

Article

High-Cardinality Categorical Attributes and Credit Card Fraud Detection

Emanuel Mineda Carneiro ^{1,2,*} , Carlos Henrique Quartucci Forster ² , Lineu Fernando Stege Mialaret ³,
Luiz Alberto Vieira Dias ² and Adilson Marques da Cunha ²

¹ Sao Paulo State Technological College (Faculdade de Tecnologia—Fatec),
Sao Jose dos Campos 12247-014, Brazil

² Brazilian Aeronautics Institute of Technology (Instituto Tecnológico de Aeronáutica—ITA),
Sao Jose dos Campos 12228-900, Brazil

³ Federal Institute of Education, Science and Technology of Sao Paulo (Instituto Federal de Sao Paulo—IFSP),
Jacarei 12322-030, Brazil

* Correspondence: emanuel.mineda@fatec.sp.gov.br

Abstract: Credit card transactions may contain some categorical attributes with large domains, involving up to hundreds of possible values, also known as high-cardinality attributes. The inclusion of such attributes makes analysis harder, due to results with poorer generalization and higher resource usage. A common practice is, therefore, to ignore such attributes, removing them, albeit wasting the information they provided. Contrariwise, this paper reports our findings on the positive impacts of using high-cardinality attributes on credit card fraud detection. Thus, we present a new algorithm for domain reduction that preserves the fraud-detection capabilities. Experiments applying a deep feedforward neural network on real datasets from a major Brazilian financial institution have shown that, when measured by the F-1 metric, the inclusion of such attributes does improve fraud-detection quality. As a main contribution, this proposed algorithm was able to reduce attribute cardinality, improving the training times of a model while preserving its predictive capabilities.

Keywords: credit card fraud; fraud-detection system; high-cardinality attribute; pattern recognition; clustering; deep learning

MSC: 68T07



Citation: Carneiro, E.M.; Forster, C.H.Q.; Mialaret, L.F.S.; Dias, L.A.V.; da Cunha, A.M. High-Cardinality Categorical Attributes and Credit Card Fraud Detection. *Mathematics* **2022**, *10*, 3808. <https://doi.org/10.3390/math10203808>

Academic Editors: Fan Zhang, Songhe Feng, Yongsheng Zhou and Junlin Hu

Received: 9 September 2022

Accepted: 11 October 2022

Published: 15 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Credit card frauds occur when card information is stolen and purchases are made without the permission of the cardholders [1]. They are responsible for global yearly losses of over billions of dollars and, since an all-time low in 2010, these losses have been increasing yearly as a percentage of the total volume of transactions and are projected to reach a total of \$43.78 billion by 2025 [2]. This phenomenon is so widespread that 30% of cardholders have reportedly experienced some kind of card fraud from 2012 to 2016 [3].

Regarding transaction entry mode, more than half of the card frauds in the United States are generated by transactions that do not make use of physical cards. This kind of fraud is known as Card-Not-Present (CNP) and means to prevent, combat, and limit them are in high demand [2].

In order to reduce their losses, credit card issuers make use of fraud-detection systems. These systems generally implement and support both an automatic and a manual process. The former is responsible for assigning a fraud likelihood score to incoming transactions, while the latter allows fraud investigators to provide binary feedback (fraudulent or legitimate) on transactions with a high fraudulence score [1].

Fraudsters constantly change their strategies in order to avoid detection. As a result, traditional rule-based fraud-detection systems turn obsolete very fast [4]. This motivates

research on the applicability of Machine Learning (ML) techniques to this problem. Themes like comparison between classification techniques [5–7], evaluation metrics [8], feature selection [9], feature engineering [10,11], class imbalance [8,11–13], dimensionality reduction [14], ensemble learning [4,9,15,16], concept drift [13], one-class classification [17], sequence classification [1], and so on have already been previously discussed by other authors.

Credit card transactions contain some attributes with large domains, up to hundreds of values, such as the Merchant Category Code (MCC) or ones related to customer or merchant addresses. These attributes are often absent from public datasets, for obvious privacy reasons. For that reason, finding real and complete datasets for research purposes is difficult [4]. While researching a credit card fraud-detection solution for a Brazilian corporation, we had the opportunity to work with such data and were challenged by dealing with this kind of attribute.

This article reports our findings on the impact of using high-cardinality attributes on credit card fraud detection. We also present an algorithm, named Value Clustering for Categorical Attributes (VCCA), that aims to reduce the cardinality of their domains while preserving fraud-detection capabilities.

The remainder of this article is organized as follows: Section 2 summarizes a fraud-detection system of a major Brazilian financial institution; Section 3 presents a brief review on techniques for dealing with high-cardinality attributes; Section 4 synthesizes the methodology and the main technologies used in this research; Section 5 presents and discusses experimental results; and finally, Section 6 summarizes our main conclusions and suggestions for future work.

2. Fraud Detection System of a Major Brazilian Financial Institution

In order to pursue this research, we have been given access to data and business knowledge regarding the fraud-detection system of a major Brazilian financial institution, as detailed in this section.

The fraud-detection system architecture was comprised of an acceptance check and an offline sanity check. The acceptance check was responsible for checking the integrity of the transaction, which was rejected in some situations because, for instance, the informed Personal Identification Number (PIN) was wrong, the account was blocked, or the amount exceeded the credit limit. Once a transaction is accepted, a fraudulence score is calculated. Contrary to the typical fraud-detection system architecture presented by [10], there was no online sanity check which was responsible for automatically rejecting a transaction. The fraudulence score was calculated and processed by the offline sanity check, implemented as a rule-based system, maintained manually at regular intervals. A transaction with a score greater than a given threshold would generate an alert and be sent to a fraud investigator, who could either dismiss the alert or contact the credit card holder to verify if the transaction was legitimate.

Once a transaction was found to be fraudulent, it was reversed and a chargeback occurred. Credit card holders were also able to dispute transactions for 180 days. In such a case, an investigation was performed, and the merchant should provide evidence that the transaction was performed by the credit card holder. If a merchant failed to provide enough evidence, the transaction was considered fraudulent, reversed, and the payer refunded. As a result of this process, it takes a long time for a financial institution to identify credit card frauds, with around 80% of them being identified in a 2-month window [5].

Alerts were handled manually by fraud investigators, making a high number of false alerts (also known as false positives) undesirable, as personnel would be overburdened and unable to verify all transactions. The financial institution also needed to detect as many frauds as possible, to minimize losses. Regarding these requirements, their fraud-detection system was unreliable, as real alerts, associated with fraudulent transactions, comprised only roughly 10% of all alerts generated. This low accuracy was one of the main factors for not using an online sanity check, as automatically denying legitimate transactions would negatively impact customer satisfaction.

3. High-Cardinality Categorical Attributes

Categorical attributes with large domains containing more than 100 distinct values, like some of the ones found on credit card transactions, are named High-Cardinality (HC) attributes [18]. Dealing with these attributes by creating dummy variables, the most common approach, causes dimensions of the attribute space to quickly explode due to the Cartesian products. For instance, a dataset with an attribute comprised of hundreds of values would result in a new dataset with hundreds of attributes, one binary attribute for each value. The inclusion of such attributes typically hinders generalization, as they could result, for example, in the creation of “lookup tables” [19].

In order to avoid these problems, HC attributes are often dismissed, but by doing so there is evidence that classification accuracy is jeopardized [18]. With this in mind, new forms to represent HC attributes from a particular domain are desirable and their cardinality should be neither too rough to make objects indistinguishable nor too fine to become redundant [20]. Therefore, one key issue is to find a mapping of the attribute domain to a new domain with minimal cardinality, but still preserving its semantics [21].

There are several approaches for reducing the domain of categorical attributes, also named symbolic or nominal attributes. Most of these approaches treat this domain reduction as a grouping problem, partitioning the set of values into a finite number of groups [22]. Aside from the grouping approach, another explored possibility is to transform a categorical attribute into a numeric one.

3.1. Numeric Encoding

Numeric encoding consists in mapping values from a categorical attribute to a corresponding numeric, discrete or continuous, attribute. Integer encoding is the simplest approach to accomplish this and consists in attributing a random integer to each value of the categorical attribute. Variations of this approach apply different metrics, like frequency, to the categorical attribute to generate corresponding numeric values.

Target Encoding [23] greatly improves over these simple approaches by including the target attribute in the encoding process. This algorithm differs depending on the type of the target attribute. For a numeric target attribute, a value v_i of the categorical attribute is mapped to the mean value of the target attribute for all associated records (records where the categorical attribute assumes value v_i). For a categorical target attribute, each value v_i of the categorical attribute is mapped to the posterior probability of the target attribute value associated with it.

Following the same principle, Ref. [18] presented three new Target Encoding methods that transform categorical attributes into continuous attributes whose values are correlated with a target attribute. They also showed that, by using this kind of transformation, HC attributes could be used in a real prediction task, churn prediction, and were able to increase classifier accuracy. These transformation methods have low complexity and can be applied on any dataset, regardless of dimensionality or size.

Quantile Encoding is another variation of Target Encoding specifically for numeric target attributes and was presented by [24]. This approach replaced the mean function by the quantile and also added a regularization term to avoid overfitting.

The Conjugate Bayesian Model (CBM) Encoder was proposed by [25] and is based on traditional Bayesian statistics. Initially, this approach encoded categorical attributes with moments of the posterior distributions on each unique value through Bayes' theorem. Afterward, the learning algorithm improved upon this encoding by accounting for variance of posterior distributions and interactions among other features.

Entity embedding uses Artificial Neural Networks (ANNs) to map a one-hot encoded categorical attribute into a reduced set of numeric attributes [26]. This approach includes an additional layer between inputs generated by one-hot encoding and the first hidden layer of an ANN. This layer, which presents lesser cardinality, maps the binary inputs generated by one-hot encoding for a categorical variable in a function approximation problem into Euclidean spaces. This mapping is learned by the ANN during its training. There are

variations of this approach using Deep Learning [27] and mapping the inputs into a single neuron, which can be seen as a dictionary [28].

