL Classifier is indeed a behavioural model. A principal component analysis showed that many of the samples were similar. Therefore there were many cases where the samples with the same set of features had different classes. Even after several runs on different features, we were unable to obtain a set of features that distinguished over 70% of CVD from

no-CVD samples (in terms of AUC). There are very few feature sets that have repeatable samples. That is, for any one combination of feature values, there are few samples with the exact same feature values. Because the RL technique is based on linking exact feature values and common samples based on behavior, the RL classifier could not find CVD as accurately as other methods. The main reason why we believe that the RL classifier worked for fraud data and not medical data is because there is more entropy in the CHHD; this means that there was more information in the data; the records are not as similar for disease (different feature values can produce disease) as for fraud (fraud cases have similar feature values) so statistical methods may work in these cases. In this sense, fraud is more deterministic may be because of the human intervention. RL Classifier is best applied to databases where at least one of the feature values is the same for the positive samples. CVD therefore should not be detected as an organized crime where several features have a few of the same values for a group of samples that are colluding to produce a positive sample.

However, RL Classifier has revealed that lifestyle and awareness features play a key role in CVD identification. Future work can focus on such variables as listed in this section to further the data mining research into CVD.

4.5 Female Labour Force Participation

The purpose of this study is to model the decision to participate in the labour market. The goal is to evaluate the relevance of explanatory variables that most contribute to the decision to enter the labour market. The study is segregated into female participation in the labour market. The work in this study was motivated by [110], a study of female labour force participation in Venezuela based off the data in the Venezuelan Household survey.

A study of labour force participation requires a data mining tool to uncover the features which are considered in the decision to participate in the labour market. The features in the decision-making process can be considered to be “hidden” information and is a component of "behavioural economics" [110]. From an economic perspective, the problem is perceived as one of allocating finite time to work, leisure and family. By selecting "work", there is a form of exposure for someone to leave the home and participate economically. In this sense, "work" can be considered as "risk" because of the exposure to the conditions of "work" (e.g. physical, mental) with the opportunity cost to leisure, and family. Therefore this study is commensurate with previous work performed on studies of risk in human behaviour [43].

In this study, we will apply the framework used by [110] to Canadian household data.

4.5.1 Database

The female labour force participation study was conducted on a public database compiled by Statistics Canada through the Labour Force Survey (LFS) in February, 2013 [111]. The features selected by the LVF for the problem are shown in Table 4-11 and a summary of database statistics is shown in Table 4-12. Note that only the female records were used for this thesis study to detect female labour force participation and uncover the related decision variables.

Table 4-11: LFS features

Feature	Variable type	Feature selection method	Feature type	#of categories
[MARSTAT] Marital Status	Socio-demographic	LVF	Binary	2
[EFAMTYPE] Type of economic family	Lifestyle	LVF	Categorical	3
[SP_AGE] Age of spouse	Biometric	LVF	Categorical	4

Female participants in the labour force survey are high (>50%) of samples in the database. This would be considered as a "balanced database" since the proportion of positive and negatives classes are almost even.

Table 4-12: LFS Database statistics – February 2013

	FEMALE
#rows	24597
#positive samples	15861
%rate of positive samples	64%

The variable describing "labour force status", [LFSSTAT], is the class variable. This question was divided into 6 categories in the Labour Force Survey: (1) employed, at work, (2) employed, absent from work, (3) unemployed, temporary layoff, (4) unemployed, job searcher, (5) unemployed, future start, (6) not in labour force. We considered (1) to be the "economically active population" and treated these samples as positive. All other categories were considered to be negative samples in that they represented unemployment, or in case of (2), a cost to the employer.

"Hourly earnings" was selected as the reward for the system. These are the responses to the question: "usual hourly wages". There are several blanks in the two databases; there is no reason given for this from Statistics Canada so we assumed that a) the respondent did not want to provide the information or b) the respondent did not collect an hourly wage. We handled this from a data mining point of view by assuming that if HRLYEARN = blank, then it is zero. It was selected because it was the most highly correlated variable with the class variable, labour force status. Additionally, hourly earnings are a reward for labour and therefore a contextual argument can be made for its inclusion.

4.5.2 Reinforcement Learning parameters

The system parameters are as follows:

1. $\{S\}$: $\{[MARSTAT]1,2; [EFAMTYPE]1-3; [SP\_AGE]1-4\}$
2. $\{A\}$: $\{marstat, efamtype, sp\_age\}$
3. $R(S,A)$: $[HRLYEARN]$

There are 24 states in this model. That reflects the number of similar samples that can be grouped under one state.

4.5.3 RL Search experimental results

- The number of records to be sampled = x% of 23091 where x is the proportion of the database to which the learning agent was exposed.
- The cutoff is 20%. That means 20% of all states were classified as "employed".
- For the exploratory component of the learning algorithms, ε= 0.7 which means that the learning agent explores 70% of the time.
- $\alpha = 0.1$. This means the agent learns slowly and only takes 1% of the reward value each iteration

RL Search performed very well in terms of sensitivity (Figure 4-25) however the false positive rates are also relatively high (Figure 4-26).

