Erzincan Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
2022, 15(1), 145-167
ISSN: 1307-9085, e-ISSN: 2149-4584
Araştırma Makalesi

Erzincan University
Journal of Science and Technology
2022, 15(1), 145-167
DOI: 10.18185/erzifbed954466
Research Article

Analysis of the Effectiveness of Various Machine Learning, Artificial Neural Network and Deep Learning Methods in Detecting Fraudulent Credit Card Transactions

Esra ÇELİK*, Deniz DAL and Ferhat BOZKURT

Department of Computer Engineering, Faculty of Engineering, Ataturk University ,25240 Erzurum, Turkey

Geliş / Received: 18/06/2021, Kabul / Accepted: 11/08/2021

Abstract

A credit card is an important financial tool that has emerged in parallel with the developments in technology from the past to the present and has become an indispensable part of human life. The credit card has many advantages that can be listed as facilitating online shopping, providing installments in purchases, and preventing cash dependence. This is why the rate of use of credit cards worldwide is increasing day by day. On the other hand, there are some risks of the credit cards highlighted by security concerns. The fraudsters who access the identity and credit card information of the consumers through different means use it to shop online without the consumer's knowledge and gain an unfair advantage. Therefore, it is crucial to eliminate this security vulnerability that the fraudsters exploit and to develop an effective solution to the customer victimization experienced by e-commerce companies due to the fraudulent credit card transactions. With this motivation, the performance of the methods from different research fields was examined to explore the solution space in detail in terms of the problem at hand within the scope of this study. For this purpose, three machine learning algorithms (K-Nearest Neighbor, Naive Bayes, Support Vector Machine), two artificial neural network algorithms (Binary Classifier, Autoencoder), and two deep learning algorithms (Deep Autoencoder and Deep Neural Network Classifier) were implemented. The effectiveness of the algorithms in question was tested with a famous dataset widely used in the literature. Experimental results showed that the Deep Neural Network Classifier outperformed the other algorithms used in this study and the best study ever reported in the literature in detecting fraudulent credit card transactions when accuracy and AUROC performance criteria were taken into account.

Keywords: Credit card fraud, machine learning, artificial neural network, deep learning, classification, deep neural network classifier, AUROC, AUPRC

Çeşitli Makine Öğrenmesi, Yapay Sinir Ağı ve Derin Öğrenme Yöntemlerinin Sahte Kredi Kartı İşlemlerini Tespit Etkinliklerinin Analizi

Öz

Kredi kartı, geçmişten günümüze teknolojide yaşanan gelişmelere paralel olarak ortaya çıkan ve insan hayatının vazgeçilmez bir parçası haline gelen önemli bir üründür. Kredi kartının çevrimiçi alışverişi kolaylaştırmak, alışverişlerde taksitlendirme imkânı sağlamak ve nakit para bağımlılığının önüne geçmek şeklinde sıralanabilecek birçok avantajı mevcuttur. Bu nedenledir ki kredi kartlarının kullanım oranı dünya çapında gün geçtikçe artmaktadır. Öte yandan kredi kartlarının güvenlik kaygılarıyla öne çıkan bazı riskleri de söz konusudur. Farklı yöntemlerle tüketicilerin kimlik ve kredi kartı bilgilerine ulaşan dolandırıcılar bu bilgileri kullanarak tüketicinin haberi olmadan çevrimiçi alışveriş yapmakta ve haksız bir çıkar elde etmektedir. Dolayısıyla dolandırıcıların istismar ettikleri bu güvenlik zafiyetini boşa çıkarmak ve sahte kredi kartı işlemlerinden dolayı e-ticaret şirketlerinin yaşadığı müşteri mağduriyetine etkili bir çözüm geliştirebilmek önem taşımaktadır. Bu motivasyonla bu çalışma kapsamında ilgili problem açısından çözüm uzayını detaylıca keşfedebilmek için farklı araştırma alanlarına ait yöntemlerin performansı mercek altına alınmıştır. Bu amaçla üç makine öğrenmesi algoritması (K-En Yakın Komşu, Naive Bayes, Destek Vektör Makinesi), iki yapay sinir ağı algoritması (İkili Sınıflandırıcı, Otomatik Kodlayıcı) ve iki derin öğrenme algoritması (Derin Otomatik Kodlayıcı ve Derin Sinir Ağı Sınıflandırıcısı) gerçeklenmiştir. Söz konusu algoritmaların etkinliği literatürde

*Corresponding Author: esra.celik@atauni.edu.tr

yaygın olarak kullanılan ünlü bir veri seti ile test edilmiştir. Deneysel sonuçlar Derin Sinir Ağı Sınıflandırıcısının sahte kredi kartı işlemlerinin tespiti noktasında bu çalışmada kullanılan diğer algoritmaları ve literatürde şu ana kadar rapor edilmiş en iyi çalışmayı doğruluk ve AUROC başarım ölçütleri dikkate alındığında geride bıraktığını göstermiştir.

Anahtar Kelimeler: Kredi kartı sahteciliği, makine öğrenmesi, yapay sinir ağı, derin öğrenme, sınıflandırma, derin sinir ağı sınıflandırıcısı, AUROC, AUPRC

1. Introduction

In parallel with the developments in technology from the past to the present, many products have become an indispensable part of human life in different fields. One of these products is a credit card. Some of the advantages of the credit card are as follows: (1) It is the only payment option other than cryptocurrencies for online purchases. (2) It is a flexible form of payment that has the possibility of installment and prevents cash dependence. (3) The risk of catching an infectious disease in exchange-based shopping with the paper money and coins is minimized through contactless credit cards. (Especially due to the Covid-19 pandemic, which has affected the whole world since the beginning of 2020, the advantage (3) of credit cards has become even more noticeable.) Besides the above-listed benefits of the credit cards, there are some risks that are especially prominent with security concerns. Although the credit card security is protected to a certain degree by the pin technology for offline payments, it is hardly possible to say the same thing for online payments. With the development of mobile device technologies and the spread of the Internet, online shopping has become a much more preferred service in facilitating human life. This service is carried out with two main components: the user (consumer) and service provider (e-commerce site). Although it is not part of this service, there is a third secondary component called the fraudster. The fraudster aims to sabotage this service to create a security vulnerability and gain an unfair advantage. The security weakness that occurs due to the fraudster's presence begins with the seizure of the identity and credit card information of the consumer. Then, the fraudster uses this information to make purchases on e-commerce sites without the consumer's knowledge. Three different methods are often employed by the fraudsters to exploit the aforementioned security vulnerability.

