engineering proceedings

MDPI

Proceedings

Anomaly and Fraud Detection in Credit Card Transactions Using the ARIMA Model †

Giulia Moschini [1], Régis Houssou [1], Jérôme Bovay [2] and Stephan Robert-Nicoud [1,*]

[1] HEIG-Vd, Haute Ecole Spécialisée de la Suisse Occidentale (HES-SO), Rue de Galilée 15, CH-1400 Yverdon-les-Bains, Switzerland; giulia.moschini@heig-vd.ch (G.M.); regis.houssou@heig-vd.ch (R.H.)
[2] NetGuardians SA, Avenue des Sciences 13, CH-1400 Yverdon-les-Bains, Switzerland; jerome.bovay@netguardians.ch
* Correspondence: stephan.robert@heig-vd.ch

† Presented at the 7th International conference on Time Series and Forecasting, Gran Canaria, Spain, 19–21 July 2021.

Abstract: This paper addresses the problem of the unsupervised approach of credit card fraud detection in unbalanced datasets using the ARIMA model. The ARIMA model is fitted to the regular spending behaviour of the customer and is used to detect fraud if some deviations or discrepancies appear. Our model is applied to credit card datasets and is compared to four anomaly detection approaches, namely, the K-means, box plot, local outlier factor and isolation forest approaches. The results show that the ARIMA model presents better detecting power than that of the benchmark models.

Keywords: anomaly; fraud; ARIMA; isolation forest; K-means

check for updates

Citation: Moschini, G.; Houssou, R.; Bovay, J.; Robert-Nicoud, S. Anomaly and Fraud Detection in Credit Card Transactions Using the ARIMA Model. *Eng. Proc.* **2021**, *5*, 56. https://doi.org/10.3390/engproc2021005056

Published: 16 July 2021

1. Introduction

In recent years, there has been a dramatic increase in the use of credit cards as a means of payment due to their ease of use and convenience. As a response to this phenomenon, fraudsters are also adapting their malicious activities to take advantage of the situation. The extent of this issue is significant; according to the Fifth report on card fraud published by the European Central Bank [1], the total value of fraudulent transactions in the SEPA area in 2016 amounted to EUR 1.8 billion. According to the Nilson Report, a publication covering global payment systems, the total loss due to frauds in 2018 amounted to USD 27.85 billion, and it is projected to reach USD 35.67 billion in 2023 [2]. More specifically, a transaction is said to be fraudulent when it is committed by an unauthorised party and without the rightful owner and/or relevant institution knowing [3]. In these cases, fraudsters could use the card for their personal interests, depleting its resources or until they are caught or the card is blocked. This issue has sparked the interest of both academia and industry, where individuals are working to identify solutions to this problem and to keep up with the ever-changing approaches adopted by malicious players [4]. Credit card fraud detection is now an active field of research, and it particularly hinges on the concept of automation; it is in fact not always feasible or possible to manually review each transaction in order to establish its nature [5]. In addition to this, it is also important to consider that there is another significant human component that could make or break the attempt of a fraudster to successfully exploit a card: the promptness of the cardholders in reporting a stolen, lost or suspiciously used card [5]. This requires the implementation of automated tools for smarter and faster detection of frauds, which has resulted in machine learning techniques being increasingly tested and implemented [6]. Various popular algorithms have been tested in this context, such as random forest, logistic regression, decision trees, support vector machines (SVM), and neural networks [7–9]. Khare and Sait in [7] compare logistic regression, SVM, decision tree and random forest using the Kaggle dataset for credit cards containing 284,807 transactions, 492 of which are fraudulent. The features of the dataset are obtained using principal component analysis (PCA) on the original data for confidentiality issues. The authors also state that they use the behavioural characteristics of the owner of the card, which are shown by a variable representing the spending habits of the customer as well as the month, hour of the day, geographical location and type of merchant. Experimental results show that random forest is the algorithm with the best performance, with an accuracy score of 98.6% compared to 97.7% of logistic regression, 97.5% of SVM and 95.5% of decision tree. Varmedja et al. in [8] compare the performances of logistic regression, naive Bayes, random forest and multi-layer perceptron on the Kaggle dataset. The number of features is reduced through the application of feature selection and the class imbalance addressed by oversampling with SMOTE. Their results show that random forest is again the best algorithm , with accuracy, precision and recall equalling 99.06%, 96.38% and 81.63%, respectively. Roy et al. in [9] use a deep learning approach to detect frauds in credit card transactions. The dataset used in the study was provided by a financial institution and contains almost 80 million anonymised transactions performed over a period of 8 months. The authors perform feature engineering to apply field knowledge to the problem and add extra features to the original ones. Due to the unbalanced nature of the dataset, the authors also perform under-sampling at the account level for each unique account ID. artificial neural networks (ANN), recurrent neural networks (RNN), long short-term memory (LSTM) and gated recurrent unit (GRU) are compared in this study; the results highlight that GRU presents the best performance with an accuracy score of 91.6%, followed by 91.2% (LSTM), 90.4% (RNN) and 88.9% (ANN). As can be noted, there is a common fundamental issue in these approaches: the unbalanced nature of the datasets. In the context of credit card fraud detection, it is in fact expected that the dataset will be very unbalanced, which greatly hinders the performance of supervised learning techniques [6]. Another issue involves the lack of properly labelled data, which again represents a substantial obstacle. Finally, many models lack the adaptability required to take into account the fact that the spending behaviour of customers is likely to change over time [6]. In order to tackle these problems, we propose a model that does not require the knowledge of ground truths and that is designed to make the spending behaviour of the customer as the main source of information when categorising transactions as either legitimate or fraudulent. More specifically, we frame the problem as an anomaly detection task in time series, where the variable represented by the time series is the daily count of transactions for a given customer. We propose a method making use of the ARIMA model and of a rolling windows approach to flag suspicious number of transactions as anomalies, which are discussed in depth in the following sections.

