Framework for Adaptive Sequential Pattern Recognition Applied on Credit Card Fraud Detection in the Online Games Industry

Michael Schaidnagel, Thomas Connolly
School of Computing
University of the West of Scotland
B00260359@studentmail.uws.ac.uk
Thomas.Connolly@uws.ac.uk

Fritz Laux
Data Management Lab
Reutlingen University
Fritz.Laux@reutlingen-university.de

Abstract—**Online credit card fraud presents a significant challenge in the field of eCommerce. In 2012 alone, the total loss due to credit card fraud in the US amounted to $ 54 billion. Especially online games merchants have difficulties applying standard fraud detection algorithms to achieve timely and accurate detection. This paper describes the special constrains of this domain and highlights the reasons why conventional algorithms are not quite effective to deal with this problem. Our suggested solution for the problem originates from the fields of feature construction joined with the field of temporal sequence data mining. We present feature construction techniques, which are able to create discriminative features based on a sequence of transaction and are able to incorporate the time into the classification process. In addition to that, a framework is presented that allows for an automated and adaptive change of features in case the underlying pattern is changing.**

Keywords-feature construction, temporal data mining, binary classification, credit card fraud

I. INTRODUCTION

This work is an extension of our work in the field of fraud detection [1]. The approximate global business volume of the computer gaming industry in total rose from $ 79 billion in 2012 to $ 93 billion in 2013 [2]. It is estimated to reach $ 111 billion in 2015. New technology developments, such as browser games and Massive Multiplayer Online Games have created new business models (based on micropayments) for online games merchants. Both, technology and business model affect the customers payment behaviour. Their first choice for performing online payments is the credit card. The downside of this development is an increase in online credit card fraud, which continues to pose a big threat for online merchants. Over all branches, the total loss due to credit card fraud rose to $ 54 billion in 2013 in the US alone [3] and is supposed to increase further. Especially merchants in the online games industry are having difficulties applying standard techniques for fraud detection. The reason for this is the lack of personal information about their customers as well as the need for real time classification. Other online retailers (e.g., for fashion, books) are able to compare the shipment address with the billing address of an order to assess if a suspicious transaction is placed. In addition to that, online retailers have more time for extensive risk checks and can also have the manpower to verify customer´s identify by phone for high amount orders.

Online game merchants dealing with low volumes micro transactions and need to deliver instantly in real time. Therefore, an automated data mining approach needs to be considered in order to deal with fraudulent credit card transactions. The advantage of games merchants is their data situation. Data about the in-game behaviour as well as the earlier transactions can be collected along the way. Therefore, it is obvious to use the so far collected data to determine genuine and fraudulent behaviour patterns.

The collected transaction sequences have a more complex structure that other tuple based data. It is collected over time and incorporates timestamps of the particular items bought. We apply feature construction techniques to incorporate the temporal dimension of previous transactions into the classification process and therefore aid finding distinctive sequential patterns. A sequential pattern is only visible in the course of time [4].

The rest of the paper is structured as follows: Section II gives a short introduction in to the field of temporal sequence data mining to set the frame for the given set of problems. This is followed by Section III, which explains feature construction and feature selection. These research fields are later used to retrieve information from data sequences. The consecutive Section IV will define the problem at hand, introduces problem-related terms and describes the contribution of this work. Section V will then give an overview of the related work, which describes different data mining algorithms that are normally suggested for the given problem set. The algorithms are also part of the experimental evaluation. Section V will detail our suggested method and describe the major components. The suggested method is applied to a real life data set in Section VII to show its classification abilities. Section VIII concludes the results and mentions a few points for future development.

II. FROM DATA MINING TO TEMPORAL DATA MINING

Data Mining is a multidisciplinary research field, which uses methods and knowledge from many areas, such as database technology, machine learning, information theory, statistics, data visualization, artificial intelligence, and computing [5]. This created a *"solid science, with a firm mathematical base, and with very powerful tools"* [6] p. 5. It is the natural result of the evolution of information technology. This development started in the 1980´s when data access techniques began to merge, the relational data model was applied, and suitable programming languages were developed. The following decade included the next significant step in data management: the use of Data Warehouses and Decision Support Systems. They allowed manipulation of data originating from several different sources and supported a dynamic and summarizing data analysis. However, these systems are not able to find more hidden patterns in data-rich but information-poor situations. This requires advanced data analysis tools which are provided by the research field of data mining [7]. In contrary to Data Warehouses and Decision Support Systems, Data Mining is computer-driven. It solves the query formulation problem. This means discovering patterns, which a user is not able to put into a database query or is only able to formulate the problem [7].

The term Temporal Data Mining refers to an emerging research issue [8], that is defined by Lin, Orgun, and Williams [9] p. 83 as *"a single step in the process of Knowledge Discovery in Temporal Databases that enumerates structures (temporal patterns or models) over the temporal data, and any algorithm that enumerates temporal patterns from, or fits models to, temporal data [...]"*. This step increased its importance, since recent advances in data storage enabled companies to keep vast amounts of data that is related to time [9]. Temporal Data Mining is concerned with inferring knowledge from such data. Thereby the following two inference techniques can be applied [9]:

- Temporal deduction: inferring information that is a logical consequence of the information in the temporal database
- Temporal induction: inferring temporal information generalized from the temporal database

According to Kriegel et al. [10], Temporal Data Mining algorithms that are able to find correlations/patterns over time will play a key role in the process of understanding relationships and behaviour of complex objects. Complex objects in this case can be, for example, data sequences that contain several columns. A prime example therefore is transactions in an online shop. All transactions from one customer form a transaction sequence. Each transaction can contain information about the purchase date, the purchased items and the customers address etc. This work will show how behaviour over time can be captured in meaningful features and then be used for real time classification.

III. Feature Selection and feature construction

Today´s data sets can have billions of tuples and thousands of (often useless) attributes [11]. This development is often referred in the literature as the curse of dimensionality. The performance of classification algorithms can deteriorate if the wrong input is given and also the computational costs can increase significantly. The reason for this is the tendency of classifiers to overfit, if provided with misleading information. So in terms of the online shop example from above, the length of the street name of the customers address does not have an effect on the purchase behaviour and needs to be ruled out. In order to do this, feature selection techniques are applied on the given data to decrease data dimensionality. This results in an increase of classification performance and also in a reduction of the execution time.

Feature selection is defined as the process of selecting a subset of attributes from the original dataset, which allows a classifier to perform at least as good as with all attributes as input [11]. There are several different feature selection techniques that can be categorized as supervised, unsupervised and semi-supervised. Supervised techniques utilize the given label while unsupervised do not. In terms of feature selection strategies, there can be a categorization into filter, wrapper or hybrid models [12]. Filter models use certain criteria to assess features and select the features with the highest score. Wrapper models use clustering algorithms to find feature subsets and then evaluate these in terms of their clustering quality. The process is repeated until a suitable quality is reached. The heuristic search strategy is quite computationally expensive compared to the filter model. Hybrid models include a filtering step before the typical wrapper process in order to increase computational efficiency. Our presented work is using a filter model based on two measurements, which are further explained in Section V.B.