1. The first method aims to physically capture the consumer's identity and credit card information through various means. For example, theft is one of these ways and is aimed at seizing the consumer's wallet.

2. The second method puts the serving end to its target. E-commerce sites may store some personal data, such as a credit card number, to make the next purchase more convenient for the consumers. Malicious individuals who aim to gain access to this stored information and acquire unfair financial earnings in this way make life difficult for the cardholders, the credit card companies, and the sites that store this data. It is also known that these individuals often benefit from malicious software while reaching their goals. As a result of cyber-attacks using such software, material and moral damages arise due to personal data seizure, and

satisfaction is replaced by grievances. In such a case, it is often not possible to renew the damaged brand image.

3. In the third method, the target is again the consumer, and a security vulnerability is created by installing malicious software on the devices (especially mobile devices) that people use for online shopping. The applications installed on mobile devices often demand specific permissions from the user. Many users blindly allow these requests, making their devices vulnerable to different types of attacks, such as paid calls and SMS sending (Wang et al., 2015). Besides, the application markets that allow direct download of mobile applications can also be used for this purpose. For example, a user who downloads an application from such markets unwittingly installs malicious software embedded in the background of the application by the attackers. Then this software can access the information on the device. Another unauthorized access is through viruses, worms, and trojans. Viruses come to the device with the infected application and spread to other programs on the device when the application runs. Worms spread across the network, often taking advantage of vulnerabilities in the software running on the networked computers and providing access to the devices on the network. Trojans, on the other hand, reach a device by acting as benign programs, but they perform malicious functions in the background (Kolter and Maloof, 2006). As a result of such access, attackers will inevitably easily obtain different types of personal data, such as credit card information, on a device.

This is why e-commerce companies providing online services need to recognize/distinguish fraudulent credit card transactions. If such fraudulent transactions can be detected with high accuracy, this security vulnerability that the fraudsters exploit is avoided.

Credit card fraud detection is an important problem that researchers focus on today due to the reasons detailed in the previous paragraphs. It is rarely possible to suspect a possible fraud by evaluating a single transaction data at the transaction time. For example, the difference between the shipping and billing addresses, the mismatch of the shipping address and the IP address, the presence of a suspicious e-mail address, and a multi-quantity or high-volume order may indicate a fraudulent transaction. On the other hand, credit card fraud is often known to be too difficult to be detected by simple tests such as in the example above (Dal Pozzolo et al., 2014). Therefore, a set of transaction data (consumer's billing address, shipping address, IP address, e-mail address, transaction date, transaction time, transaction amount, etc.) is stored in the service provider's database after each credit card transaction for later analysis. The automatic analysis of such a dataset and a label attached to every transaction data as "fraudulent" or "legitimate" for classification purposes appears as a research problem planned to be solved within the scope of this study.

Studies on the detection of credit card fraud are frequently encountered in the literature. Many of these studies have examined the impact of machine learning and deep learning algorithms on the analysis of the fraudulent transactions performed with malicious software (Kolter and Maloof, 2006; Schultz et al. 2001; Itoo and Singh, 2020; Han et al., 2020; Lebichot et al., 2019; Carcillo et al., 2018b; Dal Pozzolo et al., 2015; Lopez and Cadavid, 2016; Awoyemi et al., 2017; Mukhandi, 2018; Gitonga, 2018; Soylu, 2018; Lakshmi and Kavilla, 2018; Meker, 2018). In the study carried out by Han et al. (2020), a machine learning-based method using the support

vector machine was utilized. Li et al. (2018) showed that instead of extracting and analyzing all the data in the dataset, it is possible to identify the most essential features that can be effective in distinguishing between legitimate and fraudulent data, thereby reducing the number of features. Another study used deep learning approaches to detect credit card fraud and focused on the applicability of classification models learned in a specific transaction category, such as e-commerce, to face-to-face transactions (Lebichot et al., 2019). Carcillo et al. (2018a) examined the impact of active learning strategies on detecting credit card fraud. They emphasized that it was crucial to select a sufficient number of cardholders in the labeling process when the transaction data is stable (Carcillo et al., 2018a). In another study, fraud models were studied by a hybrid method combining supervised and unsupervised learning techniques that can be learned from the analysis of past transactions (Carcillo et al., 2019). Techniques for fraud detection that integrate open source big data tools with machine learning approaches have also been developed (Carcillo et al., 2018b). Dal Pozzolo et al. (2014) proposed an effective learning strategy that utilized the verification delay and feedback interaction for the classification problem in fraud detection (Dal Pozzolo et al., 2017). How inadequate sampling in unstable datasets used for fraud detection affects a machine learning model's potential probability was the subject of another research (Dal Pozzolo et al., 2015). Lopez and Cadavid (2016) detected malicious software by analyzing the requested permissions for mobile devices with different machine learning classifier algorithms.