2. Fraud Detection with Time Series Approach

2.1. ARIMA Model with Time Series Analysis

Two widely used models for time series are the *autoregressive* (AR) and the *moving average* (MA) models, which can be used together as an *autoregressive moving average* (ARMA) model. ARMA(p, q) is the combination of the AR(p) and MA(q) models, and can be used with univariate time series.

- *Autoregressive Model*
  The AR(p) model is defined by the equation below; it assumes that there is a dependent linear relation between the observation and the values of a specified number of lagged (previous) observations plus an error term.

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \omega_t \tag{1}$$

  where $\phi = (\phi_1, \phi_2, ..., \phi_n)$ are the coefficients of the model, p is a non-negative integer, c is a constant and $\omega_t \sim N(0, \sigma^2)$.
- *Moving Average Model*

The MA(p) model is defined by the equation below; it makes use of the dependency between an observation and the residual errors resulting from the application of a moving average model to lagged observations.

$$X_t = \mu + \sum_{j=1}^{q} \theta_j \omega_{t-j} + \omega_t \tag{2}$$

where μ is the mean of the series, $\theta = (\theta_1, \theta_2, ..., \theta_n)$ are the coefficients of the model, and q is the order and $\omega_t \sim N(0, \sigma^2)$.

The ARMA model, resulting from the combination of these two models, is defined as follows:

$$X_t = c + \omega_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \omega_{t-j} \tag{3}$$

where p refers to the order of the AR model and q refers to the order of the MA model. The main assumption in time series analysis is that the time series is stationary, meaning that its mean and variance are constant over time; however, this is not the case in many practical situations [10]. The solution to this can be found in the generalisation of the ARMA model: the autoregressive integrated moving average(ARIMA) model. ARIMA introduces the possibility to apply differencing to the data points of time series in order to make it stationary [10]. ARIMA is now one of the most popular, flexible and simple models to fit a time series [10]; it is defined as ARIMA(p, d, q), where p and q represent the orders of the AR and MA models and d indicates the degree of differencing. In the context of fraud detection, time series can be used as a tool when working with aggregated features. Aggregation is often used to derive new features from the original ones in order to feed to the model some information that is thought of and expected to be more relevant than the features per se. The number of daily transactions or the total amount spent in a week are examples of aggregated features [5].