While numeric encoding approaches have several benefits, such as low computational cost and ease of implementation, they change the attribute type, which could impair interpretability. The simpler ones could also lead to information loss, which can negatively impact on the accuracy of ML algorithms.

3.2. Grouping

The grouping approach for cardinality reduction of a categorical attribute consists in partitioning its values based upon a similarity metric.

One of such grouping approaches was introduced by [29,30] as the Optimal Family of Partitions (OFP) problem, also known as the Optimal Symbolic Value Partition (OSVP) problem, which aims to combine attribute values into a minimum number of groups, while completely preserving the original decision table.

Unfortunately, the solution for this problem was proved to be NP-Hard. Henceforth, sub-optimal solutions, not guaranteed to provide the minimum number of groups, were proposed using several approaches: the Rough Set Theory combined with a greedy algorithm [21,31], the Rough Set Theory combined with Attribute Value Taxonomies (AVT) [32], the Granular Computing combined with automatically generated AVT [33], among others.

In order to work, these solutions require a dataset solely composed of categorical attributes. As credit card transactions possess both categorical and continuous attributes, a series of transformations would be necessary before applying any of these approaches.

There are also grouping approaches that preserve the classification capabilities of datasets to an extent, using some criteria to keep losses within a predetermined confidence level. Target Encoding [23] suggested a grouping approach, where numeric values generated by it could be combined in a greedy bottom-up way. Two values could be combined into a new value if doing so did not affect a given metric, gain ratio. This process continued until it was not possible to combine any value without affecting the metric.

Khiops [22] was a similar grouping approach based on a greedy bottom-up algorithm, with a stop rule that makes use of a confidence level computed with chi-square statistics.

While the previously presented grouping approaches try to guarantee preservation of the original decision table to a certain degree, they also present complexity that could prevent their usage in problems involving high dimensional or big data datasets. Some of these approaches also require domain knowledge about the attributes, like the ones based on AVTs, which can be a problem as categorical attributes in credit card datasets are often anonymized, in order to provide privacy protection.

Some grouping approaches do not use any metric to guarantee preservation of the original decision table and group values using an unsupervised process.

Following this line of research, Similarity Encoding [34] treats the high-cardinality problem as a data-cleaning problem, assuming the existence of multiple redundant values. This type of algorithm uses Natural Language Processing (NLP) techniques, such as 3-g [34], Gamma-Poisson matrix factorization on substring counts, and min-hash encoder [35], to group semantically similar values.

Using a numeric approach, Hashing [36] maps values to a new low cardinality space using hash. Depending on the desired size of this new space, multiple unrelated values can be mapped to the same hash, a problem which is called collision.

For our research, values of HC attributes would initially be manually grouped using heuristics provided by fraud investigators. However, as the research progressed, this was deemed infeasible due to the amount of values found. We were then asked to automatically group values in a way that could be verified and changed by fraud investigators afterwards. Our first attempt [37] produced promising results but did not meet our time constraints due to the complexity of the chosen clustering algorithm. This led our team to create the algorithm presented in this article, which is a top-down variation of the grouping approach proposed by [23].

4. Materials and Methods

This section presents the proposed algorithm of Value Clustering for Categorical Attributes (VCCA) and the techniques and the metrics used in the experiments.

4.1. Techniques

Deep FeedForward Networks (FFN), also known as MultiLayer Perceptron (MLP) ANN, are a class of universal approximators [38–40] that have been successfully used in multiple recent studies on credit fraud detection [41,42]. An FFN can be seen as a multilayered chain of functions, where the length of the chain defines the depth of the model. Such terminology gave birth to the Deep Learning concept [43]. In this research we decided to use a fully connected FFN as the classifier, to validate the impact of HC attributes on fraud detection.

The FFN implementation provided by Keras [44] was used in an architecture containing one input layer, two hidden layers (H1 and H2), and one output layer, as shown in Figure 1. Attributes from a credit card transaction were fed to the input layer and processed by the hidden layers, resulting in a unique output containing a fraud score, where values higher than a threshold were translated as fraudulent transactions. The Rectified Linear Activation Function (ReLU) [45], presented in Equation (1), was used as an activation function for the hidden layers. The sigmoid function, presented in Equation (2), was used on the output neuron. Binary Cross-Entropy was used as a cost/loss function. To lower the complexity of the model during training and reduce overfitting, we used L2 regularization [46], which adds a penalty term, proportional to a new hyperparameter α , to the loss function. The loss function with L2 regularization is presented in Equation (3), where N is the number of transactions in the training dataset, y is the predicted value, p is the probability of a transaction being fraudulent, M is the number of weights in the FFN, and w is the value of a weight.

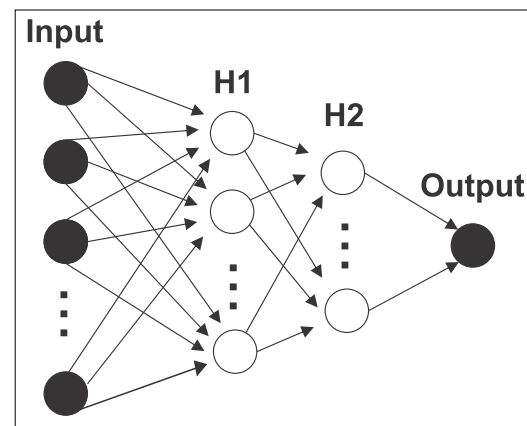


Figure 1. Fully connected feedforward MLP with two hidden layers.

$$f(x) = \max(0, x) \quad (1)$$

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

$$loss = \frac{1}{N} \sum_{i=1}^N -(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)) + \alpha * \sum_{i=1}^M w_i^2 \quad (3)$$

An FFN works only with numerical attributes, so it was necessary to binarize categorical attributes, converting them into dummy variables. The One-Hot Encoding algorithm creates a new binary attribute for each existing value in the original categorical attribute. The original attribute is excluded at the end of the process. An implementation from Scikit-Learn [47] was used for this transformation.

Our proposed algorithm, VCCA, makes use of a discretizer to reduce the cardinality of categorical attributes. The Nested Means algorithm [48] was chosen for its simplicity and performance. It is an algorithm based upon ordinary univariate statistical procedures for splitting data and iteratively using the arithmetic mean to divide a class into two new classes. At the end of the process, it generates up to 2^i classes, where i correspond to the number of iterations.

For numeric attributes, we have taken the Napierian logarithm of the ones with a large range of values, where more than 32% of the records have a value where the distance from the mean is greater than the standard deviation. All numeric attributes were also standardized, using the Scikit-Learn implementation of the StandardScaler algorithm.

As credit card datasets are highly skewed, the model was provided with the classes weights, proportional to the occurrence of each class (fraudulent and legitimate transactions). Each weight is calculated following Equation (4), where $|c|$ is the number of records associated with the class and N is the total number of records in the dataset. Accordingly, bias on the output neuron was initialized, as the Napierian logarithm of the quotient between the number of fraudulent transactions, $|t|$, and the number of legitimate transactions, $|f|$, as shown in Equation (5).

$$weight(c) = \frac{N}{2 * |c|} \quad (4)$$

$$initial_bias = \log \frac{|t|}{|f|} \quad (5)$$

To compare results from ML models, to have multiple experiment results obtained from different datasets is necessary. To comply with this requirement, an implementation from Scikit-Learn of the stratified n-fold cross-validation was used to generate 10 different datasets.

4.2. The Proposed Algorithm of Value Clustering for Categorical Attributes

The inclusion of HC attributes often results in longer classifier training times and higher resource consumption. In order to solve this issue while avoiding downsides of available approaches, presented in Section 3, we devised a low-complexity top-down grouping algorithm, named VCCA.

At the base of the algorithm is the probability of fraudulence, $P(t|v)$, of a transaction containing an attribute A with a given value v . In this scenario, there is a fraud flag attribute which assumes a true value t when the transaction is fraudulent. $P(t|v)$ is presented in Equation (6), where $|tv|$ represents the number of fraudulent transactions with a value v for attribute A and $|v|$ represents the number of transactions with value v for attribute A .

$$P(t|v) = \frac{|tv|}{|v|} \quad (6)$$

We used Information Gain (IG) as a metric to select categorical attributes to use in our fraud-detection system. To preserve this metric, we analyzed the IG equation and found out that grouping values of an attribute with close enough $P(t|v)$ preserves this metric.

Theorem 1. *Grouping values of an attribute with close enough $P(t|v)$ preserves Information Gain.*

Proof. For an attribute A , with categorical values $\{v1, \dots, vx, vy, \dots, vn\}$, in a set of training examples T containing a binary target attribute Y , the Information Gain $IG(T, A)$ can be calculated as:

$$\begin{aligned}
IG(T, A) &= H(T) \\
&- \frac{|v1|}{|T|} P(t|v1) \log_2(P(t|v1)) - \frac{|v1|}{|T|} (1 - P(t|v1)) \log_2(1 - P(t|v1)) \\
&- \dots \\
&- \frac{|vx|}{|T|} P(t|vx) \log_2(P(t|vx)) - \frac{|vx|}{|T|} (1 - P(t|vx)) \log_2(1 - P(t|vx)) \\
&- \frac{|vy|}{|T|} P(t|vy) \log_2(P(t|vy)) - \frac{|vy|}{|T|} (1 - P(t|vy)) \log_2(1 - P(t|vy)) \\
&- \dots \\
&- \frac{|vn|}{|T|} P(t|vn) \log_2(P(t|vn)) - \frac{|vn|}{|T|} (1 - P(t|vn)) \log_2(1 - P(t|vn))
\end{aligned} \quad (7)$$