It is known that the variety of methods used in scientific research is important for exploring the solution space (Aydoğdu et al., 2017). For this reason, in this study, various techniques based on machine learning, artificial neural network, and deep learning were analyzed to develop an effective solution to the customer victimization experienced by e-commerce companies due to fraudulent credit card transactions. For this purpose, three machine learning algorithms (*K-Nearest Neighbor*, *Naive Bayes*, *Support Vector Machine*), two artificial neural network algorithms (*Binary Classifier*, *Autoencoder*), and two deep learning algorithms (*Deep Autoencoder and Deep Neural Network Classifier*) were taken into consideration, and all of these algorithms were implemented with Python programming language and the *scikit-learn* library. To the best of our knowledge, there is no other study available in the literature, using so many algorithms from three different research areas for the problem at hand.

The effectiveness of these algorithms was tested with a famous dataset called *creditcard*, which is widely used in the literature (Kaggle Machine Learning Group, 2020). Experimental results showed that the Deep Neural Network Classifier outperformed the other algorithms used in this study and the best study ever reported in the literature in detecting fraudulent credit card transactions when accuracy and AUROC performance criteria were taken into account. This performance increase is considered as another advantage of this study.

The rest of this paper is organized as follows. In Section 2, material and method are introduced. Experimental results and findings are discussed with detailed charts and tables in Section 3. Finally, Section 4 concludes the paper and presents some future work opportunities.

2. Materials and Methods

The design flow diagram of this study, which was carried out for the detection of fraudulent credit card transactions through machine learning, artificial neural network, and deep learning methods, is given in Figure 1. Each step in this diagram is detailed in the following subsections.

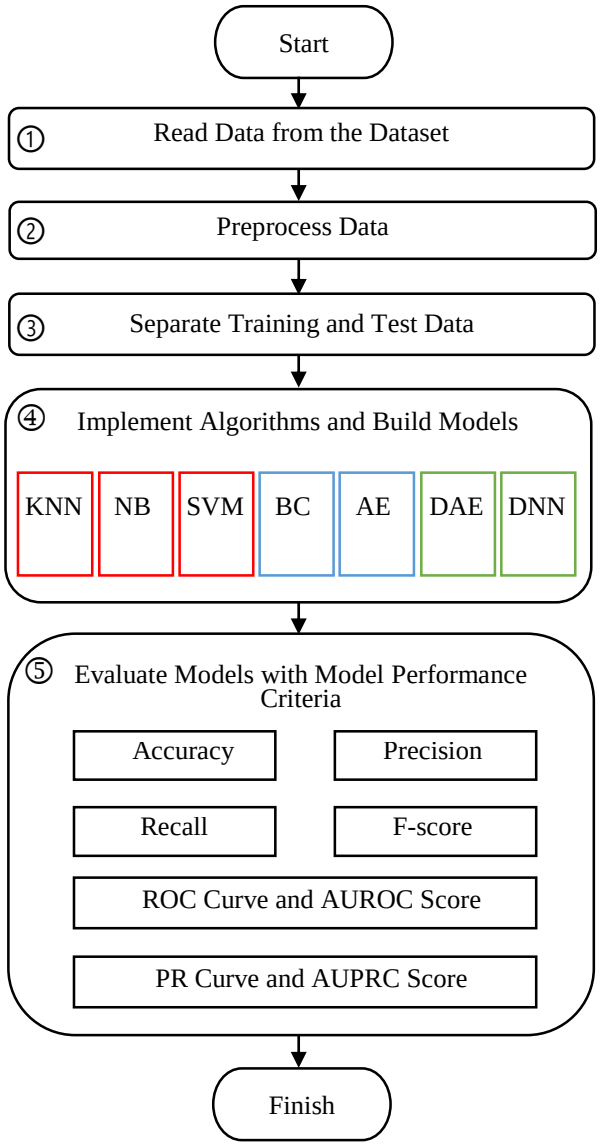

Figure 1. Design flow diagram.

2.1. Dataset

To analyze the performance of the methods evaluated in this study, a famous dataset called *creditcard* was used in step ① of the design flow. *creditcard* is an open-access data source that can be obtained through the Kaggle platform (Kaggle Machine Learning Group, 2020). This dataset includes a total of 284807 transactions made by credit cardholders living in Europe with a credit card within two days in September 2013. 492 of these transactions are labeled as fraudulent. The number of fraudulent transactions in the dataset has a fairly low rate, accounting for 0.172% of all transactions. These types of datasets, in which the ratio of one class to another is very low, are called unbalanced datasets. It is reported that the classification process with unbalanced datasets is much more difficult than the balanced datasets (Gulati, 2020).

Due to the privacy concerns, it is reported that the dataset does not contain much background information about the original features, instead, it consists of 28 features (*V1*, *V2*, … *V28*) digitized through PCA (Principal Component Analysis) transformation (Kaggle Machine Learning Group, 2020). PCA is a statistical technique used for size reduction and data digitization in large datasets and is highly effective in revealing the necessary information in the dataset. Thanks to this technique, it is possible to find the general features in the dataset and to reduce the size, as well as to convert the data into a numerical value that can be processed.

In the dataset, there are also *Time*, *Amount*, and *Class* features that are not converted with PCA. Therefore, the dataset contains 31 numerical features in total. (The *Time* and *Class* features are integers, while the remaining 29 features are floating-point numbers.) The *Time* feature refers to the time in seconds between the first transaction in the dataset and any transaction. The *Amount* feature represents the amount of the transaction. The *Class* feature of a transaction is a numeric value that indicates whether it is fraudulent. This feature takes a value of 1 in the case of fraud, and 0 otherwise.

2.2. Data preprocessing

The dataset needs to be preprocessed before it is fed as an input to the algorithms evaluated in this study. The preprocessing process usually consists of two basic steps: (1) completion of the missing data, (2) scaling of the values of the features. Since the dataset (*creditcard*) does not contain any missing data, no preprocessing has been done concerning (1). On the other hand, a scaling process was carried out for the dataset, details of which are given below.