Feature construction on the other hand is defined as the process of discovering missing information about the patterns in data by inferring or creating additional attributes in order to aid the mining process [5], [7]. An example for a simple feature construction technique on a two dimensional problem could be the following: assume that A_1 is the width and A_2 is the length of a square. This can be transformed into a one-dimensional problem by creating the feature F as area $F = A_1 * A_2$ [13]. However, the success of feature construction is dependent on the target hypothesis of the Data Mining problem at hand. There is no use in calculating the area as a feature; if the pattern (that needs to be found) is connected to the aspect ratio of the squares. Shafti and Pérez [14] distinguish between two types of features construction techniques in terms of their construction strategy:

- hypothesis-driven: create features based on a hypothesis (which is expressed as a set of rules). These features are then added to the original data set and are used for the next iteration in which a new hypothesis will be tested. This process continues until a stopping requirement is satisfied.
- data-driven methods: create features based on predetermined functional expressions, which are applied on combinations of primitive attributes of a data set. These strategies are normally non-iterative and the new features are evaluated by directly assessing the data.

Our present work is using a data-driven construction strategy since it allows for an automated feature construction process that is not dependant on human intervention.

IV. Problem definition:

The problem of credit card fraud detection involves a number of constraints, which make it difficult to apply traditional algorithms.

Firstly, gamers do not feel comfortable to reveal their real life names and addresses in an online gaming environment. This lack of personal data, in addition to the short transaction histories of players, makes it difficult to apply standard Data Mining techniques.

Secondly, the real time nature of business makes it necessary to be able to apply an algorithm in real time, or near real-time, in order to reject fraudulent transactions at authorization time. Most of the techniques proposed so far are bulk oriented and designed for offline batch processing. Hence, it would be helpful to have a technique that can be integrated into already existing systems. This is also supported by Fayyad, Piatetsky-Shapiro, and P. Smyth [15] p. 49: *"A standalone discovery system might not be very useful"*.

Thirdly, cardinality of the occurrences in a dataset of the given domain is either too high or too low. On the one hand, there can be millions of different credit card numbers involved in the transactions, so that standard algorithms are not able to recognize the sequence structure of transactions of the same credit card. On the other hand, the in-game products sold in the online store can be very few, so that frequent pattern mining algorithms having a hard time if there are, for example, only 6 different products available.

Fourthly, the so far proposed methods are focusing on static vectors of attributes without any temporal evolution. Kriegel et al. [10] argues that due to historical reasons (i.e., given their static data during the 1980´s); many researchers created their algorithms only for static descriptions of objects and are therefore not designed to input data with dynamic behaviour. The inclusion of the dynamic properties of temporal data however, allows unveiling sequential patterns that occur in the course of time. An example for this in the field of credit card fraud detection is the current status of transaction. A transaction that has been approved by the credit card company (i.e., is booked as successful) can later be charged back. This change of status represents vital information that is revealed after some amount of time has passed.

The framework described in this article is able to overcome all the challenges described in this section. Key to success is the understanding of temporal sequence based patterns. However, the framework is currently focusing on the domain of credit card fraud detection, since it also contains domain-specific features (that will be described further down). The next section highlights our contribution to the body of knowledge. That is followed by a description of the characteristics of sequential data. The last subsection of this chapter will give a definition for feature interaction, which is a pattern that can be used by our proposed algorithm.

A. Contributions

There are several hints in the literature, which give suggestions on how to solve such a sequence classification problem that is presented by online credit card fraud: one direction of development in the field of temporal data mining is described by Lin, Orgun, and Williams [9]. They postulate a temporal sequence measure method that allows *"[...] an arbitrary interval between temporal points [...]"* to create *"[...] a very powerful temporal sequence transformation method."* [9] p. 83.

Another direction of development is given by Tsai, Chen, and Chien [16]. According to their article, a sequential pattern classification problem can also be treated as a feature mining problem. This would allow feature mining algorithms to treat extracted patterns as features. However, Yang, Cao, and Liu [8] state that the well-known standard classification algorithms are difficult to be applied on sequential data due to vast number of potential features that can be generated out of a sequence. A solution for this could be to work on a different abstraction layer, i.e., to use a form of aggregation to simplify the data sequence into a row-based vector. This would then allow standard classification algorithms to be applied on complex sequential data.

Simplification is also suggested by Kriegel et al. [10] p. 90: *"Representing complex objects by means of simple objects like numerical feature vectors could be understood as a way to incorporate domain knowledge into the data mining process"*. However, he focuses more on the incorporation of domain expert knowledge into data mining. He postulates a technique to help domain experts *"to use the important features of an object to e.g., classify new objects of the same type, eventually by employing sophisticated functions to transform attributes of some type to features of some other type"* [10] p. 90 and to generalize this domain knowledge to keep pace with more complex ways of mining complex objects.

Our contribution to that research field is a novel algorithm that incorporates these ideas and puts them into one approach. The described feature construction techniques are able to include the time dimension during the aggregation of sequences. This allows using arbitrary time intervals as suggested by Lin, Orgun and Williams [9]. In a later step, the found features are normalized and arranged in a way that enables threshold based classification.

The framework presented in this work is able to handle the difficult data situation in the area of online credit card fraud detection. In addition to that, the framework is able to adapt to changes of the fraudulent behaviour. All necessary steps are described, starting from data preparation to feature construction to applying the right threshold.

In order to assess transactions without any history, a concept of cultural clusters is introduced to help classifying those transactions. In addition to that, a metric for assessing the suitability of features as well as the calculation of the threshold are introduced. The suggested approach was pitched against other algorithms on a real life data set. It was able to perform 16.26 % better than the best standard method (Bayesian Net) and achieves an almost perfect 99.59 % precision.

B. Characteristics of sequential data

Sequential Data is defined in the literature as a series of nominal symbols from a defined alphabet. This list of objects is normally registered within a domain ontology according to Adda et al. [17] as well as Antunes and Oliveira [18]. The term sequence must not be confused with the term time

TABLE I. SCHEMA OF SEQUENTIAL DATA

r	t	s_{id}	a_1	a_2	...	a_i	s_{label}
r_1	t_1	s_{id_1}	a_{1_1}	a_{2_1}	...	a_{i_1}	0
r_2	t_2	s_{id_1}	a_{1_2}	a_{2_2}	...	a_{i_2}	0
r_3	t_3	s_{id_1}	a_{1_3}	a_{2_3}	...	a_{i_3}	0
r_4	t_4	s_{id_2}	a_{1_4}	a_{2_4}	...	a_{i_4}	1
r_5	t_5	s_{id_2}	a_{1_6}	a_{2_5}	...	a_{i_5}	1
...	...	...	...	...	...	...	...
r_m	t_m	s_{id_n}	a_{1_m}	a_{2_m}	...	a_{i_m}	...

series, which is a sequence of continuous, real-valued elements. The research work proposed in this article is focusing on transactional datasets, which include information about the time of the transaction and can be attributed to logical units (i.e., sequences). This logical unit in the field of credit card fraud detection is the customer id. Every action can be represented in a data base as a row r, which has several attributes (i.e., columns). Each row is provided with a timestamp t. The attributes $a_i \in E$ of a row can be associated to a logical unit s_{id} (i.e., the customer_id). There are n sequences s_{id_n} in a data set E. Each sequence s_{id_n} consists of at least one row r. The number of rows in a sequence equals the length of a sequence ls, so that $1 \leq ls \leq m$.

Table I depicts the general schema of sequential data: It is important to differentiate between the number of rows (or tuples) m of a data set and the number of sequences n. Sequence s_1, from the example below, has a length ls = 3 and can be described as matrix such as $s_1 = \begin{pmatrix} a_{1_1} a_{2_1} ... a_{i_1} \\ a_{1_2} a_{2_2} ... a_{i_2} \\ a_{1_3} a_{2_3} ... a_{i_3} \end{pmatrix}$

C. Feature Interaction

The proposed framework is able to use interrelations among attributes of a dataset: It is possible, that the original data is not sufficient to adequately describe such an eventually existing interaction among attributes. Thereby interaction means that *"the relation between one attribute and the target concept depends on another attribute"* [19] p. 246. If the existing dependency is not constant, the interaction is called complex. An example of a complex interaction between two attributes in an instance space is shown below in Figure 1.