Looking at Equation (7), grouping two different values vx and vy under a new label vxy would result in a new probability $P(t|vxy)$ and an error e_{xy} , proportional to the difference between the original probabilities $P(t|vx)$ and $P(t|vy)$, in a way that:

$$\begin{aligned}
&- \frac{|vx|}{|T|} P(t|vx) \log_2(P(t|vx)) - \frac{|vx|}{|T|} (1 - P(t|vx)) \log_2(1 - P(t|vx)) \\
&- \frac{|vy|}{|T|} P(t|vy) \log_2(P(t|vy)) - \frac{|vy|}{|T|} (1 - P(t|vy)) \log_2(1 - P(t|vy)) = \\
&- \frac{|vx| + |vy|}{|T|} P(t|vxy) \log_2(P(t|vxy)) \\
&- \frac{|vx| + |vy|}{|T|} (1 - P(t|vxy)) \log_2(1 - P(t|vxy)) + e_{xy}
\end{aligned} \quad (8)$$

When, in Equation (8), $P(t|vx)$ approaches $P(t|vy)$, we can write it in terms of a constant k and vy , in a way that $|tvx| = k|tvy|$ and $|vx| = k|vy|$:

$$\lim_{P(t|vx) \rightarrow P(t|vy)} P(t|vx) = \frac{|tvx|}{|vx|} = \frac{k|tvy|}{k|vy|} \quad (9)$$

Considering Equation (9), when $P(t|vx)$ approaches $P(t|vy)$, $P(t|vxy) = P(t|vy)$:

$$\lim_{P(t|vx) \rightarrow P(t|vy)} P(t|vxy) = \frac{|tvx| + |tvy|}{|vx| + |vy|} = \frac{k|tvy| + |tvy|}{k|vy| + |vy|} = P(t|vy) \quad (10)$$

Finally, we can rewrite Equation (8), when $P(t|vx)$ approaches $P(t|vy)$ as:

$$\begin{aligned}
&\lim_{P(t|vx) \rightarrow P(t|vy)} - \frac{|vx|}{|T|} P(t|vy) \log_2(P(t|vy)) \\
&- \frac{|vx|}{|T|} (1 - P(t|vy)) \log_2(1 - P(t|vy)) \\
&- \frac{|vy|}{|T|} P(t|vy) \log_2(P(t|vy)) \\
&- \frac{|vy|}{|T|} (1 - P(t|vy)) \log_2(1 - P(t|vy)) = \\
&- \frac{|vx| + |vy|}{|T|} P(t|vy) \log_2(P(t|vy)) \\
&- \frac{|vx| + |vy|}{|T|} (1 - P(t|vy)) \log_2(1 - P(t|vy)) + e_{xy}
\end{aligned} \quad (11)$$

By simplifying Equation (11), we obtain Equation (12):

$$\lim_{P(t|vx) \rightarrow P(t|vy)} e_{xy} = 0 \quad (12)$$

□

The VCCA is a supervised algorithm designed for decision problems with binary target attributes, consisting of two classes with values limited to positive and negative examples. It consists of two steps: transformation and supervised discretization.

In the first step, it considers an HC attribute A , with a domain containing n values. In this first step, $P(t|v)$ is calculated for each value v of A . The final result of this step is a conversion table associating each categorical value to a corresponding numerical ratio. This conversion table is ordered by $P(t|v)$ value, in order to speed up the next step.

The second step uses a supervised discretization algorithm to group the $P(t|v)$ values calculated in the previous step. Grouping quality is decided by comparing a chosen metric applied to both original attribute values and the values grouped through discretization. As previously stated, we have decided on IG as the grouping quality metric and the Nested Means algorithm as the discretizer.

Beginning with two groups, Nested Means is applied iteratively, in a top-down approach, generating an increasing number of groups until the IG calculated for the grouped values surpasses a similarity value S , defined as the percentage of the IG calculated for the original values. Due to the way Nested Means divides the dataset, it is possible that an odd number of groups is generated. Algorithm 1 details both steps.

Algorithm 1: The VCCA algorithm

Input: Categorical attribute A

Output: Conversion table

1 **begin** Transformation

 1.1 Compute an ordered list P of Ratios for values of attribute A ;

end

2 **begin** Supervised Discretization

 2.1 Compute IG for original values;

 2.2 Iteration = 1;

 2.3 GroupedIG = 0;

 2.4 **while** $S > \text{GroupedIG} / \text{IG}$ **do**

 2.4.1 Use Nested Means to divide P into $2^{\text{Iteration}}$ groups;

 2.4.2 Compute GroupedIG as the IG for the grouped values;

 2.4.3 Iteration = Iteration + 1;

end

end

Figure 2 presents an example of the VCCA application on the *habitat* attribute, taken from the Mushroom dataset of the UCI Machine Learning Repository [49], considering a similarity S of 95%.

The end result of the algorithm is a conversion table that associates values with groups. New grouped attributes with lesser cardinality can then be created by replacing, for each record, values of HC attributes for their associated groups. It is recommended to use only a portion of the dataset to generate the conversion table, to avoid overfitting. When creating a grouped attribute, values not contained in the conversion table can appear and should be translated into a default group, instead of being left untranslated.

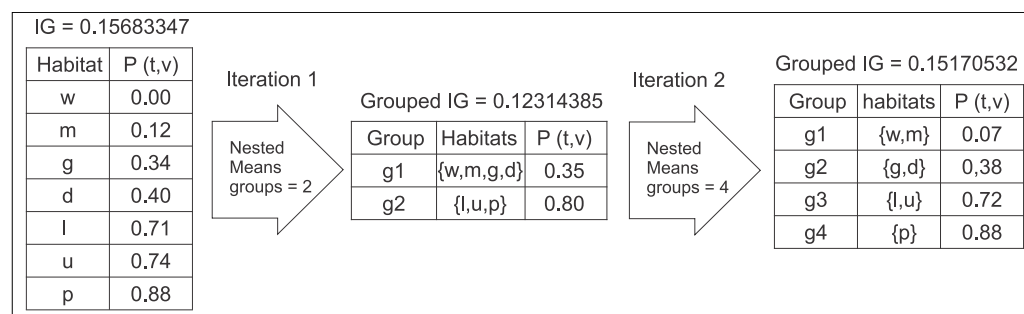


Figure 2. VCCA example.

Regarding time complexity, we consider a training dataset T , composed of N records. Computing a list P comprised of $P(t|v)$, the number of positive records and the number of negative records associated with each of the $|v|$ values of an attribute A (1.1) could be accomplished in $O(N)$, considering the use of a hashmap to store P . By using P , IG calculation (2.1 and 2.4.2) for all values takes $O(V)$, as calculating IG for each single value can be achieved in constant time.

Sorting P (2.2) by $P(t|v)$ can be achieved in $O(V \log_2 V)$ by using a sort algorithm like merge sort.

In the worst case scenario, where the grouped attribute A' has the same number of values as the original attribute A , there would occur $\log_2 V$ iterations in the main loop (2.5), making its cost $O(V \log_2 V)$.

Taking all these factors into consideration, the total complexity for the algorithm is expected to be $O(N \log_2 N)$, when $|v| \approx N$, or $O(N + |v| \log_2 |v|)$, when $|v| \ll N$. As only attributes where $|v| \ll N$ are used for training, to avoid overfitting, the latter is expected. Table 1 presents the expected complexity for all algorithm steps.

Table 1. The expected complexity by step.

Step	Expected Complexity
1	$O(N)$
1.1	$O(N)$
2	$O(V \log_2 V)$
2.1	$O(V)$
2.2	$O(V \log_2 V)$
2.3	$O(1)$
2.3	$O(1)$
2.5	$O(V \log_2 V)$
2.5.1	$O(V)$
2.5.2	$O(V)$
2.5.3	$O(1)$

4.3. Metrics

As seen in Section 2, two requirements should be met for a fraud-detection system: identify most of the fraudulent transactions and, at the same time, generate as few false alerts as possible. The first requirement could be measured through the *Recall*, also known as True Positive Rate (TPR) metric, presented in Equation (13), while the second one could be measured by the *Precision* metric, presented in Equation (14). *TP* stands for True Positives, the number of alerts generated for fraudulent transactions; *FN* stands for False Negatives, the number of fraudulent transactions that did not generate alerts; and *FP* stands for False Positives, the number of alerts generated for legitimate transactions.

$$Recall = \frac{TP}{TP + FN} \quad (13)$$

$$Precision = \frac{TP}{TP + FP} \quad (14)$$

As both metrics should achieve high values, in order to fulfill the requirements for a fraud-detection system, it was necessary to introduce a third metric, *F-1*, which is the harmonic mean of *Recall* and *Precision*. This metric is presented in Equation (15).

$$F-1 = \frac{2 * Recall * Precision}{Recall + Precision} \quad (15)$$

Receiver Operating Characteristic (ROC) curves are also largely used to evaluate the results of fraud-detection systems. As the output of such systems often has a probability of fraud, results are binarized according to a predefined threshold (usually 50%). An ROC curve is plotted using Recall for the y axis and False Positive Rate (FPR), presented in Equation (16), for the x axis. Each point is generated by calculating Recall and FPR for different values of the threshold. In this context, the Area Under the ROC Curve (AUC) is a metric that summarizes the ROC curve and measures the ability of a fraud-detection system to distinguish between fraudulent and legitimate transactions.

$$FPR = \frac{FP}{TN + FP} \quad (16)$$

For highly skewed datasets, like the ones associated with credit card fraud, it was verified that Precision–Recall curves provide a better representation of a classifier performance [50]. This curve uses Precision instead of FPR for the x axis. The associated performance metric is called Area Under the Precision-Recall Curve (PRC).

5. Results and Discussion

This Section describes the experimental scenarios and the datasets used to evaluate the influence of HC attributes on credit card detection. It also presents the discussion and evaluation concerning the results for all three experiments.

5.1. Experiments

For this research, we were given access to two datasets, containing real anonymized credit card data from a major Brazilian financial institution, henceforth named Dataset 1 and Dataset 2. These datasets were from different private labels, store-branded credit cards managed by a financial institution, containing credit card transactions from December 2013 to April 2014. Additionally, we have also used a public simulated credit card transaction dataset [51], henceforth named Dataset 3, to enable reproducibility.