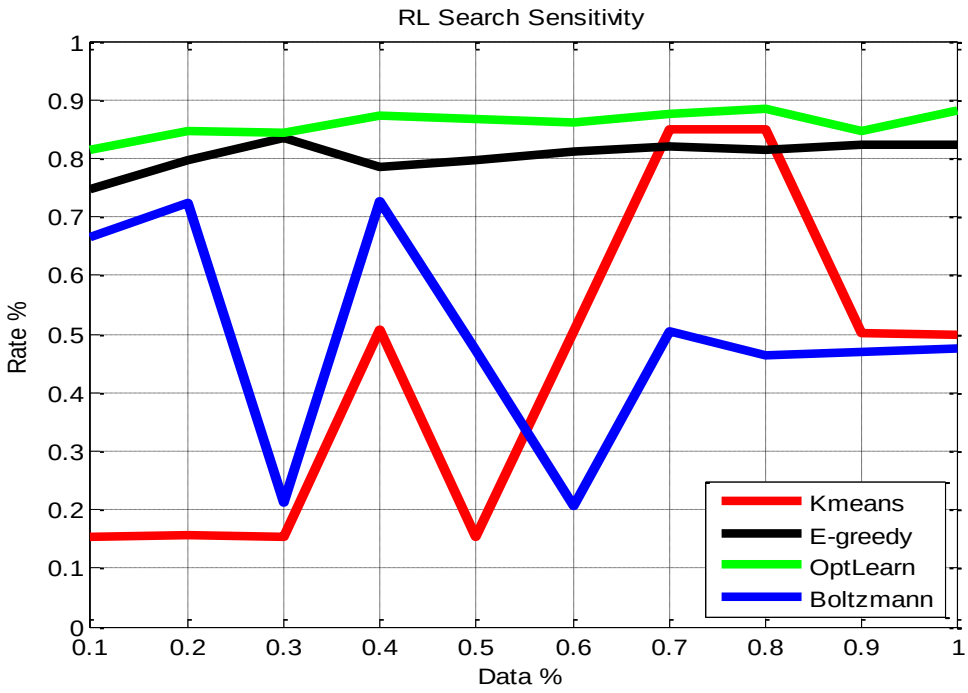

Figure 4-25: LFS RL Search Sensitivity

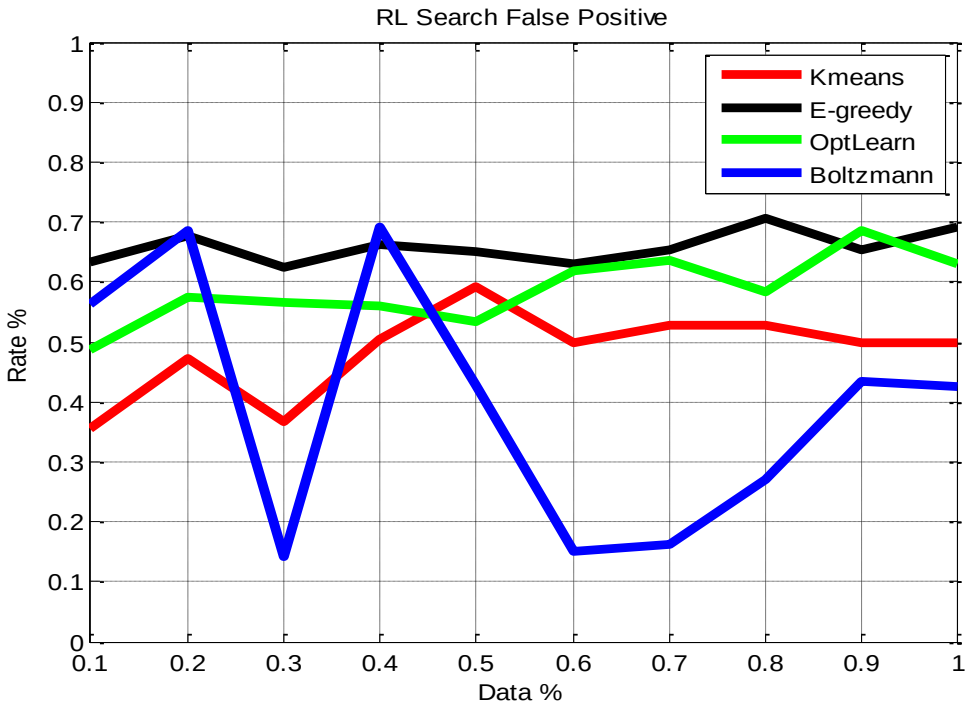

Figure 4-26: LFS RL Search False Positive

The optimal learning agent was the best learner, achieving over 80% sensitivity to female employment and 50-60% misclassification.

Over several runs of the data the standard deviations are shown in Table 4-13, demonstrating the consistency of RL Search algorithms. Boltzmann learning is highly variable in this application to a balanced database in contrast to its performance in the previous three imbalanced applications. Its jagged performance as the exposure to the database increased is evidence of its inability to distinguish between "employed" and "unemployed". The other three algorithms experienced a less dramatic jagged behaviour, with optimal learning being the most steady as it was exposed to more data.

Table 4-13: Unsupervised learning DCFD standard deviation (%).

	K-means	ε-greedy	Optlearn	Boltzmann
Sensitivity	27	3	2	18
False Positive	7	3	6	21

4.5.4 RL Classifier and RL-HMM hybrid (HyQ, HyF)

The ROC is used to demonstrate the performance of thesis models in comparison to benchmark models for supervised learning. All model parameters are the same as for unsupervised learning except for the number of records which is only 100. The results are shown in Figure 4-27 for datasets of 90%-training, 10%-testing (represented by 0.1) and 80%-training, 20%-testing (represented by 0.2).

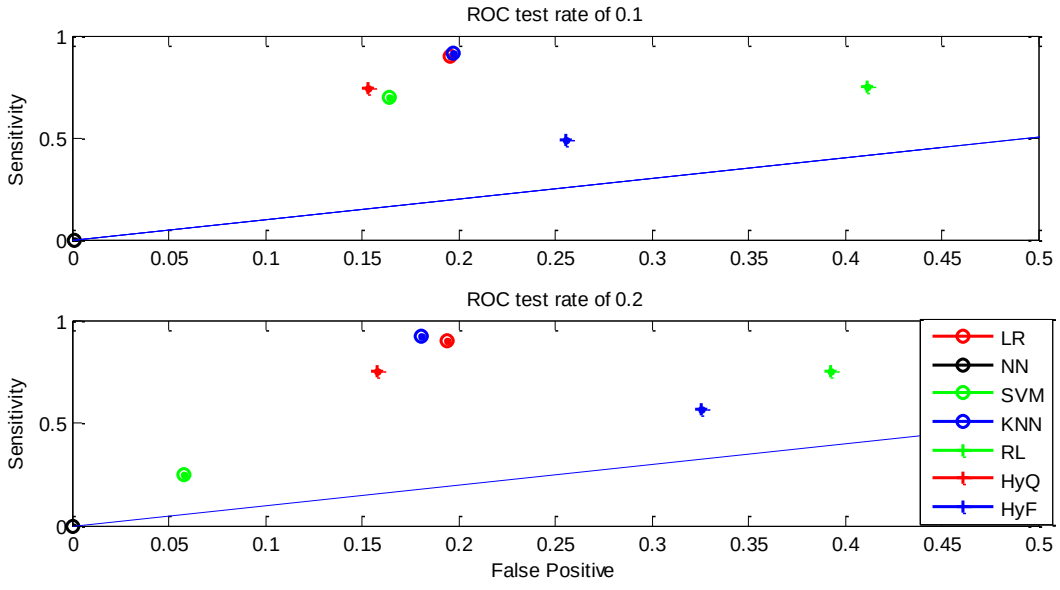

Figure 4-27: LFS ROC for supervised models

The worst performer is NN. The best performer is KNN which has a high sensitivity and low false positive. The best of the thesis algorithms is RL-HMM hybrid (HyQ) which outperforms SVM. RL

Classifier has a high false positive rate with medium sensitivity which mimics the results of the RL Search model.