The '+' and '-' signs depict the distribution of the class labels of instances. So in the case of the left hand side, instances with a high vale for a_1 and a_2 have the label '-'. Interactions among data in general pose a problem for classifiers, since neither a_1 nor a_2 by itself contains enough information to distinguish between the labels.

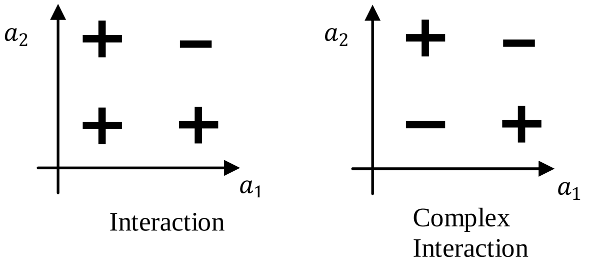

Figure 1. Schematic representation of complex feature interaction, based on Shafti and Pérez [19] p. 246

V. PROPOSED FRAMEWORK

The proposed framework is designed for classification tasks on data sequences consisting of transactions. Originally, the used features were constructed manually and incorporated domain specific knowledge [1]. The used feature construction techniques were automated, enriched, and generalized in a later step, also shown in Schaidnagel and Laux [42]. These techniques (also briefly discussed in Section V.A) are now combined to create an adaptive algorithm that is able to attune to changing fraud behaviour. Figure 2 shows an overview. The framework consists of two systems: the first one processes the credit card transactions and executes the classification (i.e., the fraud / non-fraud decision). The decision is based on a signal value that is calculated using the transaction history of the corresponding user account. The calculation is carried out by the feature assembler, which uses a formula (described in Subsection V.D) that consists of a multitude of features. The features are templates for how to aggregate a given sequence. They are provided from the feature pool, which is kept up to date by the second system.

The second system hosts the feature construction algorithm, which is briefly described in Subsection V.A. After a certain period of time, e.g., a week or a month, the second system uses a sliding window to query for training data, which is used to create new features. They are assessed using feature selection (also described in Subsection V.B). The performance of the newly constructed features is compared to the older ones in the feature pool and replaced if necessary.

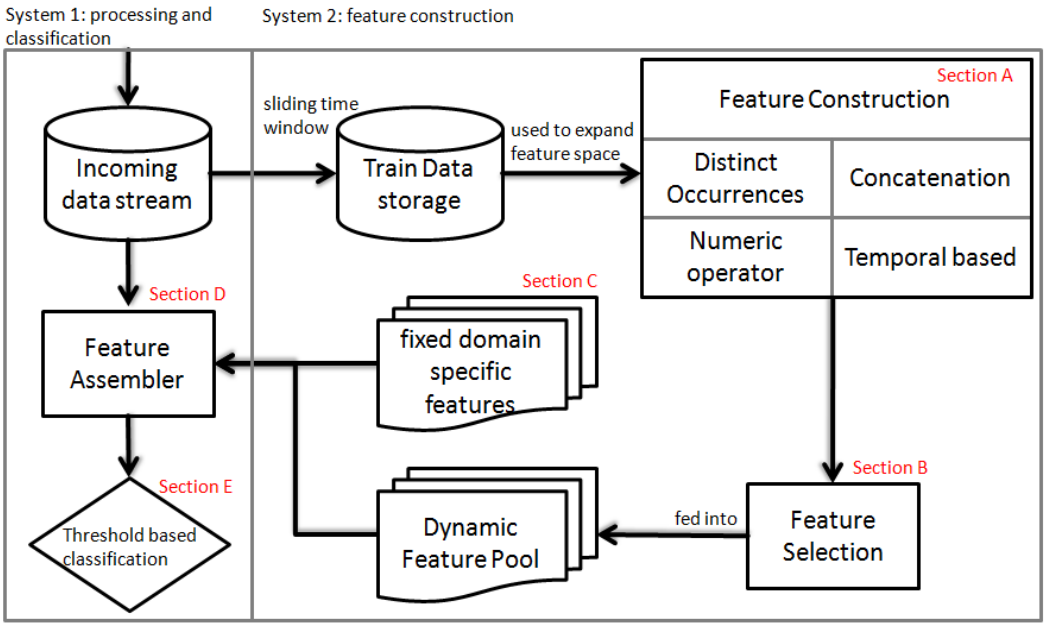

Figure 2. Framework Overview, Section A-E refer to the subsections in the description of Section VI.

A. *Feature Construction for transactional data*

As briefly described above, training data is periodically drawn from the execution environment, which is then used for 'training'. The training process consists of constructing and selecting suitable features that are able to distinguish between the two given labels. The columns of the original data set are called attributes, while the constructed data is called features.

The feature construction techniques that we used for this work utilizes a data-driven approach. It is in detail described in [42]. The data set needs to be annotated in an initial step. This is done by selecting an attribute s_{id} of the original data set that is used as a sequence identifier column for sequence aggregation. It identifies events/objects that can be logically associated to one entity. An example for such an attribute could be the account number or email address of a user. For the domain of credit card fraud detection, we use the term sequence to refer to all transactions belonging to a certain user email address. Please note that in the feature construction step, the transactions of a sequence are sorted by the transactions timestamp prior to aggregation. In a next step the user has to select two more columns: t and s_{label}. The timestamp column t is used to calculate the time elapsed between the collected data points of a sequence. The column s_{label} contains the binary target hypothesis. The label is sequence based, which means that every sequence must only have transaction of the same label value. In the domain of credit card fraud detection this means that all transactions of a user carry the genuine label until one transaction is charged back. Then the label of the sequence (i.e., all associated transactions) changes to fraudulent.

In a next step, we formulated feature construction techniques, which are able to create distinctive features if such a pattern is hidden in the data. We found four different feature construction techniques, which will be briefly described in the following subsections:

1) *Features based on distinct occurences*

A first approach for detecting sequential patterns is to investigate the number of distinct occurrences per sequence. It is possible that one target label has a higher variety in terms of occurrences than the other. This variety can be assessed by aggregating all sequences s_{id_n} of an attribute a_{i_n} and count the number of distinct occurrences, so no duplicates are counted. This can be applied on all string as well as numeric attributes of a data set. The results for all sequences s_{id_n} and attributes a_{i_n} are stored in an intermediate feature table and are assessed in terms of their suitability for classification in step B of the proposed framework.

2) *Concatenation based features*

A way to highlight interactions between two attributes is to concatenate them. Therefore, we systematically concatenate every string attribute in pairs of two and then again, count the distinct value-pairs per sequence identifier. Thereby interactions such as, if a_1 AND a_2 have low value pair variety for label 0, but a high value-pair variety for label 1, are highlighted. Even for data sets with a high number of different occurrences, this kind of feature construction will highlight distinct occurrences between both labels.

This technique can be applied on all string attributes of the given dataset. This simple technique is similar to most common column combinations that are described widely in the literature (e.g., [38], [40], [41]). However, we once again use this technique on a different abstraction layer since we aggregate by the sequence identifier s_{id}.