Both Datasets 1 and 2 had the same structure, where each record contained information about the credit card holder, the current transaction, the last transaction, transaction statistics per credit card (e.g., speed, average value), and a flag indicating if the current transaction was legitimate or fraudulent. Several HC attributes were present, such as Zone Information Postal (ZIP) code for both credit card holder and merchant, MCC, and so on.

Dataset 1 was composed of 11829 real credit card transactions. This dataset was highly imbalanced, as only 40 out of the 11829 transactions, roughly 0.34%, were fraudulent.

Regarding entry mode, 75% of all transactions (fraudulent and legitimate) used magnetic stripe reading, while the remaining were CNP. However, only for the fraudulent transactions, this proportion was reversed, with 70% of them being CNP, which means that the majority of frauds did not occur with a physical credit card. Figure 3 presents fraud distribution per month on Dataset 1.

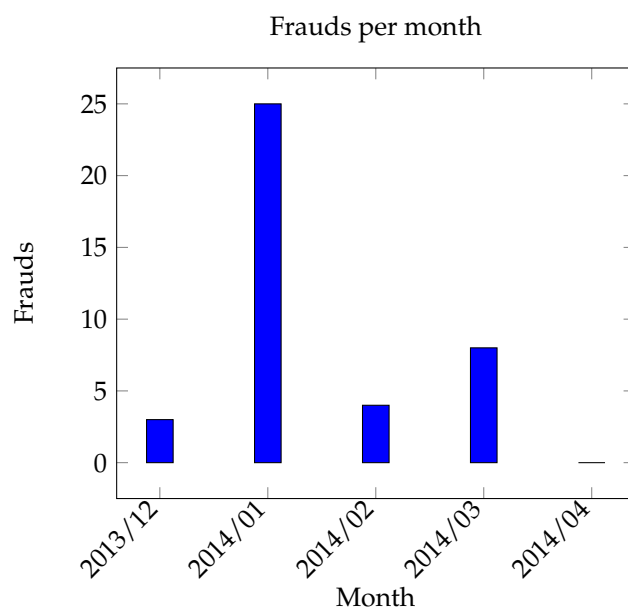


Figure 3. Frauds per month on Dataset 1.

Dataset 2 was composed of 27569 real credit card transactions. This dataset was even more imbalanced than Dataset 1, with only 51 fraudulent transactions out of the 27569, roughly 0.18%.

Regarding entry mode, 80% of all transactions (fraudulent and legitimate) from this dataset used a chip card, 2% used magnetic stripe reading, and the remaining were CNP. Nonetheless, all fraudulent transactions were CNP, which means that all frauds occurred without a physical credit card. Figure 4 presents fraud distribution per month on Dataset 2.

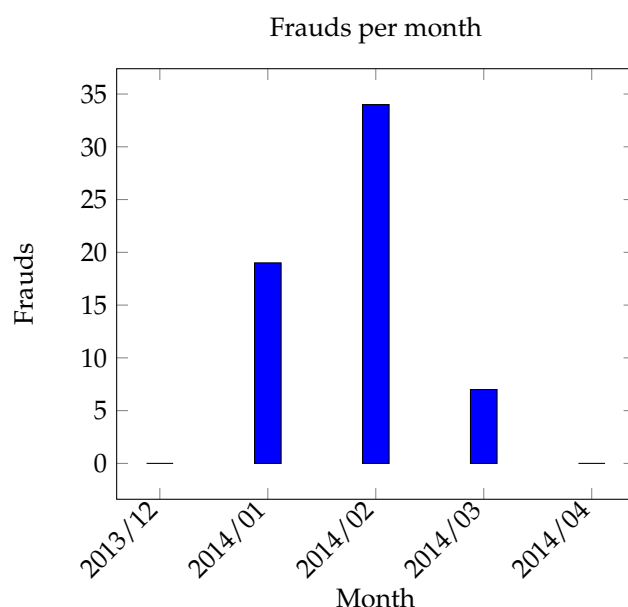


Figure 4. Frauds per month in Dataset 2.

Dataset 3 was a rare find as it is public and provided both categorical and numerical attributes. Because of hardware constraints, only the smaller dataset, available on file fraudTest.csv, was used in the experiments. Transactions were dated from June 2020 to December 2020 for a total of 555,719 records, from which only 2145, 0.9%, were fraudulent. Figure 5 presents fraud distribution per month. Table 2 presents all attributes used in the experiments.

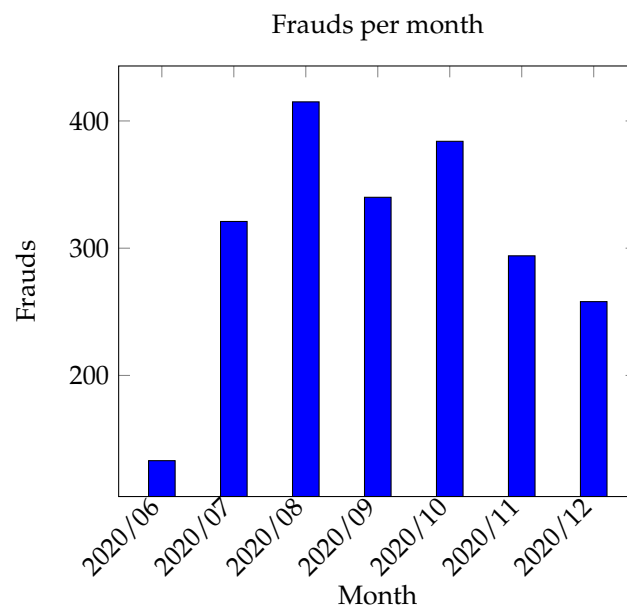


Figure 5. Frauds per month in Dataset 3.

Table 2. Attributes.

Name	Description	Type	Origin
gender	Card holder gender	Categorical	Dataset
age	Card holder age	Numerical	Calculated
job	Card holder job	Categorical	Dataset
city	Card holder city	Categorical	Dataset
city_pop	Card holder city population	Numerical	Dataset
state	Card holder state	Categorical	Dataset
lat	Card holder latitude	Numerical	Dataset
lon	Card holder longitude	Numerical	Dataset
category	Merchant category	Categorical	Dataset
merch_lat	Merchant latitude	Numerical	Dataset
merch_lon	Merchant longitude	Numerical	Dataset
amt	Transaction amount	Numerical	Dataset
trans_hour	Transaction hour	Numerical	Calculated
day_of_week	Transaction day of week	Categorical	Calculated
month	Transaction month	Numerical	Calculated
is_fraud	Transaction fraud flag	Categorical	Dataset

For each dataset, we held experiments in three different scenarios: the original dataset, containing all attributes, named henceforth HC; the original dataset without HC attributes, with more than 100 different values, named henceforth No-HC; and a new dataset, named henceforth VCCA, where all categorical attributes, including HC ones, from the original dataset were grouped by VCCA using a Similarity S of 99.9999%.

Figure 6 presents an outline of the experiments. We have used stratified 10-fold cross validation on each dataset (I), generating 10 folds with the same fraud ratio (II). From these folds, 10 different sets of training and test datasets are generated (III). For each set, stratified sampling is used to generate a validation dataset (IV). At this point, the training and the validation datasets are used by the VCCA algorithm to generate the conversion table. The training and the validation dataset are then used to train the FFN (V) and the resulting model is evaluated on the test dataset. Training and evaluation are executed 100 times for each dataset.

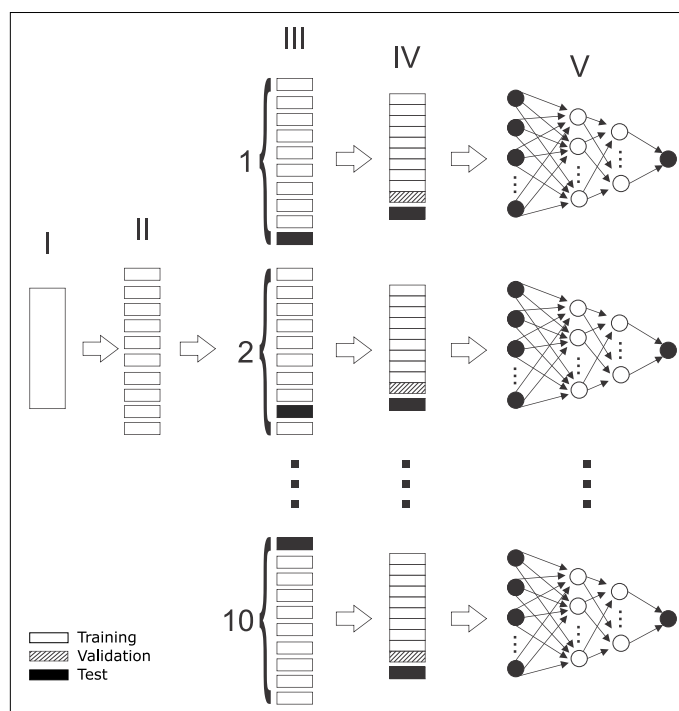


Figure 6. Experiments.

Results of the different models were compared by using the Wilcoxon signed-ranked test, as recommended by Demsar [52], with a significance level of 0.05. The entire process of FFN training and evaluation followed recommendations made by Flexer [53]: using different training and test datasets; having multiple runs with a resampling technique, in our case 10-fold cross-validation; reporting mean, variance, and confidence interval; and computing a statistical test, the Wilcoxon signed-ranked test, for a comparison of performances.

Two different optimization methods were considered to train the FFN: Adam [54], and RMSProp [55]. Default values were used for most hyperparameters. For Adam, the learning rate was set to 0.001, beta 1 was set to 0.9, beta 2 was set to 0.999, and epsilon was set to 1×10^{-7} . Finally, for RMSProp, the learning rate was set to 0.001, rho was set to 0.9, and epsilon was set to 1×10^{-7} .