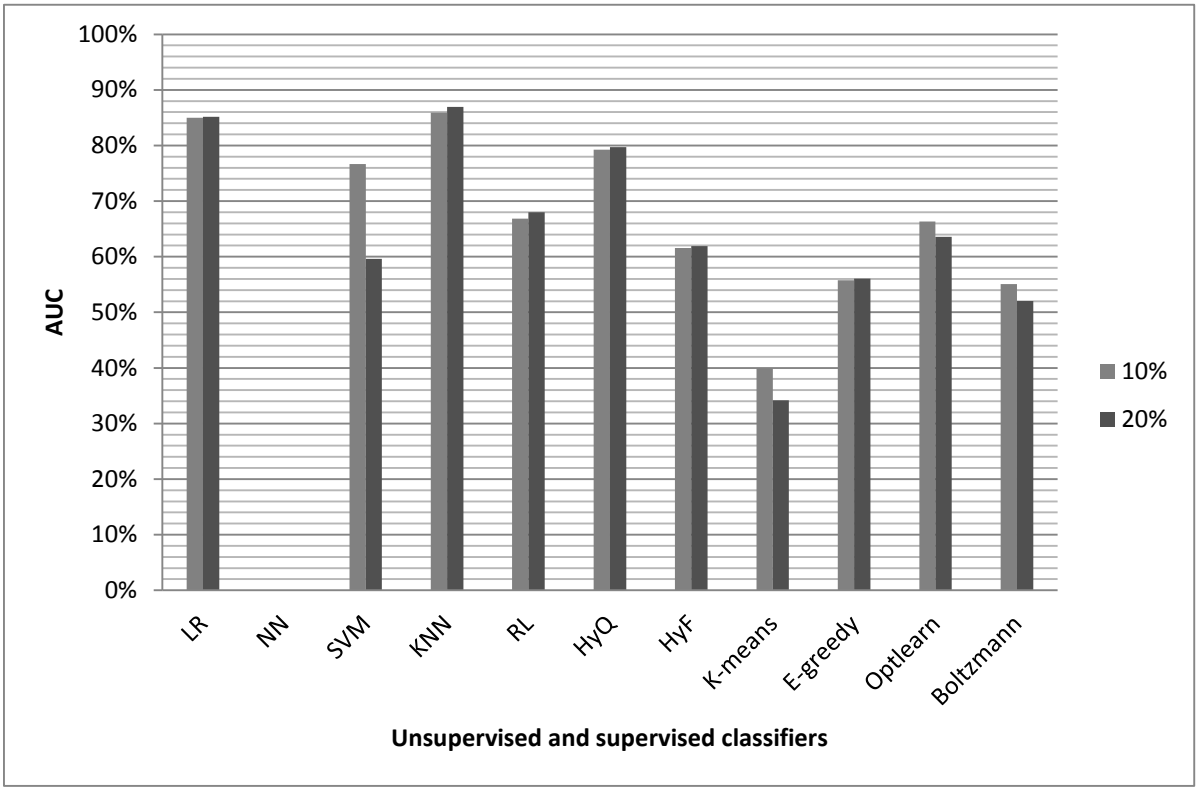

Figure 4-28: LFS AUC

Linear regression model is the most balanced model for LFS. RL-HMM hybrid (HyQ) performed competitively on a balanced database.

4.5.5 Decision variable analysis

The top explanatory variables for female employment in Canada uncovered by the state-space machine learning algorithms were: 1) marital status, 2) family type (description of spousal relationship + employment + number of children), 3) age of spouse. We can surmise that employment is generally dependent on family structure. The set of explanatory variables for the females are highly dependent on their spousal/family arrangement. In particular, female employment is dependent on her spouse: spouse's age, relationship with spouse and age of spouse. This study reinforces a lot of the economic structures that have existed since females entered the workforce, and are pervasive even in the 2013 survey; female employment is dependent on the family.

4.6 Summary of results

One of the goals of this research was to assess the suitability of the three learning algorithms for risk-reward systems.

The ε-greedy learning agent achieved a good balance between exploitation and exploration for the imbalanced databases. The advantage of this learning method is that it did not necessarily return to non-risky states. In particular, since it explored the database 80% of the time for financial fraud applications it was given an opportunity to find more fraudulent states which is beneficial because of the low ratio of fraudulent states to non-fraudulent states.

Optimal learning was also a balanced learner, obtaining AUC values close to ε-greedy. It visited more states than ε-greedy and therefore covered more of the state-space.

The optimal learner's learning method was to estimate the mean reward for each action, and then select the action with the highest mean reward. The ε-greedy learner exploited the database by selecting the action for which the Q-value was highest. In both cases, the maximized value (mean reward, Q-value) was calculated using the reward from the previous step in the algorithm. However, the ε-greedy learner had an additional piece of information: the distance between state-action pairs as calculated by the BAV (Equation (2-1)). Since it did not select actions with lower Q-values it did not return to non-risky states. In contrast, the optimal learner only had reward information however this had an advantage; since it saw more of the state-space it linked more fraudulent states together. Therefore both learning algorithms are recommended for applications in risk-reward systems.

Boltzmann learning estimated action probabilities. It became stuck in local optimums because it was biased to action 1. This bias helped the learner in the imbalanced database because there were many records belonging to a few states but it did not achieve AUC values as high as the other learning algorithms. An exploration component is necessary for a learning algorithm to find positive samples in imbalanced databases.