3) *Numeric operator based features*

Interactions among data can also occur between two numeric attributes. It is possible to capture such a pattern by combining two numeric attributes with basic arithmetic operators such as ”+”, ”-”, ”*” or ”/”. García [39] and Pagallo [37] describe such a technique for feature construction with fewer operators. Our approach incorporates more arithmetic operators and again, uses the sequence identifier attribute to aggregate the constructed features for each sequence. Let us put this into an example: attributes a_i and a_j are combined with the multiplication operator ”*” for a sequence s_{id_1}. The resulting feature $f = a_i * a_j$ is derived from the sequence

$$s_{id_1} = \begin{pmatrix} a_{i_1} & a_{j_1} \\ a_{i_2} & a_{j_2} \\ a_{i_3} & a_{j_3} \end{pmatrix} \quad (1)$$

To construct f we have to multiply each ‘row’ in the sequence and sum up the results: $f = (a_{i_1} * a_{j_1} + a_{i_2} * a_{j_2} + a_{i_3} * a_{j_3})$. If there is an interaction between two attributes for a certain target label, it will affect the resulting sum and can be measured (as described further in Section B.1). This process is repeated for all possible combinations of numeric attributes and for all of the above mentioned arithmetic operators.

4) *Temporal based attributes*

Patterns in sequences can also occur over time. Therefore, we created a feature construction technique that is able to use the time axis, which is incorporated in each sequence by the timestamp column t. This feature construction technique is applicable for both, numeric as well as string attributes. However, for string attributes, there need to be some preparations done, which are explained further down in this subsection. The process for numeric attributes basically multiplies the time interval (e.g., days, hours or minutes), between earliest data point and the current data point with the numeric value of the corresponding attribute, which results in a weighting. A hypothetical example is depicted in Table II.

The example shows two attributes a_i and a_j for two sequences ($s_{id} = 1$ and $s_{id} = 2$) as well as the t column. In order to calculate the temporal based featuref_p for attribute sequence s$_{id}$ $= 1$ in terms of attribute a_i, we first have to calculate the time between the earliest data point of $s_{id} = 1$ and each of the ’current’ data points. This is depicted in Table II by the $\Delta time\ in\ days$ column. The next step is to multiply the value of each t_i in $s_{id} = 1$ with its corresponding delta time value: $(a_{i_1} * 1, a_{i_2} * 11, \ldots, a_{i_4} * 24)$. The sum of this value is the new time based constructed feature f_p .This technique can be applied on all numeric attributes.

TABLE II. EXAMPLE FOR CREATING TEMPORAL BASED FEATURES

s_{id}	t	min(t) / s_{id}	Δ*time in days*	a_i	a_j	s_{label}
1	01.01.2013	01.01.2013	1	a_{i_1}	a_{j_1}	0
1	10.01.2013	01.01.2013	11	a_{i_2}	a_{j_2}	0
1	15.01.2013	01.01.2013	16	a_{i_3}	a_{j_3}	0
1	23.01.2013	01.01.2013	24	a_{i_4}	a_{j_4}	0
2	24.01.2013	01.01.2013	1	a_{i_5}	a_{j_5}	1
2	28.01.2013	01.01.2013	5	a_{i_6}	a_{j_6}	1
2	30.01.2013	01.01.2013	7	a_{i_7}	a_{j_7}	1

To use temporal based feature construction on string attributes, we need to incorporate an intermediate step. During this step we replace the string value by its posterior probability $p(\theta|x)$ (see also Hand [43], pp. 117-118 and pp. 354-356). The posterior probability is the probability of an occurrence a_i, given that its label $s_{label} = 1$ divided by the overall number of that occurrence $p(a_n)$. The probability is based on the distribution of the occurrences in the training data:

$$p(a_i|s_{label} = 1) = \frac{p(a_i|s_{label} = 1)}{p(a_n)} \quad (2)$$

It is possible that there is a pattern within the data that can be characterized by certain occurrences. This means that some occurrences have great tendency towards one of the target labels (i.e., having a high probability for one label). The above described technique allows us to make this pattern visible by multiplying the posterior probability with the temporal axis of the given sequences.

However, it is also possible that the number of distinct occurrences of a string attribute is too high. This will lead to very small posterior probabilities that make it difficult to create meaningful and distinctive features. In such cases, it is recommended to take the logarithm of the posterior probability for cases with high cardinality.

B. *Feature Selection*

The feature construction techniques described in previous section generate a vast amount of features, which need to be assessed if they are useful for classification. Therefore, the next step in our framework (see also Figure 2) is dealing with feature selection.

Feature selection in general is an important step in the KDD process. The performance of classification algorithms can deteriorate, if the wrong input is given and also the computational costs can increase tremendously. Reason for the deterioration in performance is the tendency of classifiers to overfit, if provided with misleading information. In order to avoid this, data miners created methods such as feature selection to decrease dimensionality of the data and as a result of that, increase classification performance and also the execution times.

A supervised filter model (see also Charu and Chandan [12]) is adopted in our framework to find the most suitable features created. There are two measurements that we used for assessing whether a constructed feature is suitable to distinguish between the two given label. The assessment is executed by applying a user defined threshold for both measurements. It is favourable to start with high thresholds, since they allow only the most distinguishing features. This also keeps the feature space, which needs to be constructed during classification, on a manageable level. If the classification performance of the top features is not satisfactory, the threshold can be subsequently lowered.

1) Split

Goal for this feature selection measurement is to find features that are 'in general' suitable for distinction between the two given labels. The average of the features for both groups (i.e., the two given labels) is calculated. The average is a sort of centre for the two clusters. The so-called split value is calculated by measuring the normalized distance between the two cluster-centres as it can be seen in (5)

$$avg_0 = avg(\{f_p \in S | s_{label} = 0\}) \quad (3)$$

$$avg_1 = avg(\{f_p \in S | s_{label} = 1\}) \quad (4)$$

$$split_{f_i} = \frac{|avg_0 - avg_1|}{avg_0 + avg_1} \quad (5)$$

A large distance is thereby favoured. The advantage of calculating the average is that a few false positives within the data do not have such a big impact on the feature selection process. However, average calculation is prone to single extreme or erroneous values if the data is completely unprepared (data normalization would not help in that case).

2) Number of null values

The second feature selection measurement is the number of *NULL* values for each target label. This is a support measurement, which denotes if the achieved split value is based on many sequences or not. So there could be the situation that a constructed feature has a high split value, but might be useless since it cannot be used very often due to large number of *NULL* values for the particular features.

C. Fixed domain specific features

In order to maximize the amount of information retrieved from the transaction history, we incorporate domain specific features to the classification process. The concept of so called cultural clusters was introduced in order to help classifying transactions without any history. The basic idea is to get as much information out of the given attributes as possible. These attributes include the origin of the user (IP country) and the origin of the credit card used in a transaction (BIN country – BIN is an abbreviation for Bank Identification Number: The first 6 digits of a credit card number, enables to locate the card issuing bank of the cardholder). Countries are grouped together by an expert regarding their cultural proximity to each other. The clusters used in this work are roughly based on continents. A range of weights is assigned to each cluster. Every country is assigned with a specific weight within its cluster´s range depending on its cultural distance to its cluster centre and the risk of the county of being defrauded. The weight of a country within a certain cultural cluster is set empirically and can be subject for adaption, in case the fraudulent behaviour changes. In other words: the weight of a country lies within the range of its cultural cluster and is set by an initial value, based on the experience of a fraud expert. If cards from this country turn out to be defrauded frequently, the weight can be increased (within the limits of its cultural clusters). This will increase the risk value of a country pair, which can be calculated as it can be seen in (6):

$$risk = |weight\ (IP\ country) - weight(BIN\ country)| \quad (6)$$

This value will be low for country pairs within the own country cluster (e.g., a user from Sweden tries to use a card originated in Norway) or 0 if the user and the corresponding card are from the same country. On the other hand, this value increases if there is a suspicious country pair involved (e.g., cross-cultural cluster). This simple metric allows depicting complex risk relationships between several countries.