2.2. Estimation Process of ARIMA

When using ARIMA, care should be taken to identify the combination of parameters that best represents the data; Box–Jenkins is a method proposed by George Box and Gwilym Jenkins in [11] that is frequently used when tuning an ARIMA model. The method is composed of three steps:

1. *Identification*, which refers to the use of all available data and related information to select the model that best represents the time series. This phase should, however, be split into two sub-steps:
   (a) Differencing
   The first step requires the establishment of whether the time series is stationary or not in order to determine whether it requires differencing. The augmented Dickey–Fuller (ADF) test is a technique that can be used to verify if the time series on hand is stationary. The null hypothesis of the ADF test states that the time series can be represented by a unit root, meaning it presents a time-dependent structure and that it is, thus, not stationary; consequently, rejecting the null hypothesis implies that the time series is stationary.
   (b) Configuration of p and q
   During this phase, it is helpful to use the correlogram to visualise the autocorrelation function (ACF) and the partial autocorrelation function (PACF) that can help to determine a suitable choice for the orders p and q. The fundamental difference between the two functions is that the PACF removes the linear dependence between the intermediate variables in order to return only the correlation between the present and lagged value. Briefly, whereas the autocorrelation function of AR(p) tails off, its partial autocorrelation function

cuts off after the lag p. Conversely, the autocorrelation function of MA(q) has a cut-off after the lag q, while its partial autocorrelation function tails off.

2. *Estimation*, which refers to the training phase. Once the values of p, d, q have been established, the ϕ and θ coefficients can be estimated. This method uses the maximum likelihood estimation process, which is solved by non-linear function maximisation; for more details about this phase, the reader is referred to [11,12].
3. *Diagnostics*, which refers to the evaluation of the model and identification of improvements. This step involves the determination of issues in the model to verify whether it is able to effectively summarise the underlying data. The forecast residuals provide an important source of information for diagnostics. In an ideal model, the error will resemble white noise and will be normally distributed with a mean of 0 and a constant variance. In addition to this, an ideal model would also leave no temporal structure in the residuals, as they should have been learned.

2.3. Fraud Detection with ARIMA Model on Daily Counts of Transactions

Our idea is to use ARIMA on time series representing the daily count of transactions for a given customer to detect frauds. This is based on an important point: we assume that the number of daily transactions for a given customer follows a certain pattern [13]. On a high level, the task of fraud detection in this context is based on the assumption that it is possible to recognise, and hence model, the regular spending behaviour of the customer; once this has been learned, any discrepancies and deviations from it would be likely frauds. We can also refer to such deviations as *anomalies*. An anomaly is a point in a dataset whose characteristics are significantly different compared to the other points; building from this, anomaly detection is the process to isolate such points by determining when they are deviating from the expected behaviour [14]. ARIMA is used to try to model the legitimate spending behaviour of the customer and to produce a forecast. The intuition behind this setting can be easily explained graphically. Figure 1 shows the daily transactions of a credit card for a customer chosen in our dataset; more details about this dataset are given in the next section. The number of legitimate transactions occurring each day for such customer is represented by the blue dot, whereas the number of frauds is represented by the red dot. A significant peak is observed at the same day of fraudulent transactions.

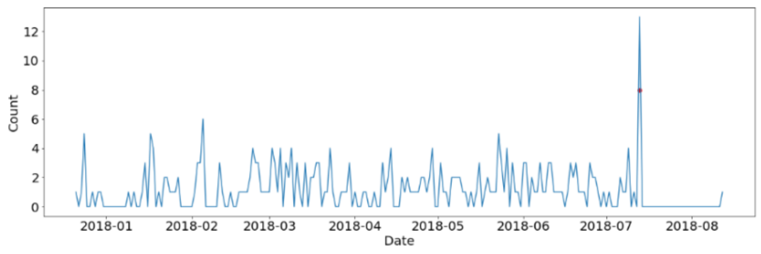

Figure 1. Plot of daily number of transactions for a customer in the dataset. Legitimate transactions are represented by the blue dot, whereas fraudulent transactions are represented by the red dot.

Based on this information, it could be argued that an anomaly detection approach based on the identification of anomalous counts of daily transactions may lead to the detection of frauds. In order to detect frauds, the following steps are proposed:

1. The time series is split into training and testing set; it is important that the training set only contains legitimate transactions so that the model can learn the legitimate behaviour of the customers. This should then allow for the identification of anomalies.
2. In the training set, based on the legitimate transactions, the order of the ARIMA model is identified using the Box–Jenkins method, and, then, the parameters of ARIMA are estimated. During this phase, care is taken to ensure that the estimated coefficients are significant and that there is no temporal structure left in the residuals. Finally, in the testing set, one-step ahead prediction is performed using rolling windows.