For each dataset, the following FFN hyperparameters were tuned by exploring different configurations: optimizer, hidden layer size, dropout rate, and l2 regularizer. For most datasets, models without a dropout rate performed better. The maximum number of epochs was defined as 100 with an early stop configured to watch the PRC metric applied to the validation set, with a patience of 10 epochs without increase.

5.2. Dataset 1

Table 3 presents the number of values, $|v|$, and IG for all categorical attributes in the original training dataset for the first fold. It also contains the number of values, $|v'|$, and IG' , obtained by applying the VCCA. Per a Non-Disclosure Agreement (NDA), attribute names and values were omitted. Using one hot encoding resulted in 4875 binary attributes for the HC dataset and 104 for the VCCA dataset.

Table 4 presents hyperparameters used for this dataset. The FFN presents similar configurations with differences only in the number of neurons at each hidden layer.

Table 3. The categorical attributes and their *IG* from Dataset 1.

Attribute	$ v $	<i>IG</i>	$ v' $	<i>IG'</i>
c	587	0.020087454009060313	9	0.020087454009060313
d	199	0.01038105195206505	9	0.01038105195206505
e	20	0.005017041759478355	7	0.005017041759478355
f	2	0.0001723760259424098	2	0.0001723760259424098
g	13	0.013065745196519488	6	0.013065745196519488
h	18	0.010828126068959505	5	0.010828126068959505
i	185	0.012616077454649672	10	0.012616077454649668
j	4	0.0034157174977478555	4	0.0034157174977478555
k	428	0.018796128663501527	7	0.018796128663501527
l	25	0.003151780053255858	5	0.003151780053255858
m	1575	0.007888460862140696	4	0.007888460862140696
o	5	0.003897359833765539	4	0.003897359833765539
p	188	0.012821572812932829	11	0.012821572812932829
q	18	0.009203556840586557	5	0.00920355684058656
r	1578	0.007300173818738483	7	0.007300173818738483
s	26	0.0029986812882387	5	0.0029986812882387033
t	4	0.002688904270427419	4	0.002688904270427419

Table 4. The hyperparameters used for Dataset 1.

	No-HC	HC	VCCA
First Hidden Layer Size	64	32	64
Second Hidden Layer Size	16	32	8
Optimizer	RMSProp	RMSProp	RMSProp
L2 Regularizer	0.001	0.001	0.001

Evaluation

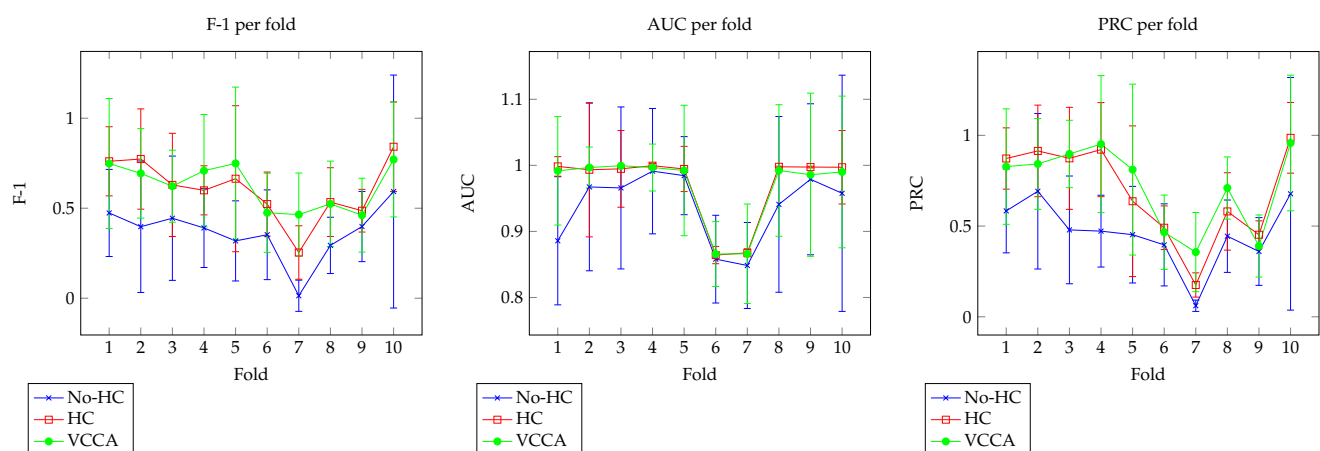
Table 5 presents metrics applied to each dataset, after applying the trained FFN on the test datasets. Per the Wilcoxon signed-ranks test [52], both HC and VCCA trained FFN models presented significantly higher values for all metrics. There was also no significant disparity in metrics values between HC and VCCA trained FFN models. Table 6 presents the variance for the same metrics. Figure 7 presents mean values for metrics applied on each fold and a 95% confidence interval.

Table 5. The Mean of Metrics for Dataset 1.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.4737	0.7607	0.7486	0.8858	0.9984	0.9917	0.5836	0.8724	0.8278
2	0.3973	0.7732	0.6936	0.9673	0.9933	0.9967	0.6919	0.9142	0.8425
3	0.4444	0.6295	0.6222	0.9658	0.9947	0.9994	0.479	0.8733	0.8974
4	0.3905	0.5994	0.7084	0.9912	0.9995	0.9967	0.4721	0.9214	0.9518
5	0.3184	0.6639	0.7489	0.9844	0.9945	0.9922	0.4526	0.637	0.8113
6	0.3526	0.523	0.4746	0.858	0.8642	0.8657	0.3964	0.4909	0.4658
7	0.0137	0.254	0.4646	0.8484	0.8676	0.8661	0.061	0.1754	0.3567
8	0.2935	0.534	0.5241	0.9409	0.9978	0.9922	0.4442	0.5807	0.7098
9	0.3982	0.4862	0.4614	0.9789	0.9974	0.9858	0.3601	0.4522	0.3901
10	0.5926	0.8416	0.7705	0.9577	0.9971	0.99	0.6779	0.9864	0.958

Table 6. The Variance of Metrics for Dataset 1.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.0152	0.0096	0.0339	0.0025	0.0001	0.0018	0.0139	0.0074	0.0263
2	0.0347	0.0202	0.0161	0.0042	0.0027	0.0003	0.0478	0.0166	0.0163
3	0.031	0.0214	0.0104	0.0039	0.0009	0	0.0229	0.0206	0.009
4	0.0126	0.0048	0.0254	0.0023	0	0.0003	0.0102	0.0175	0.0371
5	0.0129	0.0428	0.0468	0.0009	0.0003	0.0025	0.0184	0.0449	0.0577
6	0.0162	0.0082	0.0126	0.0011	0	0.0006	0.0133	0.0038	0.011
7	0.002	0.0057	0.0139	0.0011	0	0.0015	0.0003	0.0012	0.0124
8	0.0064	0.0095	0.0147	0.0046	0	0.0026	0.0103	0.0118	0.0076
9	0.0099	0.0037	0.011	0.0034	0	0.004	0.0091	0.0015	0.0076
10	0.109	0.0161	0.0265	0.0083	0.0008	0.0034	0.107	0.0099	0.0363

**Figure 7.** Mean values for the metrics applied on Dataset 1.

The worst training times are presented in Table 7 for each scenario and fold. Data transformation took at most 0.52 s, 2.24 s, and 3.34 s for No-HC, HC, and VCCA datasets. The latter dataset required more time for data transformation because of the VCCA algorithm. Data transformation times were included in the training time. Folds required significantly lower training times for both the No-HC and VCCA datasets, when compared to the HC dataset. This can be explained by the fact that hundreds of dummy inputs were generated for the categorical attributes of the HC dataset, which significantly raised its FFN complexity. Training times were significantly higher for the VCCA dataset, when compared to the No-HC dataset, because it required additional transformations.

Table 7. The training times in seconds for Dataset 1.

Fold	No-HC	HC	VCCA	VCCA (%)
1	12.94	26.02	14.68	56.42
2	13.43	22.76	15.20	66.78
3	12.58	24.11	15.07	62.50
4	13.27	20.31	13.44	66.17
5	15.66	33.63	15.37	45.70
6	14.70	30.88	18.67	60.46
7	16.03	28.15	19.25	68.38
8	13.71	18.63	15.48	83.09
9	17.37	30.59	19.16	62.63
10	17.96	27.58	18.75	67.98

In summary, in this set of experiments, we were able to verify that the inclusion of HC attributes improved fraud-detection quality, measured through F-1, AUC, and PRC metrics. We also verified that the VCCA algorithm was able to reduce training times, while maintaining the benefits obtained with the inclusion of HC attributes.

5.3. Dataset 2

Table 8 presents the number of values, $|v|$, and IG for all categorical attributes in the original training dataset for the first fold. It also contains the number of values, $|v'|$, and IG' , obtained by applying the VCCA. Per the same NDA of Dataset 1, attribute names and values were omitted. Using one hot encoding resulted in 10384 binary attributes for the HC dataset and 108 for the VCCA dataset.

Table 8. The categorical attributes and their IG from Dataset 2.

Attribute	$ v $	IG	$ v' $	IG'
a	2	0.00019165172400172417	2	0.00019165172400172417
b	3	0.0005656457858254846	3	0.0005656457858254846
c	414	0.010965553523071556	6	0.010965553523071556
d	128	0.008359619642109994	6	0.008359619642109994
e	20	0.004630761867383492	4	0.004630761867383492
f	3	0.00028618741327195163	2	0.00028618741327195163
g	28	0.00395328094102812	3	0.00395328094102812
h	53	0.00625993642734945	5	0.006259936427349448
i	290	0.010018709402812331	9	0.010018709402812331
j	6	0.004324161191009918	3	0.004324161191009918
k	1295	0.010829713469237199	10	0.010829713469237199
l	26	0.004875998819042033	4	0.004875998819042033
m	3893	0.006676635146988344	6	0.006676635146988344
n	2	0.0000003671349237989452	2	0.0000003671349237989452
o	5	0.0005324192509037727	3	0.0005324192509037727
p	287	0.00850518645299473	13	0.00850518645299473
q	51	0.004936240899563283	5	0.004936240899563283
r	3844	0.006550636310641834	11	0.006550636310641832
s	27	0.004161004607392469	7	0.004161004607392469
t	7	0.0031338015517482255	4	0.0031338015517482255

Table 9 presents hyperparameters used for this dataset. The FFN model for the VCCA dataset additionally used a drop rate of 0.2.