D. Feature Assembler

The previous subsections described how to construct and assess suitable features for the given fraud detection task. This subsection is about how to use these features to make use of feature interaction by assembling the respective features in a certain way. Prior to that, the features are normalized with the min-max normalization [5] to bring them on the same numerical level (ranging from 0 to 1). The Feature Assembler is part of System 1 and is invoked at the time a sequence of credit card transactions need to be classified. It uses the templates of features as input, which are currently held in the Dynamic Feature Pool as well as the fixed domain specific features (see also Figure 2). The templates of the features (i.e., the description on how to construct them), are then applied on the sequences in the incoming data stream that need to be classified. We thereby differentiate between two types of interactive features $f_i \in F$.

The first type of features$f_{i_{nom}}$ tends to 1 if normalized and will be summed up in the nominator of a fraction. The denominator, in contrary, is composed of the second type of features $f_{i_{denom}} \in F$, which tend to 0 if normalized with min-max normalization. If the quotient of the normalization expression is not defined, it will be discarded. The fraction depicted in (7), is used for calculating a signal value that can then be used for binary classification.

The assembling of the interactive features will result in a high signal value if the sequence in question is similar to the average of all sequences of the target label.

$$signal = \frac{\sum f_{i_{nom}}}{\sum f_{i_{denom}}} \quad (7)$$

E. *Threshold selection and application*

The resulting signal value from (7) is an indication on how predominant fraudulent behaviour is in the assessed sequence. As a last step of our framework, a threshold value, whose violation will lead to the classification fraudulent transaction, needs to be defined. This threshold is determined empirically by undertaking a series of experiments with a set of thresholds (e.g., from 0 to 100). We use accuracy metrics such as Precision P, Recall R and score F1 to assess each tested threshold.

Precision P is defined in the literature as (e.g., [44], [45]) the ratio of true positives (TP) and the total number of positives predicted by a model. That is in our case the number of genuine transactions that have been declared to be genuine plus the number of fraudulent transaction that also have been labelled genuine:

$$P = \frac{TP}{(TP+FP)} \tag{8}$$

Recall R on the other hand is defined as the number of true positives divided by the sum of true positives and false negatives. In our example, we have to divide the number of fraudulent transactions detected by our model by the sum of the detected fraudulent transactions (TP) and the not detected fraudulent transactions (FN):

$$P = \frac{TP}{(TP+FN)} \tag{9}$$

The measurement F1 represents the harmonic mean of Precision and Recall and is used to rank the performance of different methods in the experimental evaluation:

$$F1 = 2 * \frac{P*R}{P+R} \tag{10}$$

The development of these performance measures over different threshold values is depicted Figure 3 for an example case.

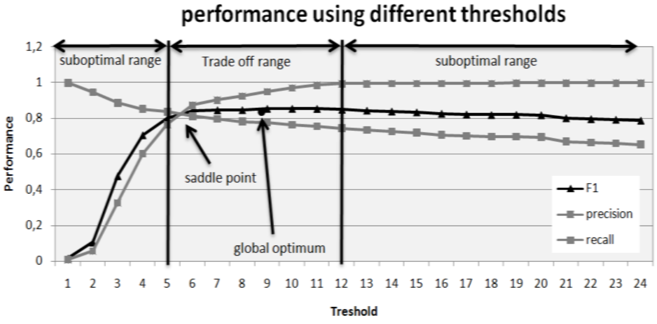

Figure 3. Determining threshold value

Accuracy indicators are increasing fast until threshold value 5. It is not reasonable to select a threshold lower than 5, since the F1 is far from the global optimum. From threshold 5 on, there is an intersection point, which will keep the F1 near the global optimum. This second range can be called "trade-off range" and spans up to threshold value 12, in the case depicted in Figure 3. Within this range the merchant can choose between detecting more fraudsters, including a higher rate of false positives or catching less fraudsters, but increase Precision and therefore avoid false positives. This choice can depend on the ability of the merchant to deal with false positives and on the merchants specific total fraud costs. In the context of fraud detection the term total fraud costs means the sum of lost value, scanning cost as well as reimbursement fees associated with a fraud case.

After a certain threshold value, in the shown case 12, the Precision is almost 1 and will only increase insignificantly. The Recall and consecutively F1, will decrease from that point. Reason for this is the intrinsic mechanic in the used formula. Fraudulent transactions with a comparable low fraud profile will be assigned a lower risk level. This level will hopefully be still higher than the risk level of genuine users. If however, the threshold is set high enough these lower profile fraud cases will be classified incorrectly as genuine. This will cause the Recall and F1 to drop. Hence, it makes no sense to choose a threshold greater than 12.

VI. Related Work

So far, there have been many standard data mining algorithms applied in the field of credit card fraud detection [5]. Please note that we do not go into details here on how they work. All mentioned methods have been implemented and will be compared in terms of fraud detection performance in Section VII.

Artificial Neural Network (ANN): Gosh and Reilly [20] were the first ones to adapt Neural Networks on credit card fraud detection. Other authors such as Dorronsoro et al. [21], Brause et al. [22] and Maes et al.[23] have also implemented ANNs in real life applications. ANNs in general are too dependent on meaningful attributes, which might not necessarily be available. The information gain from such attributes is too low to be utilized in ANNs.

Bayesian Belief Network (BBN): The first implementation for fraud detection was done by Ezawa et al. [24]. Other recent implementations are Lam et al. [23], Maes et al. [23] and Gadi et al. [25]. However, some data set do not provide enough attributes in order to construct a suitable network.

Hidden Markov Model (HMM): In recent years several research groups applied this model for fraud detection. Srivastava et al. [26] have conducted a very systematic and thorough research in their work. Other implementations were done by Mhamane et al. [27], Bhusari et al. [28] as well as Dhok and Bamnote [29]. A classic and comprehensive introduction to the topic of HMM was published by Rabbiner and Juang [30] and also Stamp [31] is worth reading for introductory purposes. HMMs in general are only able to utilize a single numeric attribute for their prediction, which is insufficient for a proper classification.

Decision Tree (DT): The biggest impact on how Decision Trees are built had Quinlan [32] in the late 90s. There have been some applications on fraud detection in recent years, e.g., Minegishi et al. [33]. Other mentionable fraud detection implementations are Sahin and Duman [34], Sherly et al. [35] and Gadi et al. [25]. DTs in general suffer the same insufficiencies as ANNs.

Support Vector Machine (SVM): Li and Sleep [49] use a Support Vector Machine for sequence classification. In essence, they compare similarity using a kernel matrix. Their similarity measure is based on n-grams of varying length. The problem of exploding features generation complexity is alleviated by the use of LZ78 algorithm. The constructed features are not only simple binary presence/absence bits, but so-called relative frequency counts. This is assumed to create a finer grain of features. They also use a weighting scheme to highlight discriminative, but infrequent patterns. Dileep and Sekhar [48] describe an intermediate matching kernel for a SVM to help classification of sequential patterns.