A summary of the classifier results follows in Table 4-14. The values are the average AUC over 10%-100% of the database for unsupervised algorithms. For the supervised models, the table shows the average AUC over 10% and 20% of the database used for testing.

Table 4-14: AUC: Summary of Results

	RL Search ε-greedy	**RL Search Optimal learning**	**RL Search Boltzmann**
Insider trading AUC	76	74	61
DCFD AUC	81	80	73
CHHD AUC	51	51	51
LFS AUC	57	64	55
	RL Classifier	**RL-HMM Hybrid Classifier Quantile discretization**	**RL-HMM Hybrid Classifier Fuzzy discretization**
Insider trading AUC	62	57	57
DCFD AUC	76	75	72
CHHD AUC	51	53	50
LFS AUC	67	79	62

Thesis algorithms performed well on insider trading and DCFD, particularly when it was untrained. RL sees more records and has more opportunity to explore the database in an unsupervised environment. Of the non-fraud databases, the RL-HMM Hybrid (HyQ) achieved the best results of the thesis algorithms. This shows that the hybridization of RL and HMM could be pursued for more general applications.

4.7 Summary

In this chapter we investigated the performance of developed RL methods in 4 different case studies: insider trading, debit card transaction fraud, heart disease, and female employment. The results have shown that unsupervised and supervised applications of RL have good results. After testing on different types of imbalanced and balanced databases, a few key observations can be made.
RL better performs in an unsupervised environment when exposed to large datasets. When fraud is buried deeper into the database RL Search can identify fraud, related samples, and not mislabel them. RL Search is the best at finding financial fraud in terms of speed, sensitivity and low misclassification cost. The RL techniques developed in this thesis are best applied to databases where at least one of the feature values is the same for positive samples. This is the framework within which the 3 behavioural scenarios were constructed. RL-HMM hybrid (HyQ), with quantized discretion, performed competitively with benchmark algorithms on a balanced database. Thesis algorithms show consistency as repeatable results are necessary for industrial application.

Chapter 5
Summary and Conclusions

The main objective in this thesis was to develop machine learning classification algorithms to uncover risky human behaviour in imbalanced databases, financial industry databases being our main targets. Reinforcement learning (RL) agents can isolate risky human behaviour using a state-space framework for linking risky data samples.

In the first part, the focus was on applying an unsupervised RL algorithm to financial fraud because RL is adaptive and model-free. Three learning algorithms were applied, two of which are popular, one of which is new to the RL literature: ε-greedy, Boltzmann learning and the statistical mean estimation method termed "optimal learning" (new). There are several contributions in this part which can be summarized as follows:

- Establishing a state-space framework in which unique combinations of feature values represented a "state". The framework decreased the search space and summarized behaviour into states which proved to be an ideal method for imbalanced databases containing the three fraudulent scenarios.
- A new approach to feature selection. Instead of looking for a feature set resulting in a sparse dataset, this research required a feature set with common attributes. This was reflected by selecting features that had common feature values occurring in many rows. In fact, dependent rows were preferred. This allowed for detection of collusive and repeatedly abusive behaviour. This thesis has shown that it is not the individual features that make a better classifier; it is how features work together and collude to result in the target variable. This approach of dependent rows for feature selection is novel to the literature for use by RL.
- Development of appropriate models and the demonstration of how to apply these three RL learning agents to financial fraud are presented. Three scenarios were clearly delineated to describe risky behaviour in financial fraud databases, drawn from the author's experience working in the financial industry. They describe two types of risky human behaviour: collusion and repeated abuse. The RL Search algorithms were consistent across samples and showed superior results. The results (for example, fraud detection rates of 75-85%) support the ability of RL Search to be a good screening tool to search very large databases and so can be implemented industry-wide. It is unlikely for an institution to have training data for financial fraud. Therefore,

a solution with unlabeled data for financial fraud is the most practical solution. The learning agents are also consistent with standard deviation of sensitivity and false positive <5%.

- Application of an adaptive learning statistical technique to multi-variable reinforcement learning problem. Typical learning techniques select actions based on a decision to exploit or explore information. Optimal learning modified this approach by selecting actions based on the highest estimated mean of the rewards received by taking an action (exploitation). It benefitted from an exploration component and proved to be the best learning agent for finding financial fraud (AUC of 75-80%). Researchers would benefit from the optimal learning algorithm introduced in this thesis because it covers more of the state-space allowing it to link together positive states.
- Introduction of the terminology "risky human behaviour" to the field of machine learning for data mining by successfully applying three learning algorithms.

In addition, the RL Search algorithm was applied to non-financial problems: heart disease (imbalanced classes), and female labour force (balanced classes).

The research then focused on supervised learning and introduced a new probabilistic framework to completely describe the database like a probability space. This was done in order to construct a novel hybridization of Reinforcement Learning and Hidden Markov (RL-HMM hybrid- HyQ, HyF) models for decision-making and classification of datasets. Combining these two approaches allowed for rapid calculations and multiple path generation for class probability estimation. Additionally, the RL agent's paths were used for empirically solving the HMM "learning" problem for application to imbalanced databases. The following achievements can be pointed out:

- The HMM decoding problem is classically dependent on time. This research produced a novel method of solving the decoding problem for posterior probabilities independent of time. By separating the model from time, the marginal probability of states was used to evaluate the state risk.
- When RL-HMM hybrid (HyQ, HyF) was applied to the two generated databases (financial fraud databases), they did not converge to a HMM solution. However, it did converge for all public databases. Public databases are considered to be those published by Government bodies (the Government of Canada in this case). Private databases are those with protected, corporate information which is not released to the public. We generated our own "private" databases using statistical and behavioural knowledge gained from work experience.

RL-HMM hybrid (HyQ, HyF) can perform well on real industrial data. HMM in particular is known to be too slow for industrial implementation however hybridizing it with RL for training the transition and emission matrices facilitates software efficiency.

- The RL-HMM hybrid (HyQ, HyF) model minimized the number of records required for searching. This is an improvement to RL methods which are known to require a lot of data.
- The RL- HMM hybrid (HyQ, HyF) models were more conservative than the RL Classifier model. This was the purpose of the hybridization; a "conservative" model makes positive classification only with strong evidence so they make few false positive errors. This is at the cost of less positive classifications. RL is model-free so the results are consistent with the literature that RL classifier is a more "liberal" classification algorithm, making positive classifications with weak evidence so they classify nearly all positives correctly.
- RL-HMM hybrid (HyQ, HyF) performed well on a balanced database proving that imposing a probabilistic framework in a smaller state-space can improve the RL agents' detection rates.