The *Time* and *Amount* features in the dataset vary in size compared to the other features *(V1, V2, …, V28)*, as is evident from the example given in Table 1. To ensure that all values equally contribute to the success of the algorithms used in this study, these values need to be brought to the same standard. Otherwise, the high-valued features will outweigh the lower-valued ones. Therefore, these two features are scaled by feature standardization in step ② of the design flow. *StandardScaler* method, which is the standard scaler of Python's *sklearn.preprocessing* library, is used for this process.

Table 1. Sample features of a transaction in the dataset.

V1	V2	Time	Amount
0.694885	-1.36182	16	231.71

The working logic of the standard scaler is exemplified in Table 2 by using the *Time* and *Amount* features of the transaction given in Table 1.

Table 2. Standard scaling of sample features.

Step	Time	Amount
❶	948138.5	88349.6
❷	-948122.5	88117.9
❸	**-1.99625**	**0.57316**

In step ❶ of the scaling process, the mean values of the *Time* and *Amount* features in the dataset are calculated (for example, the total processing time contained in the *Time* feature is divided by the total number of transactions in the dataset and the average value of the feature is obtained as 27003650904/284807=948138.5). In step ❷, the transaction data given in Table 1 is subtracted from the mean values (for example, the value of 16 of the *Time* feature from Table 1 is subtracted from the average and 16-948138.5=-948122.5 is obtained). Finally, in step ❸, the standard deviation values of the features are divided by the values calculated in step ❷. (For example, the scaled value of the *Time* feature of the sample data is -948122.5/47488.1=-1.99625).

After the standard scaling, the value of 16, which is the sample transaction data for the *Time* feature, is scaled to -1.99625, and the value of 231.71, which is the sample transaction data for the *Amount* feature, is scaled to 0.57316. These scaled values are shown in row ❸ of Table 2.

In summary, in the data preprocessing step, the standard scaler is applied to the *Time* and *Amount* features, and the dataset containing a total of 31 features is made available by considering the *Time*, *Amount*, and *Class* features. (30 features are used in the models' training while 31 features, including the *Class* feature, are utilized in the testing phase of the models.)

2.3. Dividing the dataset

To effectively analyze the dataset in all of the machine learning, artificial neural network, and deep learning algorithms considered in this study, the dataset is divided into two subsets randomly. In step ③ of the design flow, 70% of the dataset is reserved for the models' training whereas 30% is allocated for the testing of the models. It is known that this ratio is often preferred in the literature (Awoyemi et al., 2017).

2.4. Implementation of the algorithms

Machine learning, artificial neural network, and deep learning are among the most popular research areas of our age (Şeker et al., 2017). These three concepts have different characteristics in the technology universe, but they also intersect at many points. In step ④ of the design flow, seven algorithms from these three research areas are implemented. For this purpose, in this step, three machine learning algorithms (*K-Nearest Neighbor*, *Naive Bayes and Support Vector Machine*), two neural network algorithms (*Binary Classifier*, *Autoencoder*), and two deep learning algorithms (*Deep Autoencoder and Deep Neural Network Classifier*) are taken into consideration to effectively analyze the fraudulent credit card transactions.

2.4.1. Machine learning

Machine learning is a concept that refers to the process of making inferences on an existing dataset by using mathematical operations and making predictions using these inferences (Gulati, 2020). What makes this concept privileged is that it can generate predictions and make decisions against contingencies by evaluating the existing data rather than adhering to static instructions. The most basic steps in machine learning are feature extraction, modeling, and evaluation. In the feature extraction step, unnecessary data is cleared to highlight the features that best represent the model. In the model creation step, the learning process is carried out by using the models best suited to the dataset from different models. The learning models used in machine learning are divided into two groups: supervised and unsupervised. In supervised

learning, a dataset containing inputs and outputs is given as an input to the algorithm and the algorithm finds a method that determines how to access these inputs and outputs. In unsupervised learning, since there is no output related to the existing inputs (classes are unknown beforehand), the algorithm analyzes the existing data and determines the relationships itself.

In this study, three machine learning algorithms (*K-Nearest Neighbor, Naive Bayes, and Support Vector Machine*) are evaluated to analyze the effectiveness of different machine learning approaches on the same problem. These algorithms are preferred because they are easy to implement and can produce good results in terms of performance. Moreover, K-Nearest Neighbor is very competitive compared to sophisticated machine learning algorithms (Liu and Zhang, (2012)), Naive Bayes is very useful in large datasets (Mukhandi, (2018)), and the Support Vector Machine has a low computational complexity (Gitonga, 2018).

2.4.1.1. K-nearest neighbor

The classification process with K-Nearest Neighbor (KNN) is done according to the proximity relations between objects. The KNN algorithm, that has the advantage of the ease of development, also has several disadvantages: it needs a high amount of memory space, the transaction load and cost increase significantly as the dataset size increases, the performance depends heavily on the parameters such as the number of neighbors (K) (Liu and Zhang, 2012).

In the classification process with KNN, the number of elements is determined by looking at the value K, which represents the number of neighbors. When the algorithm encounters a new data, it calculates the distances to K and adds the new value to the set of the closest neighbors based on the smallest distance after sorting. Euclidean distance is often preferred in distance calculation. In this process, when the value of K is 1, only the class with the nearest neighbor is assigned, and when the value of K is closer to the number of instances, all the data in the dataset is considered. The algorithm contains a training data and this process repeats for each new value. Therefore, it is very important for KNN to have a large training set and choose the appropriate K value. In this study, the neighborhood value (K) best suited to the relevant dataset is obtained first. For this purpose, training data is utilized and the best neighborhood value for the related dataset is determined as 3. Besides, Euclidean distance is used in distance calculation. Finally, the *KNeighborsClassifier* method of the *sklearn.neighbors* library of Python is used and a model is created by giving the value of K determined earlier as an input to the *KNeighborsClassifier* method.