Earlier work in the field of feature construction was done by Setiono and Liu [36]. They used a neuronal network to construct features in an automatic way for continuous and discrete data. Pagallo [37] proposed FRINGE, which builds a decision tree based on the primitive attributes to find suitable Boolean combinations of attributes near the fringe of the tree. The newly constructed features are then added to the initial attributes and the the process is repeated until no new features are created. Zupan and Bohanec [38] used a neuronal net for attribute selection and applied the resulting feature set on the well known C4.5 [32] induction algorithm. Feature construction can also be used in conjunction with linguistic fuzzy rule models. García et al. [39] use previously defined functions over the input variables in order to test if the resulting combination returns more information about the classification than the single variables.

However, in order to deal with the increasing complexity of their genetic algorithm in the empirical part, García et al. only used three functions (*SUM*(x_i,x_j), *PRODUCT*(x_i,x_j), *SUBSTRACT_ABS*(x_i , x_j)) to enlarge the feature space. Another approach to feature construction, which utilizes a genetic algorithm, is described by Sia and Alfred [40]. Although, his approach is not using different functions to create new combinations of features, it can create a big variety of features since it is not limited to binary combination. The method is called FLFCWS (Fixed-Length feature construction with Substitution). It constructs a set that consist of randomly combined feature subsets. This allows initial features to be used more than once for feature construction. That means that it is able to combine more than two attributes at a time. The genetic algorithm selects thereby the crossover points for the feature sequences. Another mentionable contribution to the field of feature construction was done by Shafti and Pérez [41]. They describe MFE3/GA, a method that uses a global search strategy (i.e., finding the optimal solution) to reduce the original data dimension and find new non-algebraic representations of features. Her primary focus is to find interactions between the original features (such as the interaction of several cards in a poker game that form a certain hand).

Lesh, Zaki and Ogihara [46] present FeatureMine - a feature construction technique for sequential data. It combines two data mining paradigms: sequence mining and classification algorithms. They understand sequences as a series of events. Each event is described by a set of predicates, e.g. AB --> B --> CD. There is also a timestamp associated with each event. FeatureMine starts by mining frequent and strong patterns. Frequency is defined by a threshold that is specified by the user. Strong is defined as a confidence level that needs to be over a user specific threshold. The found sequences are pruned and selected using some heuristics. The prevailing sequences lattices are stored in a vertical database layout. The constructed features have been feed into the Winnow and Naive Bayes classification algorithms. However, this approach only creates frequent itemsets and is not applicable on transactional data.

Shafti and Pérez [47] present MFE3/GA, which is a feature construction technique that is able to detect and encapsulate feature interactions. The encapsulation is what allows classifiers to deal with interacting features. MFE3/GA in essence searches through the initial space of an attribute subsets to find subset of interaction attributes as well as a function over each of the found subsets. The suitable functions are then added as new features to the original data set. The C4.5 learner is then applied for the data mining process. So far only nominal attributes are being processed, so that class labels and binary/continuous attributes need to be normalized. A feature is in this context is a bit-string of length N, where each bit shows the presence or absence of one of the N original attributes. This form of representation reduces the complexity if elaborate features are constructed. The number of subsets within each feature is limited by a parameter, which is defined by the user. The bit representation of data is not sufficient to model real-world transactional data.

VII. Experimental Evaluation

The performance of the proposed feature construction techniques are compared to the standard techniques, mentioned in the related work. This comparison is based on real credit card fraud data, which was thankworthy provided by a successful gaming company in the online games market. Unfortunately, the given data did not span a long enough time frame to show the adaption capabilities of the presented feature assembler. Hence, Subsection VII.C shows a synthetic example for a pattern that is changing over the course of time.

A. Data Set

The given credit card fraud data set comprises of 156,883 credit card transactions from 63,933 unique users. The records in the data set have the schema as it can be seen in Table III. Due to the high number of occurrences in several

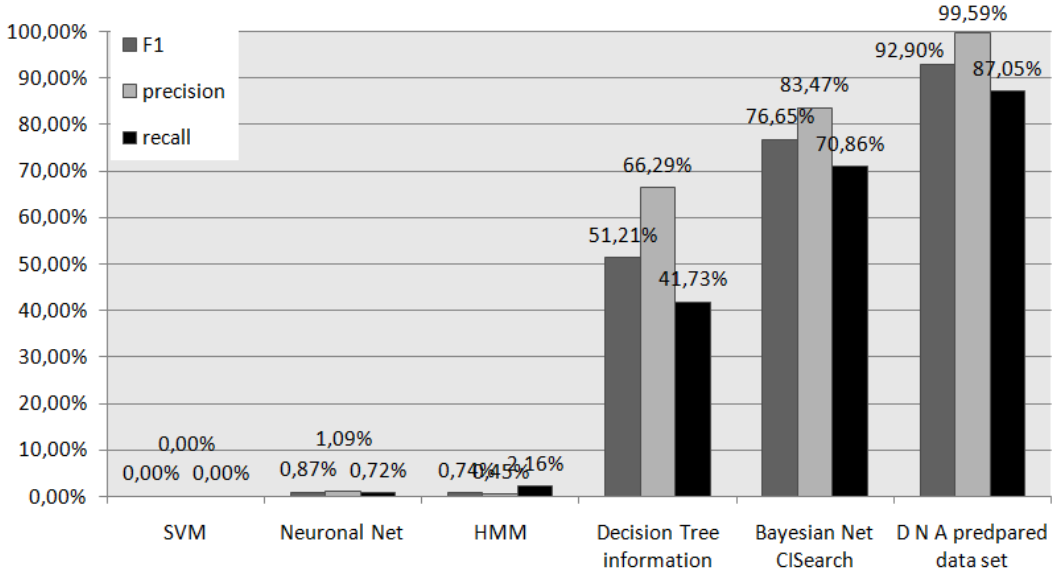

Figure 4. Fraud detection performance comparison

TABLE III. FULL DATA SET SCHEMA

column Name	description
created	timestamp of the payment transaction
user_signuptime	Time a user startet the game
creditcard_token	identifies credit card, hashed
card_bin	Bank Identification Number
user_country	user´s land of origin
user_id	User identification number
user_email	hashed for privacy compliance
transaction_amount	Volume of purchase
order_payment_status	Status of order

TABLE IV. ADDITIONAL FEATURES OF THE PREPARED DATA SET

column Name	description
bin_country	2 letter country code derived from card_bin
days_since_signup	integer attribute calculated as difference from signup_time to created
total_count	denotes total transaction figure for a particular user_email
package	a single letter attribute ranging from A to E. It was derived from the offer_price attribute to reduce the cardinality of the offer_price attribute

columns as well as the lack of distinctive attributes, most of the standard algorithms were not applicable on that data set right away. In order to get a fair comparison and to overcome these obstacles, several adaptations to the data set were made. The resulting prepared data has a minimum sequence length of three (smaller sequences have been discarded) and four derived attributes (see Table IV) were added.

The prepared data set comprised of 13,298 unique users, which are accompanied by 46,516 transactions. The data label distribution in the data set is heavily skewed, which means that there are ~99 % of genuine transactions. The last transaction of each user was cut out in order to form the test data set. This procedure segmented the prepared data set into 71.4 % train data and 28.58 % test data.