The supervised models were exposed to the same data as the unsupervised models. Once again, they were competitive with benchmark models. In fact, we observed that linear regression is inapplicable to binary imbalanced databases because it models the majority class (the negative class). Also, when KNN is supervised, it performed better than when it is unsupervised on thesis databases.
All the mentioned methods have advantages and disadvantages which can be summarized as follows:

1. RL is a superior method for data mining of imbalanced databases. It consistently produces a high rate of sensitivity with a low false positive rate. It performs best in a binary, unsupervised learning environment.
2. All learning methods can be applied in an on-policy (real-time decision-making) and off-policy (simulation) situation. The RL Search and RL Classifier algorithms are model-free which means they do not need transition probabilities. They accurately identify positive states and can discard negative states efficiently. However, they both require a lot of data to increase overall accuracy and therefore take longer than benchmark algorithms.
3. Thesis algorithms perform exceptionally well on imbalanced databases because they zone in on a few related positive states and give them the highest weight. The data constraint on the algorithms is categorical, discrete values.
4. Learning algorithms extend the traditional scheme of programming to include domain knowledge induced by examples and subject matter expert (SME) input. But they do not

remove the need for an SME to validate classifier results which, in this author's humble opinion, should not be the goal of data mining – machine autonomy.

5. The RL models perform best in an environment where the feature set has many repeatable samples. That is, for any one combination of feature values, there are many samples with the exact same feature values.

5.1 Future work

1. Strictly in terms of financial fraud data, the detectable behaviour can be extended to organized crime which involves both insiders and a network of outsiders. This can be accomplished by culling multiple sources of data from banks, brokerage companies, credit rating agencies and regulators to investigate financial fraud. RL Search could function the same but would have to be integrated with a large SQL database for example, keyed on a timestamp and entity.
2. RL algorithms could be used as a data filter because of their ability to locate positive samples so well.
3. To take advantage of the specific learning capabilities of each learning agent, a weight could be assigned to each agent and operate as an ensemble.
4. The computation of the optimal policy requires the best action for each state to be selected. In the thesis framework, that action represents a feature. RL Classifier and RL-HMM hybrid (HyQ, HyF) could be used strictly for discovering trends in attributes, for example, uncovering knowledge of the specific features values contributing to employment, heart disease or the exact name of the stock and/or trader committing fraud.
5. For feature selection further research should go into using methods like LVF that will test a group of features and use the relationship among features to predict the class variable.
6. Big Data applications can use RL Search because of its increasing accuracy when exposed to large datasets. RL Search decreases the search space into states, and then does not need to keep every reward in memory because the value of the RL Search is summarized in the Q-value. It is efficient, cost-effective, and has a high sensitivity rate with a low false positive rate.

References

[1] J. C. Hull, Introduction to Futures and Options Markets, New Jersey: Prentice-Hall Inc., 1991.

[2] D. Ariely, Predicatably Irrational, New York: Harper Collins Publishers, 2009.

[3] D. Kahneman, Thinking, Fast and Slow, Canada: Doubleday Canada, 2011.

[4] B. Dahlof, "Cardiovascular Disease Risk Factors: Epidemiology and Risk Assessment," *The American Journal of Cardiology,* pp. 3A-9A, 2010.

[5] "collusion", Webster's II New Riverside Desk Dictionary, Boston: Houghton Mifflin Company, 1988.

[6] BBC, "Six banks fined 2.6bn by regulators over forex failings," 12 11 2014. [Online]. Available: http://www.bbc.com/news/business-30016007. [Accessed 14 12 2014].

[7] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of the literature," *Decision Support Systems,* vol. 50, pp. 559-569, 2011.

[8] T. E. Senator, "Ongoing Management and Application of Discovered Knowledge in a Large Regulatory Organization: A Case Study of the Use and Impact of NASD Regulation Advanced Detection System (ADS)," *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining,* vol. 6, no. 1, pp. 44-53, 2000.

[9] "learning", Webster's II New Riverside Desk Dictionary, Boston: Houghton Mifflin Company, 1988.

[10] F. N. Ogwueleka, S. Misra, R. Colomo-Palacios and L. Fernandez, "Neural Network and Classification Approach in Identifiying Customer Behaviour in the Banking Sector: A Case Study of an International Bank," *Human Factor and Ergonomic in Manufacturing & Service Industries,* pp. 1-15, 2012.

[11] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman and F. Provost, "Machine learning for targeted display advertising: transfer learning in action," *Machine learning,* vol. 95, no. 1, pp. 103-127, 2014.

[12] D. Zhang and L. Zhou, "Discovering Golden Nuggets: Data Mining in Financial Application," *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews,* vol. 34, pp. 513-522, November 2004.

[13] M. Mahootchi, "Storage System Management Using Reinforcement Learning Techniques and Nonlinear Models," University of Waterloo, Waterloo, 2009.

[14] Y. Sun, *Cost-Sensitive Boosting for Classification of Imbalanced Data (thesis)*, Waterloo: University of Waterloo, 2007.

[15] Y. Qian, "A resampling ensemble algorithm for classification of imbalance problems," *Neurocomputing*, p. ARTICLE IN PRESS, 2014.

[16] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-12847, 2009.

[17] R. Lowenstein, "The Greed Police," *The New York Times Magazine September 25, 2011*, pp. 36-39, 52, 2011.

[18] "The World's Largest Hedge Fund is a Fraud," 7 11 2005. [Online]. Available: http://www.sec.gov/news/studies/2009/oig-509/exhibit-0293.pdf. [Accessed 12 1 2014].

[19] L. Francis, "Banking on Robbery: The Role of Fraud in the Financial Crisis," *Casualty Actuarily Society E-Forum*, vol. 2, no. 1, pp. 1-54, 2010.

[20] IIROC, "Investment Industry Regulatory Organization of Canada," IIROC, [Online]. Available: http://www.iiroc.ca/English/Documents/Rulebook/UMIR0401$_en.pd. [Accessed 18 May 2012].