3. In order to detect fraud in the testing set, the errors are calculated in terms of difference between the predicted and actual daily count of transactions. Then, the Z-Scores are computed and used to flag the anomalies (i.e., the frauds). The Z-Score is calculated as z-score $= \frac{x-\mu}{\sigma}$, where x is the prediction error on the daily count of transaction in the testing set. μ and σ are the mean and the variance based on the errors of in-sample prediction on the basis of the training set using our model. If the Z-Score is greater than a threshold, the day is flagged as anomalous (i.e., as fraud).

3. Application to Dataset

3.1. Dataset Description

The dataset used for this study was provided by NetGuardians SA and contains information about credit card transactions for 24 customers of a financial institution; it covers the period from June 2017 to February 2019. For reasons of confidentiality, the name of the financial institution is not mentioned. Each row is related to a customer ID and represents a transaction with its various features (i.e., timestamp, amount etc.) including the class label (1 for fraud and 0 for legitimate transaction). An important aspect is that each of the 24 customers presents at least 1 fraud in the whole period. Figure 2 and Table 1 show the number of daily transactions for all customers and the frequency of fraud and legitimate transactions in the whole dataset. We remark that the dataset is highly imbalanced with a proportion of fraud of 0.76%.

Table 1. Frequency of fraud and legitimate transactions in the whole dataset.

	Legitimate	Fraud	Total
Number	11,384	87	11,471
Percentage	99.24%	0.76%	100%

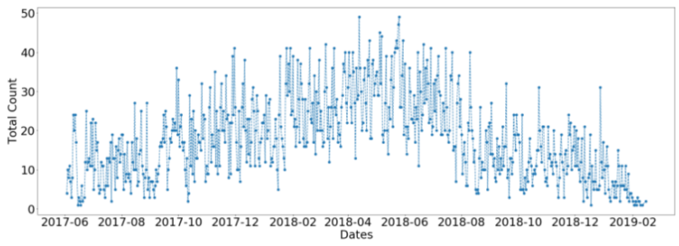

Figure 2. Number of daily transactions summing up all customers.

However, it is important to note that the customers are not necessarily active during the whole period. In fact, as illustrated in Figure 3, some of them perform transactions only in the first part of the considered time frame, others only at the end, and others in the middle. Our approach based on the ARIMA model requires sufficient legitimate transactions in the training set in order to learn the legitimate behaviour of the customers. In addition, our approach requires at least one fraud in the testing set to evaluate the performance of the model. In this context, initially, we propose to split the dataset into the training and testing set with a 70–30 ratio. With this setting, there is at least one fraud in the testing set and no fraudulent transactions in the training set, but, unfortunately, this reduces the number of customers' time series from 24 to 9. Table 2 summarises the composition of the final 9 time series that are used in the next section. The last column indicates the number of frauds over the total number of transactions occurring on the same day; as can be seen, only in one of the time series (number 10) do frauds occur on two different days.

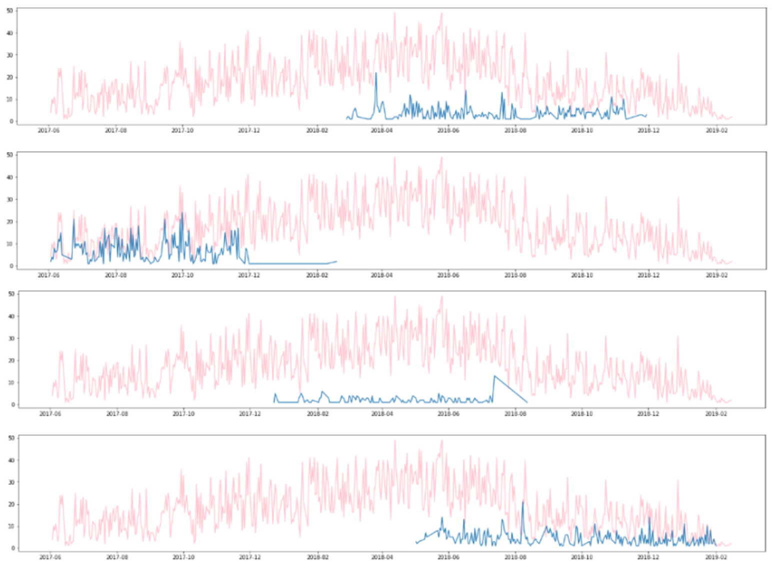
Figure 3. Number of daily transactions: the blue dot represents a specific customer and the red dot represents all customers.