B. Fraud Detection Performance

All tests in this section were performed using the prepared data set. We used the F1 score in order to rank the compared methods. As shown in Figure 4, the proposed approach is able to perform 16.25 % better than the best standard method, which is the Bayesian Net. The SVM was not able to detect the pattern within the rather short sequences. Hence, it defaulted and predicted genuine for all transaction. This resulted in a F1 measure of 0.00 %. Main reason for the poor performance of HMM´s in credit card fraud detection for online gaming merchants is the very low sequence length. These models are successful at credit card issuing banks since their sequence length enfolds the entire history of the cardholder. The HMM is also not able to properly use the time elapsed during the transactions. The neural nets were performing poor due to their focus on just the tuple level of the underlying data. They were not able to incorporate the sequence dimension into the model.

There have also been additional experiments with various combinations of neurons and different learning rates were carried out. All experiments resulted in the weak performance as shown above. The algorithm proposed in this article is also able to achieve an almost perfect 99.59 % Precision, which is especially valuable for online gaming merchants, since it reduces the risk of punishing genuine users and consecutively reduces the risk of reputation loss.

C. Adaptive Framework example

This subsection describes a small example that will show the adaptive capabilities of the proposed framework. To keep things simple, we left out the sequential part of the construction algorithm and just focus on a two dimensional problem (rectangle classification).

Assume that we are given data about rectangles as shown in Table V. Given are the two attributes width and length of the rectangles as well as the label column. Goal is to be able to distinguish between the two given classes. We can then use this data to create several features, as described in Section A. Please note that for the sake of simplicity in this case we only create some operator numeric based features. The constructed features are depicted in Table VI. The constructed features are assessed by calculating the split value, as described in Section B and shown in Table VII. The highest split values have features f_3 and f_4. Reason for this is that these features are capturing the underlying pattern behind the labels: the blue rectangles are 'laying' (length <width) while the orange rectangles are 'standing' (length > width).

So we can now use features f_3 and f_4 for classification of new rectangles where the label is unknown (e.g., we set up a threshold of 1 for feature f_4 so that $f_4 \geq 1 \rightarrow blue$). The blueprint of f_4 and the threshold are forwarded to the feature assembler, which is used to execute the classification.

Now let us assume that some time has passed and new training data is handed into the features construction system. Table VIII depicts the new data. Again, we apply the feature construction algorithm on the data (depicted in Table IX) and calculate the split value (depicted in Table X). It can be seen that features f_3 and f_4 are not suitable anymore to distinguish between the two given labels. Reason for this is that the underlying pattern has changed. It seems like that the area of the rectangles is now useful for classification. The proposed framework can adapt to this change by sending the blueprints for feature f_1 to the feature assembler, which will then use the new selected features for classification.

TABLE V. RECTANGLE EXAMPLE DATA

id	width a_1	length a_2	label
#1	2	3	orange
#2	4	1	blue
#3	2	2	blue
#4	3	2	blue
#5	1	3	orange

TABLE VI. CONSTRUCTED FEATURES

id	width a_1	length a_2	label	$f_1 = a_1 * a_2$	$f_2 = a_1+a_2$	$f_3 = a_1-a_2$	$f_4 = a_1/a_2$
#1	2	3	orange	6	5	-1	0.67
#2	4	1	blue	4	5	3	4
#3	2	2	blue	4	4	0	1
#4	3	2	blue	6	5	1	1.5
#5	1	3	orange	3	4	-2	0.33

TABLE VII. CALCULATING SPLIT VALUE

average	$a_1 * a_2$	a_1+a_2	a_1-a_2	a_1/a_2
blue	4.67	4.67	1.33	2.17
orange	4.5	4.5	-1.5	0.5
split	0.17	0.17	2.83	1.67

TABLE VIII. NEW TRAINING DATA

id	width a_1	length a_2	label
#6	2	1	orange
#7	3	2	orange
#8	5	3	blue
#9	6	2	blue

TABLE IX. NEW CONSTRUCTED FEATURES

id	width a_1	length a_2	label	$f_1 = a_1 * a_2$	$f_2 = a_1+a_2$	$f_3 = a_1-a_2$	$f_4 =a_1/a_2$
#6	2	1	orange	2	3	1	2
#7	3	2	orange	6	5	1	1.5
#8	5	3	blue	15	8	2	1.67
#9	6	2	blue	12	8	4	3

TABLE X. NEW FEATURE SELECTION

average	$a_1 * a_2$	a_1+a_2	a_1-a_2	a_1/a_2
blue	13.5	8	3	2.33
orange	4	4	1	1.75
split	9.5	4	2	0.58

VIII. CONCLUSION

Data pre-processing and selection are very important steps in the data mining process. This can be challenging, if there is no domain expert knowledge available. The framework proposed in this work aims to give guidance on how to systematically find knowledge in data by using an automated feature construction algorithm. In addition to that it shows how these features can be used for binary classification. The proposed automated feature construction algorithm is able to systematically find and assess suitable sequence based features for binary classification tasks. It thereby is able to utilize the time dimension in a sequence of actions in order to access information, which can have a significant impact on the discriminatory power of features. The feature assembling formula is an efficient way to store discovered patterns and use them without starting each time from scratch when a new transaction is added to the sequence.

The framework was applied on the problem of credit card fraud detection in online games. The problem is caused by the lack of useful financial data, the anonymity in online games as well as the comparably short transaction sequences. In addition, a domain specific concept of country clusters is used to evaluate the legitimacy of a transaction. The proposed techniques were able to perform 16.25 % better than the best standard method (Bayesian Net) and achieve 99.59 % Precision. The achieved Recall rate (87.05 %) reduced the probability for false negatives and therefore the need for human intervention is reduced.

Future Work: The next steps in the development of the proposed algorithms and its associated techniques, is to apply it on other domains with similar specifications. Intrusion detection in networks or detecting DDOS attacks are both fields in which few attributes are available and behaviour over time is important.

The further development of the feature construction algorithm comprises of the implementation of further mathematical functions into the construction process. So it is possible to generate features with logarithm or exponential powers. It would also be possible to create features based on more than two attributes.

In terms of feature alignment, it would also be helpful to incorporate the sequence length into the algorithm. The algorithm may be susceptible to the sequence length due to the proposed additive technique depicted in Formula (7). The used data set did not allow us to precisely quantify possible impacts.

References

[1] M. Schaidnagel and F. Laux, “DNA: an online algorithm for credit card fraud detection for games merchants,” in *The Second International Conference on Data Analytics*, pp.1-6 (2014, Jan. 09).

[2] R. van der Meulen, Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013. Available: http://www.gartner.com/newsroom/id/2614915 (2013, Feb. 03).

[3] F. Briggs, Fraud costs US retailers $54bn a year, according to new KountVolumatic 2013 survey. Available: http://www.kount.com/_blog/Press_Coverage/post/fraud-costs-usretailers-54-year-according/ (2013, Feb. 03).

[4] Y.-H. Hu, F. Wu, and C.-I. Yang, “Mining multi-level time-interval sequential patterns in sequence databases,” in *Software Engineering and Data Mining (SEDM)*, 2010 2nd International Conference on, pp. 416–421.

[5] J. Han, M. Kamber, and J. Pei, *Data mining: Concepts and techniques*, third edition, 3rd ed. Waltham, Mass: Morgan Kaufmann Publishers, 2012.

[6] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition. Permissions Oxford, UK: permissions: Elsevier Inc, 2005.

[7] A. Symeonidis and P. Mitkaas, Agent Intelligence Through Data Mining: Data Mining and Knowledge Discovery: A Brief Overview: Springer US, 2005.

[8] Y. Yang, L. Cao, and L. Liu, “Time-sensitive feature mining for temporal sequence classification,” in *11th Pacific Rim International Conference on Artificial Intelligence*, pp. 315–326.