[21] G. K. Palshikar and M. M. Apte, "Collusion set detection using graph clustering," *Data Mining Knowledge Discovery*, vol. 16, pp. 135-164, 2008.

[22] Z. Wu and X. Wu, "Collusion set detection using a quasi hidden Markov model," *Statistics and It's Interface*, vol. 6, pp. 53-64, 2013.

[23] "UK to extend Libor manipulation laws to cover gold, oil, silver," 22 12 2014. [Online]. Available: http://www.reuters.com/article/2014/12/22/us-britain-forex-law-idUSKBN0K00TU20141222. [Accessed 12 1 2014].

[24] Yahoo, "Yahoo Finance," Yahoo, [Online]. Available: http://ca.finance.yahoo.com. [Accessed 23 April 2012].

[25] H. Goldberg, D. Kirkland, D. Lee, P. Shyr and D. Thakker, "The NASD Securities Observation, News Analysis and Regulation System (SONAR)," *Proceedings of Innovative Applications of Artificial Intelligence Conference 2003*, vol. 15, no. 1, pp. 11-18, 2003.

[26] F. Lu, J. E. Boritz and D. Covvey, "Adaptive Fraud Detection Using Benford's Law," *Advances in Artificial Intelligence, Lecture Notes in Computer Science,* pp. 347-358, 2006.

[27] D. Zhang and L. Zhou, "Discovering Golden Nuggets: Data Mining in Financial Application," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 34, no. 4, pp. 513-522, 2004.

[28] J. D. Kirkland, T. E. Senator, J. J. Hayden, T. Dybala, H. G. Goldberg and P. Shyr, "The NASD Regulation Advanced-Detection System (ADS)," *AI Magazine,* vol. 20, pp. 55-67, 1999.

[29] E. Ciolko, F. Lu and A. Joshi, "Intelligent Clinical Decision Support Systems Based on SNOWMED CT," *32nd Annual International Conference of the IEEE EMBS,* vol. 32, no. 1, pp. 6781-6784, 4 September 2010.

[30] N. Lui, Z. X. Koh, E. C.-P. Chua, L. M.-L. Tan, Z. Lin, B. Mirza and M. E. H. Ong, "Risk Scoring for Prediction of Acute Cardiac Complications from Imbalanced Clinical Data," *IEEE Journal of Biomedical and Health Informatics,* vol. 18, no. 6, pp. 1894-1902, 2014.

[31] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge: MIT Press, 1998.

[32] O. C. Ibe, Markov Processes for Stochastic Modeling, USA: Elsevier Inc., 2009.

[33] J. Si, A. G. Barto, W. B. Powell and D. Wunsch II, Handbook of Learning and Approximate Dynamic Programming, New Jersey: John Wiley & Sons, Inc, 2004.

[34] E. L. Thorndike, Animal Intelligence, Darien, CT: Hafner, 1911.

[35] O. Maimon and S. Cohen, "A Review of Reinforcement Learning Methods," *Data Mining and Knowledge Discovery Handbook, 2nd ed,* vol. 2, pp. 401-417, 2010.

[36] O. Sigaud and S. W. Wilson, "Learning classifier systems: a survey," *Soft Computing,* vol. 11, pp. 1065-1078, 2007.

[37] M. G. Lagoudakis and R. Parr, "Reinforcement Learning as Classification: Leveraging Modern Classifiers," *Proceedings of the Twentieth International Conference on Machine Learning,* vol. 20, no. 1, 2003.

[38] T. Ruckstieb, C. Osendorfer and P. van der Smagt, "Sequential Feature Selection for Classification," *Lecture Notes in Artificial Intelligence,* vol. 7106, pp. 132-141, 2011.

[39] W. Barbakh and C. Fyfe, "Clustering with Reinforcement Learning," *Intelligent Data Engineering and Automated Learning - IDEAL 2007,* vol. 1, no. 1, pp. 507-516, 2007.

[40] N. Entezari, M. E. Shiri and P. Moradi, "A local graph clustering algorithm for discovering subgoals in reinforcement learning," *Communication and Networking,* vol. 120, pp. 41-50, 2010.

[41] F. Lu, "Uncovering Fraud in Direct Marketing Data with a Fraud Auditing Case Builder," *Lecture Notes in Artificial Intelligence PKDD 2007,* pp. 540-547, 2007.

[42] D. Ippoliti, *Automated Network Anomaly Detection with Learning, Control and Mitigation (thesis),* Colorado Springs: University of Colorado, 2006.

[43] D. Sawh, K. Ponnambalam and F. O. Karray, "Artificial Intelligence Modeling of Financial Profit and Fraud," *Proceedings of the World Congress on Engineering 2011,* pp. 381-383, 2011.

[44] S. Gregory, "An algorithm to find overlapping community structure in networks," *Lecture Notes in Artificial Intelligence,* vol. 4702, pp. 91-102, 2007.

[45] M. Palatucci and T. M. Mitchell, "Classification in Very High Dimensional Problems with Handfuls of Examples," *PKDD 2007, LNAI 4702,* vol. PKDD 2007, pp. 212-223, 2007.

[46] L. P. Kaelbling and M. L. M. A. W. Littman, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research,* vol. 4, no. 1, pp. 237-285, 1996.

[47] W. B. Powell and I. O. Ryzhov, Optimal Learning, New Jersey: John Wiley & Sons, Inc., 2012.

[48] J. Quastel, "Stochastic calculus for mathematical finance notes," [Online]. Available: http://www.math.toronto.edu/quastel/fin.html. [Accessed 23 April 2012].

[49] M. E. Ayadi, M. S. Kamel and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition,* vol. 44, pp. 572-587, 2011.

[50] Z. Eilser, J. Perello and J. Masoliver, "Volatility: A hidden Markov process in financial time series," *Physical Review,* vol. 76, pp. 056105-1:056105-11, 2007.

[51] M. Kalimipalli and S. Nayak, "Idiosyncratic volatility vs. liquidity? Evidence from the US corporate bond market," *Journal of Financial Intermediation,* vol. 21, pp. 217-242, 2012.