Table 2. Structure of the 9 time series.

Time Series ID	# Days in Train	# Days in Test	Fraud Proportion
0	192	83	1/14
4	193	84	1/3
5	192	83	1/16
7	186	80	1/11
8	131	57	3/15
9	164	71	8/21
10	193	84	4/17
15	191	82	1/11 and 1/2
17	119	51	2/12

3.2. Application of ARIMA Model for Daily Counts of Transactions

The previously outlined steps are performed for each of the 9 time series separately. These are now described in detail for just one of the time series for the sake of clarity and brevity as an illustration. As previously discussed, the first step involves establishing whether the time series is stationary. To do this, we perform the ADF test, whose results are shown in the Table 3. It can be observed that the time series is stationary with significant results.

Next, Figure 4a,b show the PACF and ACF that are used to determine the best values for the order p and q of the ARIMA model. For this time series, there may be a drop-off in the PACF at lag 1 and in the ACF at either lag 1 or 2, suggesting an ARIMA(1,0,1) or ARIMA(1,0,2). The steps for parameter estimation and residual analysis in the training set are carried out to select one of the two models; the model ARIMA(1,0,2) is determined to be a good model for this time series and is used to make forecasts. Figure 4c shows the correlogram of the residuals for the selected model, and this confirms that they have a white noise pattern. These above steps are performed for the remaining eight time series; in some cases, the configuration may require multiple attempts to identify the best parameters. All parameters passed on to the next stage of the study are found to be significant. It is important to mention that for the forecasting in the testing set, we set the threshold to three. So, when the Z-Score is greater than three, there is fraud.

Table 3. Statistics of ADF on the stationarity for one time series.

t-Statistic	−8.73162539099
p-value	$3.180176629 \times 10^{-14}$

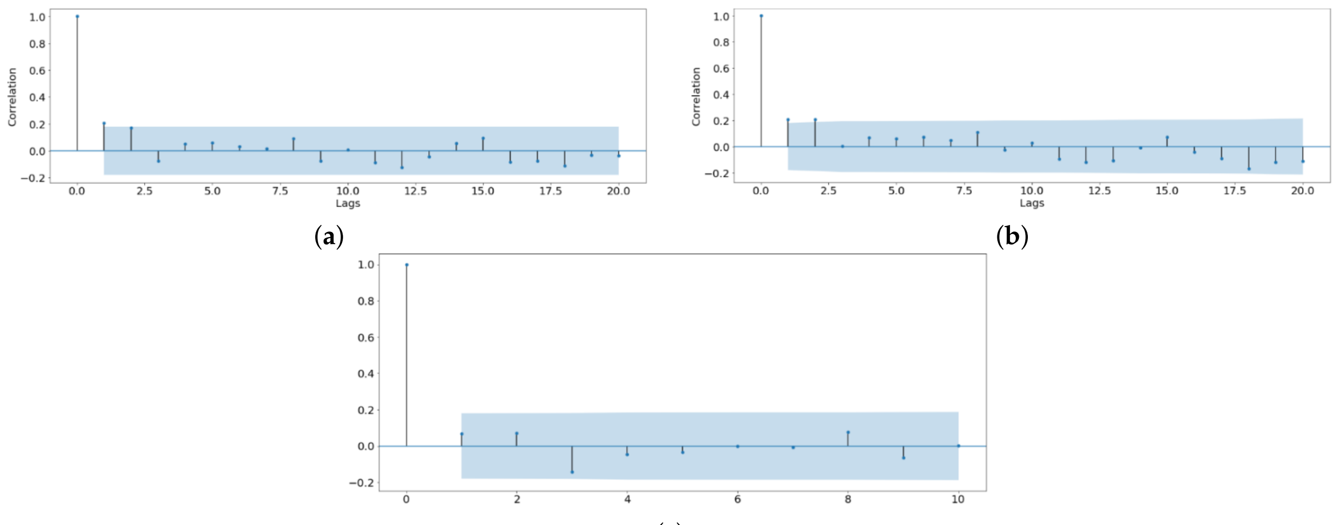

Figure 4. (**a**) Partial autocorrelation plot for sample time series; (**b**) autocorrelation plot for sample time series; (**c**) correlogram of residuals.

3.3. Benchmark Models

Our model is compared to four different models of anomaly detection, namely, the box plot, local outlier factor (LOF), isolation forest and the K-means models. Each benchmark model is briefly explained in the following section.