2.4.1.2. Naive bayes

Naive Bayes (NB) is a probabilistic machine learning algorithm used for classification. The NB algorithm handles values independently in the classification process. The algorithm calculates the probability of each state for a value and classifies it according to the highest probability value. If a data in the test set does not have a counterpart in the training set, then 0 (zero) is assigned as the probability value for that data and no prediction can be made. This is known as Zero Frequency in the literature (Wu, 2013). In this study, a model was created for the NB classification algorithm by using the *GaussianNB* method of Python's *sklearn.naive_bayes* library.

2.4.1.3. Support vector machine

Support Vector Machine (SVM) is a highly effective and simple learning theory-based algorithm used for classification purposes. It is employed to optimally separate the data of two classes from each other. For this, it is necessary to draw a border between two groups in a plane. The algorithm determines how this boundary is drawn. SVM transforms the classification problem into a quadratic optimization problem and solves it. With this transformation, the number of operations decreases in the learning phase and a faster solution is achieved compared to other algorithms (Osowski, 2004). On the other hand, the algorithm is often preferred in determining the classes that can be parsed linearly by kernel functions. In this study, the linear SVM algorithm was utilized and a model was created for the SVM classification algorithm by using the *SVC* method of Python's *sklearn.svm* library.

2.4.2. Artificial neural network

Artificial Neural Network (ANN) is a technique that consists of interconnected nodes similar to the human brain and makes use of these nodes when sending data from the input layer to the output layer (Kaynar, 2017). ANN is suitable for use in almost every field in which machine learning algorithms are applied and has basically a structure consisting of only input and output layers. However, since this structure fails to solve nonlinear problems, a hidden layer is added between the two layers (Arı and Berberler, 2017). Thus, the ANN consists of three layers, the input layer, the hidden layer, and the output layer, as shown in Figure 2, and a certain number of neurons in each layer. The input layer contains as many neurons as the number of features in the dataset, and the output layer contains as many neurons as the number of classes in the dataset. The number of hidden layers in the model and the number of neurons in these layers are not constant; in other words, they differ according to the problem.

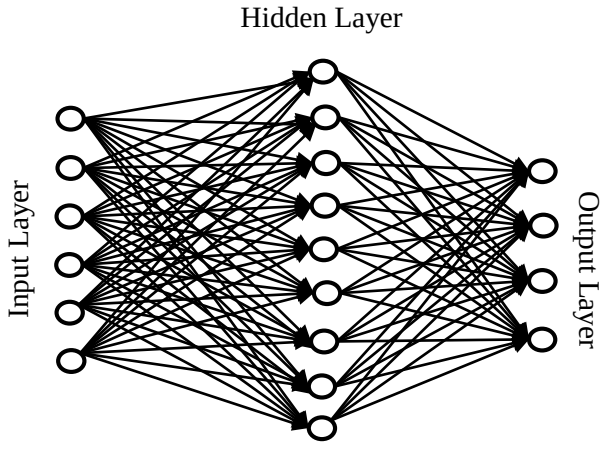

Figure 2. Artificial neural network (Nielsen, 2015).

ANN, consisting of more than one hidden layer, is called deep artificial neural network as shown in Figure 3. The increase in the number of hidden layers that deepen ANN also allows the modeling of highly complex data.

In ANN, learning is accomplished by using the connections belonging to the neurons in the network. Since the connections are active throughout the network, the information is also spread all over the network. For this reason, the optimal values of the weights of all nodes are taken

into consideration to access the information, instead of using the weight of a single node. To reach these weights, the entire network needs to be trained (Soylu, 2018).

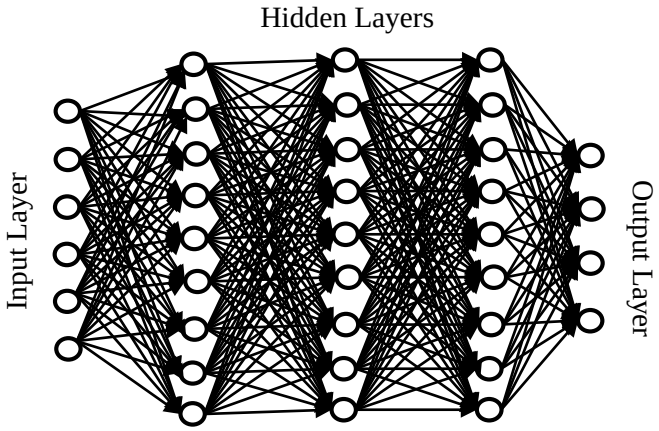

Figure 3. Deep artificial neural network (Nielsen, 2015).

In ANN, it is aimed to minimize the loss by optimizing the parameters (weights) of the neurons. Each neural network produces an output and a loss is calculated by comparing this output with the actual output. Loss functions are utilized for this operation in ANN. To minimize this loss, optimization algorithms are employed to optimize the weights of the network's neurons. In this way, the neural network is trained.

In this study, two different ANN algorithms (*Binary Classifier, Autoencoder*) were evaluated to examine their effects on the problem at hand. Binary Classifier has been preferred due to its ease of implementation and applicability, and Autoencoder has been chosen because of its ability to detect malware (Yousefi-Azar, 2017).

2.4.2.1. Binary classifier

Binary Classifier (BC) is a classification method that produces one of the two outputs for input data. With this algorithm, classification is divided into two categories. These are the categories that are mutually exclusive and have only two possible responses. It is not guaranteed that the model will always work perfectly and make accurate predictions.