[52] A. Khan, T. Singh and A. Sinhal, "Implement Credit Card Fraudulent Detection System Using Observation Probabilistic in Hidden Markov Model," *2012 Nirma University International Conference on Engineering,* pp. 1-6, 2012.

[53] K. Ponnambalam, "SYDE 531: Design Optimization Under Probabilistic Uncertainty," 1 January 2014. [Online]. Available: http://epoch.uwaterloo.ca/sd511f98lect1.pdf. [Accessed 28

August 2014].

[54] S. Romdhani, "Implementation fo DNN-HMM Acoustic Models for Phoneme Recognition," University of Waterloo, Waterloo, 2014.

[55] I. Visser, "Seven things to remember about hidden Markov models: A tutorial on Markovian models for time series," *Journal of Mathematical Psychology*, vol. 55, pp. 403-415, 2011.

[56] M. Kim and V. Pavlovic, "Sequence classification via large margin hidden Markov models," *Data Mining Knowledge Discovery*, vol. 23, pp. 322-344, 2011.

[57] I. Konenko and M. Kukar, Machine Learning and Data Mining, West Sussex: Horwood Publishing Limited, 2007.

[58] D. Michie, D. Spiegelhalter and C. Taylor, Machine Learning, Neural and Statistical Classification, Hertfordshire: Ellis Horwood Limited, 1994.

[59] M. Krivko, "A hybrid model for plastic card fraud detection systems," *Expert Systems with Applications*, vol. 37, pp. 6070-6076, 2010.

[60] D. J. Weston, D. J. Hand, N. M. Adams, C. Whitrow and P. Juszczak, "Plastic card fraud detection using peer group analysis," *ADAC*, vol. 2, pp. 45-62, 2008.

[61] G. Mamalakis, C. Diou, A. Symeonidis and L. Georgiadis, "Of daemons and men: A file system approach towards intrusion detection," *Applied Soft Computing*, vol. 25, pp. 1-14, 2014.

[62] R. J. B. a. H. ,. D. J. Bolton, "Statistical Fraud Detection: A Review," *Statistical Science*, pp. 235-255, 2002.

[63] A. Taylor- Butts and S. Perreault, "Fraud Against Businesses in Canada: Results from a National Survey," Statistics Canada, Ottawa, 2008.

[64] C. Phua, V. Lee, K. Smith and R. Gayler, "A Comprehensive Survey of Data Mining-based Fraud Detection Research," *Artficial Intelligence Review*, pp. 1-14, 2005.

[65] S. Jha, M. Guillen and J. C. Westland, "Employing transaction aggregation strategy to detect credit card fraud," *Expert Systems with Applications*, vol. 39, pp. 12650-12657, 2012.

[66] A. Khorasani and M. R. Daliri, "HMM for Classification of Parkinson's Disease Based on the Raw Gait Data," *Journal of Medical Systems*, vol. 38, no. 147, pp. 1-6, 2014.

[67] F. O. Karray and C. de Silva, Soft Computing and Intelligent Systems Design: Theory, Tools

and Applications, Essex: Pearson Education Limited, 2004.

[68] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.

[69] B. M. Abidine, B. Fergani, M. Oussalah and L. Fergani, "A new classification strategy for human activity recognition using cost sensitive support vector machines for imbalanced data," *Kybernetes,* vol. 43, no. 8, pp. 1150-1164, 2014.

[70] Z. Yu, Y. Guang and J. Zi-qi, "Violations detection of listed companies based on decision tree and k-nearest neighbour," *2013 International Conference on Management Science & Engineering,* vol. 20, no. 1, pp. 1671-1676, 2013.

[71] A. Majid, S. Ali, M. Iqbal and N. Kausar, "Prediction of human breast and colon cancers from imbalanced data using nearest neighbor and support vector machines," *Computer methods and program in Biomedicine,* vol. 113, pp. 792-808, 2014.

[72] Y. Finance, "Yahoo! Finance FICO," Yahoo!, 4 August 2013. [Online]. Available: http://finance.yahoo.com/q?s=fico&ql=1. [Accessed 4 August 2013].

[73] F. I. Corporation, "FICO," 16 October 2006. [Online]. Available: http://investors.fairisaac.com/phoenix.zhtml?c=67528&p=irol-newsArticle_print&ID=916693&highlight=. [Accessed 4 August 2013].

[74] Y. Finance, "Yahoo! Finance EFX," 4 8 2013. [Online]. Available: http://finance.yahoo.com/q?s=EFX&ql=1. [Accessed 4 8 2013].

[75] DSstar, "DSstar," 1999. [Online]. Available: http://www.tgc.com/dsstar/00/0229/101351.html. [Accessed 4 8 2013].

[76] H. Yoon, C.-S. Park, J. S. Kim and J.-G. Baek, "Algorithm learning based neural network integrating feature selection and classification," *Expert Systems with Applications,* no. 40, pp. 231-241, 2013.

[77] R. Guzman-Martinez and R. Alaiz-Rodriguez, "Feature Selection Stability Assessment Based on the Jensen-Shannon Divergence," *Lecture Notes in Computer Science,* vol. 6911, pp. 597-612, 2011.

[78] D. C. d. L. Vieira, P. J. Adeodato and P. M. Goncalves Jr., "Improving Reinforcement Learning Algorithms by the Use of Data Mining Techniques for Feature and Action Selection," *IEEE International Conference on Systems, Man and Cybernetics,* pp. 1863-1870, 2010.

[79] C. Springer, Statistics 230 Course Notes, Waterloo: University of Waterloo, 1996.

[80] I. The Mathworks, *MATLAB R2012b version 8.0.0.783*, Natick, Massachusetts: The Mathworks, Inc., 2012.

[81] L. Zadeh, "Fuzzy Sets," *Information and Control,* vol. 8, pp. 338-353, 1965.

[82] A. Bagheri, H. M. Peyhani and M. Akbari, "Financial forecasting using ANFIS networks wtih Quantum-behaved Particle Swarm Optimization," *Expert Systems with Applications,* vol. 41, pp. 6235-6250, 2014.

[83] L.-Y. Wei, C.-H. Cheng and H.-H. Wu, "A hybrid ANFIS based on n-period moving average model to forecast TAIEX stock," *Applied Soft Computing,* vol. 19, pp. 86-92, 2014.