3.3.1. Box Plot

Box plots are used in the context of exploratory data analysis; they can be used to graphically represent data using their descriptive statistics. Box plots do not make any assumptions about the statistical distribution followed by the sample, meaning that potential outliers are identifies solely based on the degree of dispersion of the data points in the sample. Box plots are very useful, because they can be used to effectively identify patterns in groups of numbers that might be invisible to the human eye [15]. Being a visual tool, box plots are often used to increase our understanding of data, thereby allowing for a better interpretation of quantitative data [15]. We apply a box plot to the entire dataset (for each time series); however, only the testing portion of the dataset is considered to calculate the results. This is carried out for consistency reasons in order to ensure a fair comparison of the performances.

3.3.2. Local Outlier Factor (LOF)

The local outlier factor (LOF) is an algorithm that was introduced by Breunig, Kriegel, T. Ng and Sander in 2000 with the aim of identifying anomalous data points based on their local deviation from their neighbours. LOF is a density-based algorithm, and it is centred on the concept of degree of being an outlier [16], as opposed to a binary classification of outliers. The model is local because the anomaly score assigned to each point derives from the degree of isolation of that point compared to the its k neighbours, where k can be specified. More precisely, the locality of a point is given by its k-nearest neighbours. A point is considered to be an outlier when its local density is significantly lower than the densities of its neighbours [17]. For more details about LOF, see [16]. As in the case of the box plot, LOF is applied to the entire dataset only considering the testing set to calculate the results. As previously explained, this is carried out in order to retain consistency across the tests.

3.3.3. Isolation Forest

Isolation forest is an anomaly detection algorithm that implements a new approach compared to other models used for this purpose: rather than focussing on identifying normal points and their deviations (i.e., anomalies), isolation forest directly focuses on the detection of these anomalies without profiling. This can be achieved on the basis of two fundamental properties of outliers; that is, they are few in number and different, which means that they are isolated from the other—regular—points [18]. In order to isolate anomalies, this algorithm makes use of a tree structure, which results in outliers being placed closer to the root of the tree compared to the other points [19]. In isolation forest, each isolation tree isolates the anomalies by randomly selecting features and a split value between the minimum and the maximum values of that feature; the random partitioning should result in anomalies having a shorter path due to both the low number of such instances and to their inherently different characteristics leading to early partitioning. More details about the algorithm of isolation forest can be seen in [19]. Isolation forest does not require labels to work; however, it is trained on the training set comprising only of legitimate transactions and used to classify the data points in the testing set.

3.3.4. K-Means

K-means is an unsupervised learning model used for *clustering*. Clustering is the process by which from a given input, clusters or groupings are identified [20]. The process by which K-means operates can be divided into two parts: given an input comprising of a set of instances x_1, x_2, x_3, ..., x_n, and a number of clusters K, the algorithm places the centroids c_1, c_2, c_3, ..., c_n for each cluster J at random locations, and then the steps presented below are followed:

1. For each point x_n:
    (a) Find the nearest centroid c_j. K-means computes the Euclidean distance between each point x_n and centroid c_j. This approach is often called minimising the inertia of the clusters [21] and can be defined as follows:

    $$SS_{w_i} = \sum_n ||x_n - c_j||^2 \forall i \in (1, K)$$

    where n is the number of points x and i is the number of centroids c.
    (b) Assign instance x_n to cluster J.
2. For each cluster $J : 1, 2, ...K$
    (a) Compute the new centroid c_j. This is achieved by calculating the mean from each point x to the centroid x of the cluster J to which is was firstly assigned.
3. Stop when convergence is reached; that is, there are no more changes after the iterations.

For more details on K-means, see also [21,22]. We fit K-means to the entire dataset specifying two clusters (for legitimate and fraudulent daily counts). The cluster containing the smallest number of instances is considered to be the cluster indicating the positive class. As with the box plot and LOF, only the part of the outliers in the testing set is taken into account.