Table 9. The hyperparameters used for Dataset 2.

	No-HC	HC	VCCA
First Hidden Layer Size	64	64	64
Second Hidden Layer Size	32	16	4
Optimizer	RMSProp	adam	RMSProp
L2 Regularizer	0.001	0.001	0.001

Evaluation

Table 10 presents metrics applied to each dataset, after using the trained FFN on the test dataset. Per the Wilcoxon signed-ranks test [52], both HC and VCCA trained FFN models presented significantly higher values for the F-1 score, without any significant disparity between them. For values obtained from the AUC metric, the HC trained FFN models presented significantly higher values, compared to the other models. Conversely, the VCCA trained FFN model had significantly lower values than the others for the AUC metric. Regarding the PRC metric, there was no significant disparity between values when comparing the FFN models trained for HC and VCCA with the one trained for No-HC. Conversely, the HC trained FFN model has presented significantly higher values for this

metric, when compared directly with the one trained for the VCCA. Table 11 presents the variance for the metrics. Figure 8 presents mean values for metrics applied on each fold and a 95% confidence interval.

Table 10. The Mean of Metrics from Dataset 2.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.2593	0.6212	0.6252	0.9981	0.9993	0.9892	0.3748	0.5059	0.473
2	0.2521	0.4798	0.4193	0.9987	0.9989	0.9889	0.4114	0.4724	0.4951
3	0.2191	0.2671	0.3705	0.9995	0.9994	0.9843	0.6161	0.5671	0.5381
4	0.2582	0.5402	0.4944	0.9977	0.9955	0.9785	0.4372	0.5029	0.401
5	0.3362	0.5012	0.5006	0.9954	0.9987	0.9583	0.53	0.5683	0.5691
6	0.3365	0.479	0.4398	0.9984	0.999	0.964	0.4404	0.5194	0.4698
7	0.2789	0.4431	0.3537	0.9989	0.9987	0.9633	0.5533	0.4712	0.379
8	0.2895	0.534	0.5486	0.9989	0.9993	0.9743	0.6592	0.6071	0.5366
9	0.2296	0.6152	0.5454	0.9992	0.9996	0.9496	0.559	0.6517	0.5958
10	0.1964	0.5033	0.5351	0.9999	1	0.965	0.8968	1	0.9299

Table 11. The Variance of Metrics from Dataset 2.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.0084	0.0035	0.0117	0	0	0.0049	0.0022	0.0009	0.0072
2	0.0029	0.0038	0.0087	0	0	0.0049	0.002	0.0007	0.0205
3	0.003	0.0035	0.0084	0	0	0.0073	0.0034	0.0029	0.0209
4	0.0032	0.0032	0.0144	0	0.0011	0.0096	0.0031	0.0025	0.0088
5	0.0063	0.0058	0.0322	0	0	0.0185	0.0022	0.0022	0.0341
6	0.0066	0.0009	0.0154	0	0	0.0164	0.0011	0.0022	0.0197
7	0.0062	0.0005	0.0153	0	0	0.0163	0.0037	0.0019	0.0287
8	0.003	0.0004	0.0188	0	0	0.012	0.0047	0.0017	0.0189
9	0.0036	0.0258	0.0642	0	0	0.0227	0.0048	0.0012	0.0426
10	0.005	0.0006	0.0268	0	0	0.0164	0.004	0	0.0655

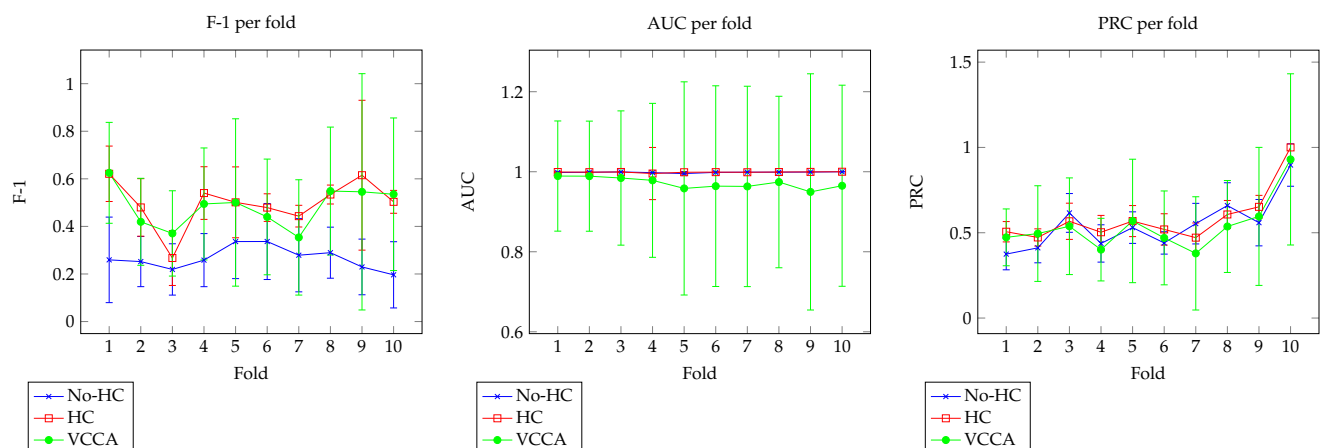


Figure 8. Mean values for the metrics applied on Dataset 2.

The worst training times are presented in Table 12 for each scenario and fold. Data transformation took at most 1.26 s, 8.25 s, and 4.52 s for No-HC, HC, and the VCCA datasets. Unexpectedly, the HC dataset required more time for data transformation than the VCCA dataset. In this instance, this happened because one hot encoding took much more time processing HC attributes. Data transformation times were included in the training time. Folds required significantly lower training times for both the No-HC and the

VCCA datasets, when compared to the HC dataset. This can be explained by the fact that hundreds of dummy inputs were generated for the categorical attributes of the HC dataset, which significantly raised its FFN complexity. Training times were significantly higher for the VCCA dataset, when compared to the No-HC dataset, because it required additional transformation.

Table 12. The training times in seconds for Dataset 2.

Fold	No-HC	HC	VCCA	VCCA (%)
1	14.71	116.47	18.20	15.63
2	10.14	44.74	14.04	31.38
3	15.56	55.40	15.71	28.36
4	17.46	78.39	20.66	26.36
5	18.45	106.61	22.38	20.99
6	18.46	79.64	19.87	24.95
7	22.53	85.16	25.43	29.86
8	21.76	92.89	24.14	25.99
9	19.66	56.55	20.08	35.51
10	20.46	45.92	19.72	42.94

In summary, in this set of experiments, we were able to verify that the inclusion of HC attributes improved fraud-detection quality, measured through the F-1 metric. For the AUC metric, the inclusion of HC attributes improved fraud detection only for the dataset where VCCA was not applied. Moreover, there was no significant improvement in fraud detection with the inclusion of HC attributes when considering the PRC metric. We were also able to verify that the VCCA algorithm reduced training times, while maintaining the benefits obtained with the inclusion of HC attributes, but only for the F-1 metric.

5.4. Dataset 3

Table 13 presents the number of values, $|v|$, and IG for all categorical attributes in the original training dataset for the first fold. It also includes the number of values, $|v'|$, and IG' , obtained by applying the VCCA. Per a Non-Disclosure Agreement (NDA), attribute names and values were also omitted. Using one hot encoding resulted in 10384 binary attributes for the HC dataset and 108 for the VCCA dataset.

Table 13. The categorical attributes and their IG for Dataset 3.

Attribute	$ v $	IG	$ v' $	IG'
category	14	0.0019799792770756885	14	0.0019799792770756885
gender	2	0.0000006409831799781	2	0.0000006409831799781
city	849	0.0103333005133997846	177	0.0103332999129037914
state	50	0.0006413767623995176	45	0.0006413767623995245
job	478	0.0060285748349879244	166	0.0060285744210841971
day_of_week	7	0.0001846527228808251	7	0.0001846527228808251

Table 14 presents hyperparameters used for this dataset. The FFN presented similar configurations with differences only in the number of neurons in the first hidden layer for the No-HC dataset.

Table 14. The hyperparameters used for the Dataset 3.

	No-HC	HC	VCCA
First Hidden Layer Size	64	32	32
Second Hidden Layer Size	16	16	16
Optimizer	adam	adam	adam
L2 Regularizer	0.001	0.001	0.001

Evaluation

Table 15 presents metrics applied to each dataset after using the trained FFN on the test set. Per the Wilcoxon signed-ranks test [52], both HC and VCCA trained FFN models presented significantly higher values for all metrics. The VCCA model presented significantly lower values for the F-1 and PRC metrics, but significantly higher values for the AUC metric, compared to the HC model. Table 16 presents the variance for the metrics. Figure 9 presents mean values for metrics applied on each fold and a 95% confidence interval.

Table 15. The Mean of Metrics from Dataset 3.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.2298	0.4387	0.4317	0.9917	0.9904	0.9905	0.733	0.8394	0.8229
2	0.2331	0.4437	0.4383	0.9825	0.9905	0.9923	0.6973	0.8152	0.8002
3	0.2514	0.4461	0.4194	0.9937	0.9985	0.9985	0.7342	0.8694	0.8467
4	0.2386	0.4361	0.4073	0.9908	0.9983	0.9985	0.7074	0.8353	0.8186
5	0.2263	0.429	0.4181	0.99	0.9978	0.998	0.6818	0.826	0.8093
6	0.2468	0.4297	0.4218	0.9918	0.996	0.9968	0.7306	0.8226	0.8024
7	0.239	0.445	0.4205	0.9943	0.9988	0.999	0.7329	0.8573	0.8477
8	0.2255	0.4564	0.4324	0.9943	0.998	0.9984	0.7484	0.8657	0.843
9	0.2385	0.4366	0.3889	0.9927	0.9984	0.9982	0.7519	0.8755	0.841
10	0.2383	0.4442	0.4269	0.9923	0.9981	0.9982	0.7543	0.8358	0.8236

Table 16. The Variance of Metrics from Dataset 3.