In this study, the parameters listed in the second column of Table 3 were used in the model created for BC. In this model, which consists of only the input and output layers, activation functions are utilized to decide whether the neurons in the layers will be active or not. For this purpose, two different activation functions were employed in the input and output layers. The activation function used in the input layer produces 0 for negative inputs and the input itself for positive inputs. In contrast, the activation function in the output layer produces a probabilistic output in the range of 0 and 1. Besides, a loss function and an optimization algorithm that are frequently utilized during the compilation of models in the literature were also used. Finally, the model was created through the use of the method from the related Python library.

2.4.2.2. Autoencoder

Autoencoder (AE) is an artificial neural network that copies data from the input layer to the output layer without generating a code to represent the input and is used for unsupervised learning. This neural network consists of two different parts: the encoder and decoder. The

encoder is located between the input layer and the hidden layer, and creates code from the input parameter, reducing the multi-dimensional data to a small size. On the other hand, the decoder is stationed between the hidden layer and the output layer, and converts the compressed data into output using the code from the encoder.

In this study, the parameters listed in the third column of Table 3 were used in the model created for AE. AE employed a single encoder and decoder, and the neurons accessed through the input layer were transferred to the encoder. The code generated by the encoder was then converted with the decoder and routed to the output layer. In fact, the conversion process here meant that the input was copied exactly to the output. This process, in which the generated output is equal to the input, is called as reconstruction in the literature (An and Cho, 2015).

2.4.3. Deep learning

Deep learning is categorized as a sub-branch of machine learning and defined as a machine learning method that uses multiple layers to make predictions by making inferences on existing data (Şeker et al., 2017). Deep learning takes an input parameter as in machine learning, but unlike machine learning, it makes assessments by inferring features directly with the parameters it has discovered.

The theoretical foundations of deep learning are based on the classical artificial neural network literature. However, unlike the traditional use of classical neural networks, in deep learning, many hidden neurons and layers are considered as an architectural advantage along with the new training paradigms (Ravi, 2016). On the other hand, the deep learning algorithms whose inspiration is the human brain need to be fed with a lot of data to make data-related decisions. It is not a problem that these data fed through neural networks are very diverse and unstructured. On the contrary, such data can be used by deep learning algorithms and it is possible to solve complex problems easily. The success of deep learning techniques depends on the ability of the underlying hardware platform to perform fast and supervised training of complex networks using large amounts of data (Gupta, 2015). In other words, the faster and more the algorithm learns, the higher the performance.

In this study, two different deep learning algorithms (*Deep Autoencoder*, *Deep Neural Network Classifier*) were evaluated. The Deep Autoencoder was chosen since it is a deep learning algorithm and has ties to the artificial neural network, while the Deep Neural Network Classifier was preferred because of its applicability to different areas, its ability to overcome computing difficulties in large-scale data and its computational time performance (Gitonga, 2018).

2.4.3.1. Deep autoencoder

The increase in the amount of data in automatic encoders decreases the performance of the encoder. It is possible to avoid this situation with the feed-forward Deep Autoencoder (DAE) consisting of multiple hidden layers. DAE used in deep learning is a special form of artificial neural network. Numerous hidden layers used in DAE enable modeling of complex data. On the other hand, overfit of the network poses a major problem for deep autoencoders. Excessive training leads the model to memorize and prevents DAE from being successful in each model. The method of regularization has been proposed in the literature to overcome this problem (Liang and Liu, 2015).

In this study, the parameters listed in the fourth column of Table 3 were used in the model created for DAE. (A DAE model were obtained by tripling the number of hidden layers in AE.) The number of inputs (the number of neurons) in each layer is equal to half of the number of inputs in the previous layer. Unlike AE, non-linear *tanh* activation function producing output in the range of [-1,1] (wider range) was used for a fast learning and classification process. Besides, the *L1* norm, which is expressed as weight regulation in the input layer, was used to reduce the over-training of the network and increase the performance of the model on the new data it may encounter. For this purpose, the resulting parameter obtained by multiplying the weights by a value in line with the *L1* norm was added to the error value that the model will generate. Thus, the precise changes causing over-learning were stretched and the memorization was prevented.

2.4.3.2. Deep neural network classifier

It is difficult to optimize deep autoencoders that have multiple encoders and decoders (i.e. deep autoencoders with multiple hidden layers.) Besides, initiation of any deep neural network with very large weights leads to a weak local minimum, while initiation with very small weights causes small inclines that slow the learning process (Shin, 2016). To prevent this situation, a Deep Neural Network Classifier (DNN) with more layers than a conventional artificial neural network is used. Thanks to the numerous hidden layers it contains, DNN facilitates the training of the network through highly complex operations. On the other hand, regularization techniques used to prevent over-training of the network can reduce the complexity of the model without reducing the performance.

One of the deep learning methods evaluated in this study is DNN. In the model created for DNN, the parameters listed in the fifth column of Table 3 were used. Since the model has a very complex structure, it is crucial to reduce the size by maintaining the performance ratio where resources such as processor power and memory are limited. Therefore, the proposed study employed a *Dropout* layer, an improvement technique that is frequently preferred in deep learning methods. With this layer, specific neurons are removed from the hidden layer using random neurons or a threshold value. In this study, a threshold value of 0.25 was used and the number of neurons in the hidden layer applied to *Dropout* was reduced to one-fourth in the next layer.

Table 3. Parameters used in ANN and deep learning algorithms.