4. Results

The results are presented based on three metrics: precision, recall and F-measure. Precision refers to the ability of the model to be trustworthy in regard to its classified positive points; that is, precision tells us how many of the predicted frauds are actually frauds. High precision means that when the model classifies a point as positive, it is highly likely that it is a correct classification. This metric is defined by the following equation:

Precision = True Positive/(True Positive + False Positive). Recall indicates the ability of the model to detect the positive class. When a model presents a high recall, it means that the majority of positive data points are correctly identified. The equation for recall is as follows: *Recall = True Positive/(True Positive + False Negative)*. Precision and recall indicate two opposite properties of a model, meaning that optimising one implies worsening the other. In order to gain a more comprehensive overview of the performance of the model, we can use the F-measure metric, defined as shown in the following equation: *F-Measure = 2(Precision ∗ Recall)/(Precision + Recall)*. These metrics are calculated for each of the nine time series analysed and are used to obtain the average as described in the previous section. The results are presented in the Table 4. As can be noted, ARIMA presents the best result in terms of precision and F-measure, whereas K-means provides the best performance in terms of recall. The worst-performing model in this setting is the local outlier factor, which presents precision and F-measure scores of 8.4% and 14.04%, respectively. It should be pointed out that LOF was designed to be effective with multidimensional datasets [16], which might explain its bad performance in this particular setting. The box plot model performs the best amongst the benchmarks with a F-measure of 72.22% and is, thus, the only one that is comparable to our model. The advantage of our model that it is based on the concept of modelling the normal behaviour of the customer. In addition, the forecasting by the rolling windows takes into account the dynamic changes in the spending behaviour of the customer. While it can be argued that our model is overall the best one, it underperforms when compared to the box plot, isolation forest and K-means in terms of recall. As previously discussed, only 9 out of the 24 possible time series are retained for analysis due to the lack of frauds in the testing set. Consequently, the results that were presented are highly dependent of that particular set of data. In order to assess the robustness of the model, the time series that were originally discarded are reintegrated through the injection of one fake fraudulent transaction in the testing set. The occurrence of frauds is simulated by the addition of a varying number of counts ranging from 1 to 8 to a random date in the testing set for each time series. The range is set from 1 to 8 as it reflects that observed in the 9 time series previously discussed. It should be noted that the performance of the models highly varies depending on how many counts are added and on which day. In order to account for this randomness, this process is repeated 100 times, and the average of the metrics is computed. In order to gain an overview of the performances over the 24 time series, a global average is computed, which is shown in Table 5.

Table 4. Comparison of the performances of the 5 models using the 9 time series.

METRICS	ARIMA	BOX-PLOT	LOF	IF	K-MEANS
Precision	50%	43.98%	8.4%	25.01%	21.82%
Recall	66.67%	72.22%	66.67%,	72.22%	83.33%
F-Measure	55.56%	52.22%,	14.04%	32.56%	28.95%

Table 5. Global performance of the 5 models using the 24 times series.

METRICS	ARIMA	BOX-PLOT	LOF	IF	K-MEANS
Precision	34.29%	28.96%	6.41%	19.94%	22.51%
Recall	42.03%	60.54%	69.57%,	64.09%	68.16%
F-Measure	36.19%	34.91%,	11.17%	24.82%	26.81%

Despite the fact that all of models under-perform after the injection of fake frauds, the ARIMA presents the best performance in terms of precision and F-measure, whereas the best recall score is achieved by the local outlier factor. The precision of the latter is, however, the worst, which means that in this case too, only the box plot is comparable to ARIMA.

5. Conclusions

This paper addresses the problem of the unsupervised approach of credit card fraud detection using the ARIMA model. The main reason for focussing on time series model is the lack of fraud data due to confidential issues, which could represent a substantial obstacle in the development of machine learning algorithms. In this context, the goal of our approach is to model the regular spending behaviour of the customer, allowing any discrepancies and deviations from it to be deemed potential anomalies. The intuition behind this approach is centred on the assumption that the occurrence of frauds on a given day would cause the daily number of transactions to be altered in such a way that could be detected as suspicious. In the training set, the ARIMA model is first calibrated on the daily number of legitimate transactions in order to learn the regular spending behaviour of the customer. In the second step, the fitted model is used to predict fraud in the testing set by using the rolling windows. The criterion of flagging fraud is based on the Z-score calculated on the prediction errors in the testing set. Our methodology is applied to the dataset that is provided by NetGuardians and is compared with four anomaly detection algorithms, namely, the K-means, box plot, local outlier factor and isolation forest algorithms. It is observed in terms of prediction power that the ARIMA model outperforms the other models following by the box plot method. Among the four benchmark models, the local outlier factor performs the worst. Our model is successful when compared to the benchmark models for two reasons:

1. It works better when there is a significant number of frauds occurring on the same day. This is often the case, as fraudsters are known to take advantage of the time they have before the card is blocked to make several fraudulent transactions in a short time span [13].
2. It presents the best precision; i.e., it reduces the number of false positives compared to the benchmark models.
3. It takes into account the dynamic spending behaviour of the customer by using the rolling windows.