[9] W. Lin, M. A. Orgun, and G. J. Williams, “An overview of temporal data mining,” in *Proceedings of the 1st Australian data mining workshop (ADM02)*. Canberra, Australia, vol. 2002, pp. 83–90.

[10] H.-P. Kriegel, K. M. Borgwardt, P. Kröger, A. Pryakhin, M. Schubert, and A. Zimek, “Future trends in data mining,” in *Data Min Knowl Disc*, vol. 15, no. 1, pp. 87–97.

[11] K.J. Cios, R.W. Swiniarski, W. Pedrycz, and L.A. Kurgan, Eds, The Knowledge Discovery Process: *A Knowledge Discovery Approach*. US: Springer US, 2007.

[12] A. Charu and R. Chandan, Eds, *Data Clustering: Algorithms and Applications: Feature Selection for Clustering: A Review*. CRC: Chapman and Hall, 2012.

[13] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Dordrecht: Kluwer Academic Publishers, 1998.

[14] L. S. Shafti and E. Pérez, “Genetic approach to constructive induction based on non-algebraic feature representation,” in *5th International Symposium on Intelligent Data Analysis*, IDA 2003, Berlin, Germany, August 28-30, 2003, pp. 599–610.

[15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Magazine*, pp. 36–54.

[16] C.-Y. Tsai, C.-J.Chen, and C.-J. Chien, “A time-interval sequence classification method,” *Knowl. Inf. Syst.*, vol. 37, no. 2, pp. 251–278.

[17] M. Adda, P. Valtchev, R. Missaoui, and C. Djeraba, “On the discovery of semantically enhanced sequential patterns,” in *Machine Learning and Applications*, 2005, pp. 383–390.

[18] C. Antunes and M. L. Oliveira, “Temporal data mining: an overview,” in *Workshop on Artificial Intelligence for Financial Time Series Analysis*, pp. 1–13.

[19] L. S. Shafti and E. Pérez, “Machine learning by multi-feature extraction using genetic algorithms,” in *Advances in Artificial Intelligence – IBERAMIA 2004*, pp. 246–255.

[20] S. Ghosh and D. Reilly, Eds, *Credit card fraud detection with a neural-network*, 1994.

[21] J. Dorronsoro, F. Ginel, C. Sgnchez, and C. Cruz, “Neural fraud detection in credit card operations,” in *IEEE Trans. Neural Netw*, Vol. 8, No. 4, pp. 827–834.

[22] R. Brause, T. Langsdorf, and M. Hepp, “Neural data mining for credit card fraud detection,” in *Tools with Artificial Intelligence*, 1999, pp. 103–106.

[23] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, “Credit card fraud detection using bayesian and neural networks,” in *Proceedings of the 1st international asian congress on neuro fuzzy technologies*, 2002.

[24] K. Ezawa and S. Norton, “Constructing bayesian networks to predict uncollectible telecommunications accounts,” in *IEEE Expert*, vol. 11, no. 5, pp. 45–51.

[25] M. F. A. Gadi, X. Wang, and A. P. d. Lago, “Comparison with parametric optimization in credit card fraud detection,” in *Machine Learning and Applications*, 2008. ICMLA '08. Seventh International Conference on, pp. 279–285.

[26] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, “Credit card fraud detection using hidden markov model,” in *IEEE Trans. Dependable and Secure Comput*, Vol. 5, No.1, pp. 37–48, 2008 (2013, Nov. 21).

[27] S. Mhamane and L. Lobo, “Fraud detection in online banking using HMM,” in *International Proceedings of Computer Science & Information Technology*, 2012, *International Proceedings of Computer Science and Information Technology*, Vol. 37, pp. 200–204.

[28] V. Bhusari and S. Patil, “Study of hidden markov model in credit card fraudulent detection,” in *IJCA*, Vol. 20, No. 5, pp. 33–36.

[29] S. S. Dhok and G. R. Bamnote, “Credit card fraud detection using hidden markov model,” in *International Journal of Advanced Research in Computer Science*, Vol. 3, No. 3, 2012.

[30] L. Rabiner and B. Juang, “An introduction to hidden markov models,” in *IEEE ASSP Mag*, Vol. 3, No. 1, pp. 4–16, 1986 (2013, Nov. 26).

[31] M. Stamp, “A revealing introduction to hidden Markov models,” Department of Computer Science San Jose State University.

[32] J. R. Quinlan, *C4. 5: programs for machine learning*, Morgan Kaufmann Publishers, 1993.

[33] T. Minegishi and A. Niimi, “Detection of fraud use of credit card by extended VFDT,” in *Internet Security (WorldCIS)*, World Congress on, 2011, pp. 152–159.

[34] Y. Sahin and E. Duman, “Detecting credit card fraud by decision trees and support vector machines,” in *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, 2011.

[35] K. K. Sherly and R. Nedunchezhian, “BOAT adaptive credit card fraud detection system,” in *Computational Intelligence and Computing Research (ICCIC)*, 2010 IEEE International Conference on, pp. 1–7.

[36] R. Setiono and Huan Liu, “Fragmentation problem and automated feature construction,” in *Tools with Artificial Intelligence*, Proceedings. Tenth IEEE International Conference on, 1998, pp. 208–215.

[37] G. Pagallo, “Learning DNF by Decision Trees,” in *IJCAI*, 1989, pp. 639–644.

[38] B. Zupan, M. Bohanec, J. Demsar, and I. Bratko, “Feature transformation by function decomposition,” in *IEEE Intell. Syst*, Vol. 13, No. 2, pp. 38–43.

[39] D. Garcia, A. Gonzalez, and R. Perez, “A two-step approach of feature construction for a genetic learning algorithm,” in *Fuzzy Systems (FUZZ)*, 2011 IEEE International Conference on, pp. 1255–1262.

[40] F. Sia and R. Alfred, “Evolutionary-based feature construction with substitution for data summarization using DARA,” in *4th Conference on Data Mining and Optimization* (DMO), 2012, pp. 53–58.

[41] L. S. Shafti and E. Pérez, “Data reduction by genetic algorithms and non-algebraic feature construction: A Case Study,” in *Eighth International Conference on Hybrid Intelligent Systems*, pp. 573–578.

[42] M. Schaidnagel and F. Laux, “Feature construction for time ordered data sequences,” in *Proceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications*, 2014, Jan. 10.

[43] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining (adaptive computation and machine learning)*, The MIT Press.

[44] Claude Sammut and Geoffrey I. Webb, *Encyclopedia of Machine Learning*: Precision and Recall. US: Springer, 2010.

[45] C. Goutte and E. Gaussier, “A probabilistic interpretation of precision, recall and f-Score, with Implication for Evaluation,” in *27th European Conference on IR Research*, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings, pp. 345–359.

[46] N. Lesh, MJ. Zaki and M. Ogihara, “Scalable feature mining for sequential data,” in *IEEE Intelligent Systems*, Vol 2, 2000, pp. 48-56.

[47] L. S. Shafti and E. Pérez, “Feature construction and feature selection in presence of attribute interaction,” in *4th International Conference HAIS 2009*, Salamanca, Spain, June 10-12, 2009, pp. 589-596.

[48] A. D. Dileep and C. Chandra Sekhar, “HMM based intermediate matching kernel for classification of sequential patterns of speech using support vector machines,” in *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 12, December 2013, pp. 2570-2582.

[49] M. Li and R. Sleep, “A robust approach to sequence classification,” in *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, 2005, pp. 5-8.