Fold	F-1			AUC			PRC		
	No-HC	HC	VCCA	No-HC	HC	VCCA	No-HC	HC	VCCA
1	0.0029	0.0044	0.0053	0	0	0	0.0006	0.0007	0.0011
2	0.0025	0.0049	0.005	0	0	0	0.0004	0.0008	0.0009
3	0.0036	0.005	0.0045	0	0	0	0.0007	0.0006	0.0008
4	0.0027	0.0054	0.0052	0	0	0	0.0007	0.0008	0.001
5	0.0026	0.0054	0.0058	0	0	0	0.0011	0.0014	0.0012
6	0.0024	0.0062	0.005	0	0	0	0.0006	0.0013	0.0012
7	0.0032	0.0041	0.0045	0	0	0	0.0006	0.0009	0.0009
8	0.0026	0.0051	0.0054	0	0	0	0.0007	0.0006	0.0009
9	0.0029	0.0049	0.0046	0	0	0	0.0009	0.0006	0.001
10	0.0023	0.0044	0.0051	0	0	0	0.0007	0.0007	0.001

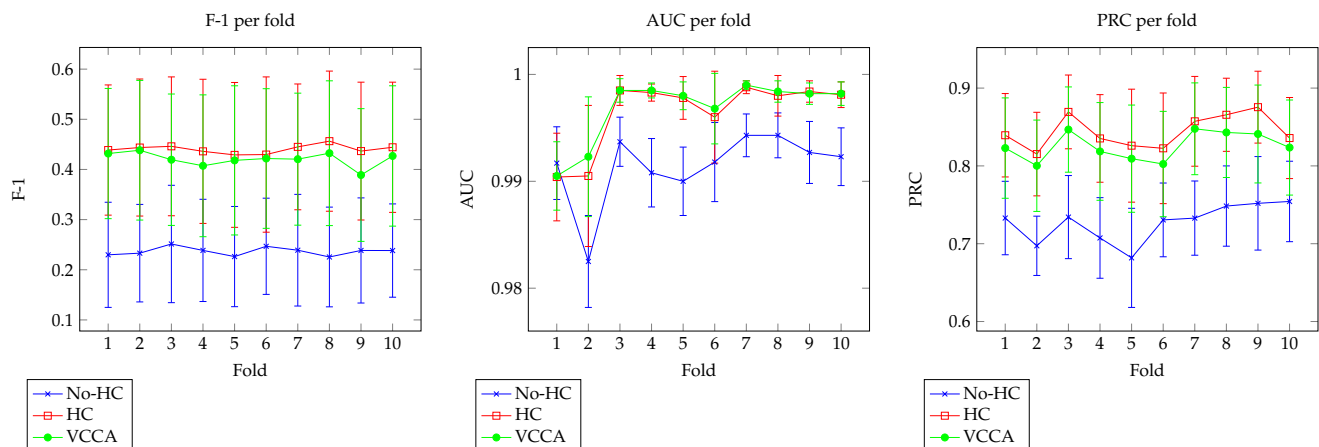


Figure 9. Mean values for the metrics applied on dataset 3.

The worst training times are presented in Table 17 for each scenario and fold. Data transformation took at most 5.6 s, 20.12 s, and 21.36 s for No-HC, HC, and the VCCA datasets. As expected, the VCCA dataset took more time for data transformation. Data transformation times were included in the training time. Folds required significantly lower training times for both the No-HC and the VCCA datasets, when compared to the HC dataset. This can be explained by the fact that hundreds of dummy inputs were generated for the categorical attributes of the HC dataset, which significantly raised its FFN complexity. Training times were significantly higher for the VCCA dataset, when compared to the No-HC dataset, because it required additional transformation.

Table 17. The training times in seconds for Dataset 3.

Fold	No-HC	HC	VCCA	VCCA (%)
1	164.94	391.94	232.28	59.26
2	195.97	380.57	243.34	63.94
3	162.02	336.06	226.62	67.43
4	181.93	314.25	238.56	75.91
5	166.18	356.02	240.85	67.65
6	183.79	353.9	232.66	65.74
7	162.81	361.18	273.42	75.70
8	196.52	418.69	244.34	58.36
9	214.63	345.03	285.8	82.83
10	206.16	410.03	266.18	64.91

In summary, in this set of experiments, we were able to verify that the inclusion of HC attributes improved fraud-detection quality, measured through all metrics. We also verified that the VCCA algorithm was able to reduce training times, while maintaining the benefits obtained with the inclusion of HC attributes for the F-1 metric.

6. Conclusions

In this article, we analyzed the impact of High-Cardinality (HC) attributes on credit card fraud detection. In order to achieve that, we trained a deep fully connected FFN for three datasets in three scenarios: without HC attributes, including HC attributes, and including HC attributes transformed using the new proposed algorithm named Value Clustering for Categorical Attributes (VCCA).

Results indicate that the inclusion of HC attributes can improve detection quality, measured by F-1, in credit card fraud detection. For other metrics, such as AUC and PRC, there are cases, such as the experiment run on Dataset 2, where including HC attributes will not bring any significant benefits.

The VCCA, Value Clustering for Categorical Attributes, approach presented in this article, has shown promising results by allowing the inclusion of HC attributes with reduced training times without significantly compromising detection quality, which can reduce costs. The algorithm has provided the automatic creation of a conversion table that could be verified and modified by fraud investigators.

This article focuses solely on credit card fraud detection using the Deep Learning domain. Future works could verify the impact of HC attributes, as well as the applicability of the VCCA algorithm, in other knowledge domains. The use of other Machine Learning classifiers, such as Random Forests and Support-Vector Machines, could also be investigated. Finally, it would be desirable to make a comparison between approaches for dealing with HC attributes, such as the ones mentioned in Section 3.

Author Contributions: E.M.C.: Conceptualization, Methodology, Software, Investigation, Data Curation, Writing—Original Draft, Visualization. C.H.Q.F.: Formal Analysis, Writing—Review & Editing. L.F.S.M.: Conceptualization, Methodology, Investigation, Validation. L.A.V.D.: Formal Analysis, Writing—Review & Editing, Project Administration, Funding Acquisition. A.M.d.C.:

Resources, Writing—Review & Editing, Supervision, Project Administration, Funding Acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: General and financial support during this investigation: the Brazilian Aeronautics Institute of Technology (ITA); the Casimiro Montenegro Filho Foundation (FCMF); the 2RP Net Enterprise; and the Brazilian Ministry of Education (Ministério da Educação-MEC).

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <https://www.kaggle.com/kartik2112/fraud-detection>, accessed on 4 July 2022.

Acknowledgments: The authors wish to thank the Brazilian Aeronautics Institute of Technology (ITA), the Casimiro Montenegro Filho Foundation (FCMF), the 2RP Net Enterprise, and the Brazilian Ministry of Education (Ministério da Educação - MEC) for their support and research infrastructure and also the Software Engineering Research Group (GPES) members for their assistance and advice. We also wish to thank professor Paulo Marcelo Tasinoffo for his support and insightful comments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNP	Card-Not-Present
ML	Machine Learning
MCC	Merchant Category Code
VCCA	Value Clustering for Categorical Attributes
PIN	Personal Identification Number
HC	High-Cardinality
CBM	Conjugate Bayesian Model
ANN	Artificial Neural Network
OFP	Optimal Family of Partitions
OSVP	Optimal Symbolic Value Partition
AVT	Attribute Value Taxonomy
NLP	Natural Language Processing
FFN	FeedForward Networks
MLP	MultiLayer Perceptron
ReLU	Rectified Linear Activation Function
IG	Information Gain
TPR	True Positive Rate
ROC	Receiver Operating Characteristic
FPR	False Positive Rate
AUC	Area Under the ROC Curve
PRC	Area Under the Precision-Recall Curve
ZIP	Zone Information Postal
NDA	Non-Disclosure Agreement

References

1. Jurgovsky, J.; Granitzer, M.; Ziegler, K.; Calabretto, S.; Portier, P.E.; He-Guelton, L.; Caelen, O. Sequence classification for credit-card fraud detection. *Expert Syst. Appl.* **2018**, *100*, 234–245. <https://doi.org/10.1016/j.eswa.2018.01.037>.
2. HSN Consultants, Inc. *Card Fraud Losses Reach \$22.80 Billion*; Technical Report 1118; The Nilson Report, Issue 1118, Oxnard, CA, USA, 2017. Available online: https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1118 (accessed on 4 July 2022).
3. Knief, B. *2016 Global Consumer Card Fraud: Where Card Fraud Is Coming From*; Technical Report; Aite Group LLC, Boston, MA, USA, 2016. Available online: <https://aite-novarica.com/report/2016-global-consumer-card-fraud-where-card-fraud-coming> (accessed on 4 July 2022).
4. Sohony, I.; Pratap, R.; Nambiar, U. Ensemble Learning for Credit Card Fraud Detection. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD '18), Goa, India, 11–13 January 2018; ACM: New York, NY, USA, 2018; pp. 289–294. <https://doi.org/10.1145/3152494.3156815>.
5. Gadi, M.F.A.; Wang, X.; do Lago, A.P. Credit Card Fraud Detection with Artificial Immune System. In Proceedings of the Artificial Immune Systems, Phuket, Thailand, 10–13 August 2008; Bentley, P.J., Lee, D., Jung, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 119–131.