Parameter	**BC**	**AE**	**DAE**	**DNN**
Number of Neurons (Input Layer)	30	30	30	16
Number of Neurons (Output Layer)	1	30	30	1
Activation Function (Input Layer)	relu	relu	tanh	relu
Activation Function (Output Layer)	sigmoid	sigmoid	relu	sigmoid
Loss Function	binary crossentropy			
Optimization Algorithm	Adam			
Python Library	keras.wrappers.scikit_learn	tensorflow.keras.models	tensorflow.keras.models	tensorflow.keras.models
Method	KerasClassifier	Model	Model	Sequential

2.5. Model performance criteria

In the evaluation process (⑤), a set of model performance criteria, that are frequently used in the literature, were utilized to analyze the performance of the methods under investigation (Varol and İşeri, 2019; Kaynar, 2016). The parameters used in the formulation of these criteria are defined as follows:

TP (True Positive): It is the number of transactions that are fraudulent and are classified as fraudulent by the related model.

TN (True Negative): It is the number of transactions that are legitimate and are classified as legitimate by the related model.

FP (False Positive): It is the number of transactions that are legitimate, but are classified as fraudulent by the related model.

FN (False Negative): It is the number of transactions that are fraudulent, but are classified as legitimate by the related model.

TPR (True Positive Rate): It is the ratio of TP to TP+FN.

FPR (False Positive Rate): It is the ratio of FP to FP+TN.

The definitions and formulations of the model performance criteria expressed by the above parameters are given below:

Accuracy: It is the ratio of the correct predictions to all predictions. The accuracy criterion is calculated using the formula given with (2.5.1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(2.5.1)}$$

Precision: It is the ratio of the number of true positive predictions to the sum of the numbers of true and false positive predictions. The precision criterion is calculated using the formula given with (2.5.2).

$$Precision = \frac{TP}{TP+FP} \quad \textbf{(2.5.2)}$$

Recall: It is the ratio of the number of true positive predictions to the sum of the numbers of true positive and false negative predictions. The recall criterion is calculated using the formula given with (2.5.3).

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(2.5.3)}$$

F-score: It is the harmonic mean of precision and recall. The F-score criterion is calculated using the formula given with (2.5.4).

$$F - score = 2 \times \frac{precision \times recall}{precision + recall} \quad \textbf{(2.5.4)}$$

ROC Curve and AUROC Score: The curve drawn using different threshold values of FPR on the x-axis and TPR on the y-axis is called ROC (Receiver Operating Characteristic) curve, and the area under this curve (AU, Area Under) is called AUROC score. It is reported that the AUROC score is an essential measure of success in comparing the classification performance of the algorithms applied to unstable datasets such as *creditcard* (Soylu, 2018).

PR Curve and AUPRC Score: The curve drawn using different threshold values of recall on the x-axis and precision on the y-axis is called PRC (Precision-Recall Curve), and the area under this curve is called AUPRC score. It is recommended by the creators of the dataset that the AUPRC score is an essential metric for comparing the classification performance of the algorithms that use *creditcard* due to the class imbalance ratio (Kaggle Machine Learning Group, 2020).

2.6. Test environment

The algorithms evaluated in this study were implemented on a laptop computer with the configurations listed in the Table 4.

Table 4. Configurations of the test environment.

Component	Feature
Processor	Intel® Core™ i7-4870HQ Processor, 6 MB Cache, 2.5 GHz (Launch Date: Q3'14)
Memory	16 GB, 1600 MHz DDR3
Operating System	Partition 1: macOS High Sierra Partition 2: Windows 10+Ubuntu 18.04 (WSL, Windows Subsystem for Linux)

3. Experimental Results and Discussion of Findings

It is known that the variety of methods used in scientific research is important for exploring the solution space. Therefore, in addition to machine learning methods, popular artificial neural network and deep learning methods, commonly used as sub-branches of machine learning, were used to identify fraudulent credit card transactions. For this purpose, K-Nearest Neighbor, Naive Bayes, Support Vector Machine, Binary Classifier, Autoencoder, Deep Autoencoder, and Deep Neural Network Classifier algorithms were implemented using Python programming language and *scikit-learn* library in *Anaconda* environment and their performances were tested with a dataset frequently employed in the literature.

In the first stage of this study, called data preprocessing, the data that varies considerably in size in the relevant dataset was scaled by feature standardization. Later, a 70-30% train-test split was randomly performed on the dataset containing 31 features. Since the process of dividing the dataset is essentially random, different results can be obtained each time the algorithms are executed due to differences in both training and test data.

Therefore, each algorithm was run 10 times to analyze the effectiveness of the related algorithms more accurately, and the best, worst, and average results were noted. Besides, six different model performance criteria were used to evaluate the performance of each method.

When drawing ROC and PR curves for all classifiers, the *predict* function of the *scikit-learn* library was used and the default threshold value of 0.5 was utilized. Although tests were conducted to find the optimal value of the threshold (the threshold value that maximizes the accuracy) by means of the *predict* function, no significant difference was obtained compared to the default value of 0.5.

The experimental results obtained in this study are discussed in five subsections of this section. The performances of the algorithms are analyzed through:

- traditional performance criteria (accuracy, precision, recall, and f-score) in the first subsection,
- ROC curve and AUROC score in the second subsection,
- PR curve and AUPRC score in the third subsection and
- working (execution) times in the fourth subsection.

In the fifth subsection, the experimental results are compared with the results of similar studies in the literature.

3.1. Performance analysis with traditional performance criteria

Table 5 contains the experimental results for each of the seven different algorithms considered in this study. In this table, columns and rows are reserved for the algorithms and the model performance criteria, respectively. In this subsection, an analysis is solely made through the traditional performance criteria. According to the first four rows of Table 5:

- The best results were achieved using DNN, KNN, BC, and SVM methods with a ratio of 99.9% in terms of the accuracy criterion. On the other hand, NB performed very close to these methods with an accuracy rate of 97.8%, while DAE and AE showed relatively lower performance compared to other approaches.
- In terms of the precision criterion, the best result was achieved with KNN (98.4%), and the worst result was obtained with DAE and AE (49.9%).
- When the success criterion is the recall, DNN (91%) produced the best result while the worst performance was achieved by AE (38.3%).
- A comparison made by taking the F-score into account showed that KNN breasted the tape with a ratio of 93.2% and the worst performance was obtained with AE (38.3%).