One main problem in our approach is that the ARIMA model assumes that the data come from observations that are equally spaced in time. However, this assumption does not hold in our study since the transaction times are unequally spaced. This issue will be addressed in future research by using advanced approaches, such as the continuous-time autoregressive moving average (CARMA) processes.

Funding: This research was funded by InnoSuisse grant number 34137.1.

References

1. Bank, E.C. *Fifth Report on Card Fraud*; European Central Bank: Frankfurt am Main, Germany, 2019
2. Nilson. The Nilson Report | News and Statistics for Card and Mobile Payment Executives. Available online: Nilsonreport.com (accessed on 1 June 2019)
3. Maniraj, S.P. Credit Card Fraud Detection using Machine Learning and Data Science. *Int. J. Eng. Res. Technol.* **2019**, *8*, 110–115 [CrossRef]
4. Tripathi, D.; Sharma, Y.; Tushar, L.; Shubhra, D. Credit Dard Fraud Detection Using Local Outlier Factor. *Int. J. Pure Appl. Math.* **2018**, *118*, 229–234.
5. Pozzolo, A.D. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Syst. Appl.* **2014**, *41*, 4915–4928. [CrossRef]
6. Singh, D.; Vardhan, S.; Agrawal, N. Credit Card Fraud Detection Analysis. *Int. Res. J. Eng. Technol. (IRJET)* **2018**, *5*, 1600–1603.
7. Khare, N.; Sait, S.Y. Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models. *Int. J. Pure Appl. Math.* **2018**, *118–120*, 825–838.
8. Varmedja, D. Credit Card Fraud Detection—Machine Learning methods. In Proceedings of the 18th International Symposium INFOTEH-JAHORINA, Jahorina, Bosnia and Herzegovina, 20–22 March 2019
9. Roy, A. Deep learning detecting fraud in credit card transactions. In Proceedings of the 2018 Systems and Information Engineering Design Symposium, Charlotteville, VA, USA, 27 April 2018; pp. 129–134.
10. Adhikari, R.; Agrawal, R.K. An Introductory Study on Time Series Modeling and Forecasting. *arXiv* **2013**, arXiv:1302.6613.

11. Box, G.E.P.; Jenkins, G.M.; Reinsel, G.C. *Time Series Analysis: Forecasting and Control*, 4th ed.; John Wiley & Sons: Hoboken, NJ, USA, 2008
12. Azrak, R.; Melard, G. *Exact Maximum Likelihood Estimation for Extended ARIMA Models*; Université Libre de Bruxelles Institutional Repository: Brussels, Belgium, 2013.
13. Seyedhossein, L.; Hashemi, M.R. Mining information from credit card time series for timelier fraud detection. *Int. J. Inf. Commun. Technol.* **2010**, *2*, 619–624.
14. Ounacer, S. Using Isolation Forest in anomaly detection: The case of credit card transactions. *Period. Eng. Nat. Sci. (PEN)* **2018**, *6*, 394. [CrossRef]
15. Williamson, D.F. The Box Plot: A Simple Visual Method to Interpret Data. *Ann. Intern. Med.* **1989**, *110*, 916–921. [CrossRef] [PubMed]
16. Breunig, M.M.; Kriegel, H.-P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. *ACM SIGMOD Rec.* **2000**, *29*, 93–104. [CrossRef]
17. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn.* **2011**, *12*, 2825–2830
18. John, H.; Naaz, S. Credit Card Fraud Detection using Local Outlier Factor and Isolation Forest. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 1060–1064. [CrossRef]
19. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 8th IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; Volume 7, pp. 413–422.
20. Kubat, M. *An Introduction to Machine Learning*; Springer: Berlin, Germany, 2015.
21. Dalatu, P.I. Time Complexity of K-Means and K-Medians Clustering Algorithms in Outliers Detection. *Glob. J. Pure Appl. Math.* **2018**, *12*, 4405–4418. .
22. Bonaccorso, G. *Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning*; Packt: Birmingham, UK, 2018