6. Singh, R.; Rani, R. Comparative Evaluation of Predictive Modeling Techniques on Credit Card Data. *Int. J. Comput. Theory Eng.* **2011**, *598*–603. <https://doi.org/10.7763/IJCTE.2011.V3.377>.
7. Ngai, E.W.T.; Hu, Y.; Wong, Y.H.; Chen, Y.; Sun, X. The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and an Academic Review of Literature. *Decis. Support Syst.* **2011**, *50*, 559–569. <https://doi.org/10.1016/j.dss.2010.08.006>.
8. Pozzolo, A.D.; Caelen, O.; Borgne, Y.A.L.; Waterschoot, S.; Bontempi, G. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Syst. Appl.* **2014**, *41*, 4915–4928. <https://doi.org/10.1016/j.eswa.2014.02.026>.
9. Fadaei Noghani, F.; Moattar, M. Ensemble Classification and Extended Feature Selection for Credit Card Fraud Detection. *J. AI Data Min.* **2017**, *5*, 235–243. <https://doi.org/10.22044/jadm.2016.788>.
10. Vlasselaer, V.V.; Bravo, C.; Caelen, O.; Eliassi-Rad, T.; Akoglu, L.; Snoeck, M.; Baesens, B. APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decis. Support Syst.* **2015**, *75*, 38–48. <https://doi.org/10.1016/j.dss.2015.04.013>.
11. Bahnsen, A.C.; Aouada, D.; Stojanovic, A.; Ottersten, B. Feature engineering strategies for credit card fraud detection. *Expert Syst. Appl.* **2016**, *51*, 134–142. <https://doi.org/10.1016/j.eswa.2015.12.030>.
12. Wang, C.; Han, D. Credit card fraud forecasting model based on clustering analysis and integrated support vector machine. *Clust. Comput.* **2018**. <https://doi.org/10.1007/s10586-018-2118-y>.
13. Somasundaram, A.; Reddy, S. Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance. *Neural Comput. Appl.* **2018**. <https://doi.org/10.1007/s00521-018-3633-8>.
14. Mahmoudi, N.; Duman, E. Detecting credit card fraud by Modified Fisher Discriminant Analysis. *Expert Syst. Appl.* **2015**, *42*, 2510–2516. <https://doi.org/10.1016/j.eswa.2014.10.037>.
15. Zareapoor, M.; Shamsolmoali, P. Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier. *Procedia Comput. Sci.* **2015**, *48*, 679–685. <https://doi.org/10.1016/j.procs.2015.04.201>.
16. Bekirev, A.S.; Klimov, V.V.; Kuzin, M.V.; Shchukin, B.A. Payment card fraud detection using neural network committee and clustering. *Opt. Mem. Neural Netw.* **2015**, *24*, 193–200. <https://doi.org/10.3103/S1060992X15030030>.
17. Juszczak, P.; Adams, N.M.; Hand, D.J.; Whitrow, C.; Weston, D.J. Off-the-peg and bespoke classifiers for fraud detection. *Comput. Stat. Data Anal.* **2008**, *52*, 4521–4532. <https://doi.org/10.1016/j.csda.2008.03.014>.
18. Moeyersoms, J.; Martens, D. Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector. *Decis. Support Syst.* **2015**, *72*, 72–81.
19. Perlich, C.; Provost, F. Distribution-based aggregation for relational learning with identifier attributes. *Mach. Learn.* **2006**, *62*, 65–105. <https://doi.org/10.1007/s10994-006-6064-1>.
20. Muto, Y.; Kudo, M.; Murai, T. Reduction of Attribute Values for Kansei Representation. *JACIII* **2006**, *10*, 666–672. <https://doi.org/10.20965/jaciii.2006.p0666>.
21. Min, F.; Liu, Q.; Fang, C.; Zhang, J. Reduction Based Symbolic Value Partition. In Proceedings of the Advances in Hybrid Information Technology, First International Conference, ICHIT 2006, Jeju Island, Korea, 9–11 November 2006; pp. 20–30. https://doi.org/10.1007/978-3-540-77368-9_3.
22. Boullé, M. A robust method for partitioning the values of categorical attributes. In Proceedings of the Extraction et gestion des connaissances (EGC'2004), Clermont Ferrand, France, 20–23 January 2004; pp. 173–184.
23. Micci-Barreca, D. A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. *SIGKDD Explor. Newsl.* **2001**, *3*, 27–32. <https://doi.org/10.1145/507533.507538>.
24. Mougan, C.; Masip, D.; Nin, J.; Pujol, O. Quantile Encoder: Tackling High Cardinality Categorical Features in Regression Problems. In Proceedings of the Modeling Decisions for Artificial Intelligence: 18th International Conference, MDAI 2021, Umeå, Sweden, 27–30 September 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 168–180. https://doi.org/10.1007/978-3-030-85529-1_14.
25. Slakey, A.; Salas, D.; Schamroth, Y. Encoding Categorical Variables with Conjugate Bayesian Models for WeWork Lead Scoring Engine. *arXiv* **2019**, arXiv:1904.13001.
26. Guo, C.; Berkhahn, F. Entity Embeddings of Categorical Variables. *arXiv* **2016**, arXiv:1604.06737.
27. Dahouda, M.K.; Joe, I. A Deep-Learned Embedding Technique for Categorical Features Encoding. *IEEE Access* **2021**, *9*, 114381–114391. <https://doi.org/10.1109/ACCESS.2021.3104357>.
28. Arat, M.M. Learning From High-Cardinality Categorical Features in Deep Neural Networks. *J. Adv. Res. Nat. Appl. Sci.* **2022**, *8*, 222–236. <https://doi.org/10.28979/jarnas.1014469>.
29. Nguyen, H.S. Discretization of Real Value Attributes, Boolean Reasoning Approach. Ph.D. Thesis, Warsaw University, Warsaw, Poland, 1997.
30. Nguyen, S.H. Regularity Analysis and Its Applications in Data Mining. Ph.D. Thesis, Warsaw University, Warsaw, Poland, 1999.
31. Min, F.; Liu, Q.; Fang, C. Rough sets approach to symbolic value partition. *Int. J. Approx. Reason.* **2008**, *49*, 689–700.
32. Ye, M.; Wu, X.; Hu, X.; Hu, D. Knowledge reduction for decision tables with attribute value taxonomies. *Knowl.-Based Syst.* **2014**, *56*, 68–78.
33. Wen, L.; Min, F. A Granular Computing Approach to Symbolic Value Partitioning. *Fundam. Inform.* **2015**, *142*, 337–371. <https://doi.org/10.3233/FI-2015-1297>.
34. Cerda, P.; Varoquaux, G.; Kégl, B. Similarity Encoding for learning with dirty categorical variables. *Mach. Learn.* **2018**, *107*, 1477–1494. <https://doi.org/10.1007/s10994-018-5724-2>.

35. Cerda, P.; Varoquaux, G. Encoding High-Cardinality String Categorical Variables. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 1164–1176. <https://doi.org/10.1109/TKDE.2020.2992529>.
36. Weinberger, K.Q.; Dasgupta, A.; Attenberg, J.; Langford, J.; Smola, A.J. Feature Hashing for Large Scale Multitask Learning. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09), Montreal, QC, Canada, 14–18 June 2009; Association for Computing Machinery: New York, NY, USA; pp. 1113–1120. <https://doi.org/10.1145/1553374.1553516>.
37. Carneiro, E.M.; Dias, L.A.V.; Cunha, A.M.; Mialaret, L.F.S. Cluster Analysis and Artificial Neural Networks A Case Study in Credit Card Fraud Detection. In Proceedings of the 12th International Conference on Information Technology–New Generations, Las Vegas, NV, USA, 13–15 April 2015. <https://doi.org/10.1109/ITNG.2015.25>.
38. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366.
39. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst. (MCSS)* **1989**, *2*, 303–314. <https://doi.org/10.1007/BF02551274>.
40. Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed.; Prentice Hall: Englewood Cliffs, NJ, USA, 1998.
41. Adewumi, A.O.; Akinyelu, A.A. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *Int. J. Syst. Assur. Eng. Manag.* **2017**, *8*, 937–953.
42. Al-Hashedi, K.G.; Magalingam, P. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Comput. Sci. Rev.* **2021**, *40*, 100402. <https://doi.org/10.1016/j.cosrev.2021.100402>.
43. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2016.
44. Chollet, F. Keras. 2015. Available online: <https://keras.io> (accessed on 4 July 2022).
45. Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; Lecun, Y. What is the Best Multi-Stage Architecture for Object Recognition? In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; Volume 12. <https://doi.org/10.1109/ICCV.2009.5459469>.
46. Tikhonov, A.N.; Arsenin, V.Y. *Solutions of Ill-Posed Problems*; V. H. Winston & Sons.: Washington, DC, USA; John Wiley & Sons: New York, NY, USA, 1977; p. xiii+258. Translated from the Russian, Preface by translation editor Fritz John, Scripta Series in Mathematics.
47. 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. Res.* **2011**, *12*, 2825–2830.
48. Scriptor, M.W. Nested-Means map classes for statistical maps. *Ann. Assoc. Am. Geogr.* **1970**, *60*, 385–392. <https://doi.org/10.1111/j.1467-8306.1970.tb00727.x>.
49. Lichman, M. [dataset] UCI Machine Learning Repository. 2013. Available online: <https://archive.ics.uci.edu/ml/datasets/mushroom> (accessed on 14 July 2022).
50. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the ICML '06: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; ACM: New York, NY, USA, 2006; pp. 233–240. <https://doi.org/10.1145/1143844.1143874>.
51. Shenoy, K.; Brandon, H. [dataset] Credit Card Transactions Fraud Detection Dataset. 2020. Available online: <https://www.kaggle.com/kartik2112/fraud-detection> (accessed on 14 July 2022).
52. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
53. Flexer, A. Statistical Evaluation of Neural Network Experiments: Minimum Requirements and Current Practice. In Proceedings of the 13th European Meeting on Cybernetics and Systems Research, Vienna, Austria, 9–12 April 1996; Volume 2.
54. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
55. Tieleman, T.; Hinton, G. Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. 2012. Available online: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (accessed on 14 July 2022).