[84] R. Jain and A. Abraham, "A comparative study of fuzzy classification methods on breast cancer data," *Australasian Physical and Engineering Sciences in Medicine,* vol. 27, no. 4, pp. 213-218, 2004.

[85] Wang, Shuo and X. Yao, "Multiclass Imbalance Problems: Analysis and Potential Solutions," *IEEE Transactions on Systems, Man, and Cybernetics - Part B,* vol. 42, no. 4, pp. 1119-1130, 2012.

[86] H. He, Self-Adaptive Systems for Machine Inteligence, New Jersey: John Wiley & Sons, Inc. , 2011.

[87] R. Diao, F. Chao, T. Peng, N. Snooke and Q. Shen, "Feature Selection Inspired Classifier Ensemble Reduction," *IEEE Transactions on Cybernetics,* vol. 44, no. 8, pp. 1259-1268, 2014.

[88] N. Liu, X. K. Koh, E. C.-P. Chua, L. M.-L. Tan, Z. Lin, B. Mirza and M. E. H. Ong, "Risk Scoring for Prediction of Acute Cardiac Complications from Imbalanced Clinical Data," *IEEE Journal of Biomedical and Health Informatics,* vol. 18, no. 6, pp. 2168-2194, 2014.

[89] N. Chawla, "SMOTE," 2 6 2002. [Online]. Available: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/node6.html. [Accessed 2 2 2015].

[90] E. Turban, R. Sharda and D. Delen, Decision Support and Business Intelligent Systems 9th Edition, New Jersey: Prentice Hall, 2011.

[91] A. P. Bradely, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition,* vol. 30, no. 7, pp. 1145-1159, 1997.

[92] V. Garcia, J. Sanchez, R. Martin-Felex and R. Mollineda, "Surrounding neighbourhood-based SMOTE for learning from imbalanced data sets," *Progress in Artifiicial Intelligence,* vol. 1, no. 4, pp. 347-362, 2012.

[93] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management,* vol. 45, pp. 427-437, 2009.

[94] A. G. V. S. J. Marques, "Two-level classifier ensembles for credit risk assessment," *Expert Systems with Applications,* vol. 39, pp. 10916-10922, 2012.

[95] T. Fawcett, "An introduction to ROC analysis," *Patter Recognition Letters,* vol. 27, no. 1, pp. 861-874, 2006.

[96] B. C. Wallace and I. J. Dahabreh, "Improving class probability estimates for imbalanced data," *Knowledge Information Systems,* vol. 41, pp. 33-52, 2014.

[97] A. Bagherjeiran, C. F. Eick and R. Vilalta, "Adaptive Clustering: Better Representatives with Reinforcement Learning," Department of Computer Science, University of Houston, Houston, 2005.

[98] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," *Statistical Science,* pp. 235-255, 2002.

[99] Z. Wu and X. Wu, "Collusion set detection using a quasi hidden Markov model," *Statistics and It's Interface,* vol. 6, pp. 53-64, 2013.

[100] R. Bellman, "The Theory of Dynamic Programming," Santa Monica, 1954.

[101] F. Karray, *SYDE558 Note Set 2,* Waterloo: University of Waterloo, 2001.

[102] R. Saracoglu, "Hidden Markov model-based classification of heart valve disease with PCA for dimension reduction," *Engineering Applications of Artificial Intelligence,* vol. 25, pp. 1523-1528, 2012.

[103] L. Batista, E. Granger and R. Sabourin, "Improving performance of HMM-based off-line signature verification systems through a multi-hypothesis approach," *International Journal on Document Analysis and Recognition,* vol. 13, pp. 33-47, 2010.

[104] G. o. Canada, "Access to Funds Regulations," 10 July 2013. [Online]. Available: http://laws-lois.justice.gc.ca/PDF/SOR-2012-24.pdf. [Accessed 30 July 2013].

[105] P. Kumaraswamy, "A Generalized Probability Density Function for Double-Bounded Random

Processes," *Journal of Hydrology,* vol. 46, pp. 79-88, 1980.

[106] R. C. Hibbeler, Engineering Mechanics Statics and Dynamics Seventh Edition, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1995.

[107] C. Li, "Investigating the Process of Developing a KDD Model for the Classification of Cases with Cardiovascular Disease Based on a Canadian Database," University of Waterloo, Waterloo, 2012.

[108] H. M. School, "Gender matters: Heart disease risk in women," Harvard Health Publications, [Online]. Available: http://www.health.harvard.edu/newsweek/Gender_matters_Heart_disease_risk_in_women.htm. [Accessed 11 1 2014].

[109] C. Mavriplis, Interviewee, *Associate Professor, Department of Mechanical Engineering, University of Ottawa.* [Interview]. 29 11 2013.

[110] O. Zambrano and C. M. M. M. Rocco S, "Estimating female labor force participation through statistical and machine learning methods: a comparison," *Computational Intelligence in Economics and Finance,* pp. 93-105, 2005.

[111] L. D. Statistics Canada, *Labour Force Survey, February 2013 [Canada] Study Documentation,* Ottawa: Statistics Canada, 2014.

[112] M. Santora, "In Hours, Theives Took $45 Million in A.T.M. Scheme," 9 5 2013. [Online]. Available: http://www.nytimes.com/2013/05/10/nyregion/eight-charged-in-45-million-global-cyber-bank-thefts.html?hp&_r=3&. [Accessed 1 6 2013].

[113] D.-L. Xu, J. Liu, J.-B. Yang, G.-P. Liu, J. Wang, I. Jenkinson and J. Ren, "Inference and learning methodology of belief-rule-based expert system for pipeline leakage," *Expert Systems with Applications,* vol. 32, pp. 103-113, 2007.

[114] S. N. Joshi, SYDE334 Applied Statistics Course Notes, Waterloo: University of Waterloo, 2000.

[115] R. S., I. Michalski and M. K. Bratko, Machine Learning and Data Mining Methods and Applications, West Susses: John Wiley & Sons Ltd, 1998.

[116] A. Salazar, G. Safont, A. Soriano and L. Vergara, "Automatic Credit Card Fraud Detection based on Non-linear Signal Processing," *46th Annual IEEE International Camahann Conference on Security Technology,* pp. 207-212, 2012.