Table 5. The results of the algorithms based on six performance criteria

Performance Criterion		DNN	KNN	BC	SVM	NB	DAE	AE
Accuracy	*worst*						0.659	0.703
	best	0.999	0.999	0.999	0.999	0.978	0.776	0.715
	average						0.749	0.708
Precision	*worst*	0.901	0.964	0.910	0.928		0.333	0.488
	best	0.940	0.984	0.938	0.946	0.531	0.499	0.499
	average	0.920	0.966	0.928	0.932		0.484	0.490
Recall	*worst*	0.864	0.856	0.881	0.888	0.903	0.307	0.365
	best	0.910	0.897	0.907	0.897	0.905	0.500	0.383
	average	0.885	0.873	0.887	0.892	0.905	0.464	0.378
F-score	*worst*	0.884	0.903	0.895	0.880	0.550	0.355	0.316
	best	0.919	0.932	0.922	0.920	0.552	0.500	0.383
	average	0.902	0.913	0.906	0.900	0.551	0.449	0.365
AUROC	*worst*	0.972	0.856	0.881	0.864	0.888	0.499	0.240
	best	0.999	0.900	0.910	0.910	0.910	0.720	0.370
	average	0.996	0.864	0.887	0.873	0.894	0.684	0.264
AUPRC	*worst*	0.724	0.821	0.791	0.765	0.320	0.152	0.005
	best	0.820	0.880	0.850	0.840	0.450	0.500	0.020
	average	0.778	0.841	0.816	0.804	0.388	0.464	0.120

3.2. Performance analysis with ROC curve and AUROC score

Classification in unbalanced datasets is very difficult and it is not advised to compare the performances of the methods solely with the accuracy criterion (Soylu, 2018). This is because algorithms running on such datasets often tend towards the class holding the majority. Therefore, the ROC curve is of great importance in the classification process with unbalanced data sets (Soylu, 2018). Consequently, to accurately evaluate the success of the methods used in this study, the commonly preferred ROC curve and AUROC score were also utilized. The ROC curve visualizes the balance between FPR and TPR, while the AUROC score numerically expresses the corresponding performance ratio. The closer the ROC curve gets to the upper left corner of the graph, the higher the related AUROC score of the test.

The ROC curve of each of the seven algorithms is shown in Figure 4 whereas the corresponding AUROC scores are listed in the fifth row of Table 5. When Figure 4 and Table 5 are examined together, it is seen that the ROC curve of the DNN classification is closest to the top left corner of the graph, and not surprisingly, DNN has the best AUROC score among all methods with a success rate of 99.9%. This finding points out that deep learning is more effective than machine learning in the analysis of fraudulent credit card transactions in terms of the AUROC criterion. Besides, on a comparison between DNN and ANN methods, the AUROC score performance of DNN was better than DAE (72%) and AE (37%). This advantage is directly due to the excess number of hidden layers contained in AE and DAE, and indirectly due to the complexity of the encoder and decoder that occurs in parallel to the excess number of layers. This finding makes it clear that artificial neural networks need to be deepened for this particular problem. Finally, when it comes to the evaluation of the machine learning models, the results highlight that NB and SVM showed better ROC curve performance than KNN.

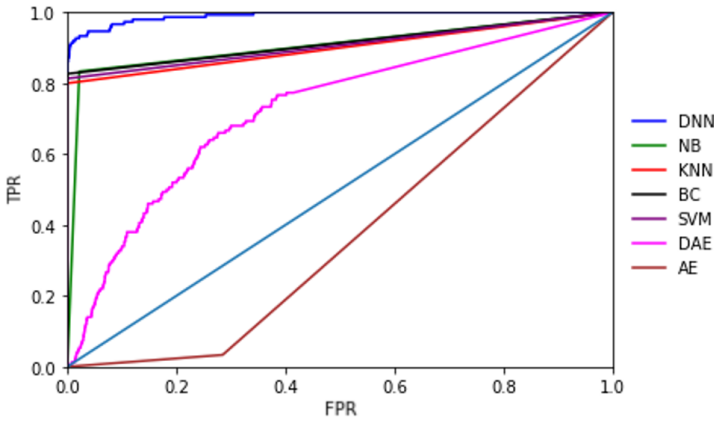

Figure 4. ROC curves of the algorithms.

3.3. Performance analysis with PR curve and AUPRC score

The research group that developed the *creditcard* dataset recommends the use of AUPRC score to analyze the performance of the algorithms with their dataset that is classified as unstable (Kaggle Machine Learning Group, 2020). Therefore, the PR curve and AUPRC score were also used to evaluate the success rates of the methods used in this study. (To the best of our knowledge, this is the first study in the literature that reports the AUPRC scores along with the

PR curves on this dataset.)The PR curve visualizes the balance between recall and precision, while the AUPRC score numerically expresses the corresponding performance ratio. The closer the PR curve gets to the upper right corner of the graph, the higher the related AUPRC score of the test.

The PR curve of each of the seven algorithms is shown in Figure 5 while the sixth row of Table 5 lists the corresponding AUPRC scores. It is seen from Figure 5 and Table 5 that:

- The PR curve of KNN classification is closest to the top right corner of the graph, in other words, KNN has the best AUPRC score among all methods with 88% success rate.
- The algorithm that produces the worst result is AE (%2).
- The AUPRC score performance of DNN (82%) is better than DAE (50%) and AE (2%).
- NB has the worst AUPRC score performance in machine learning algorithms.

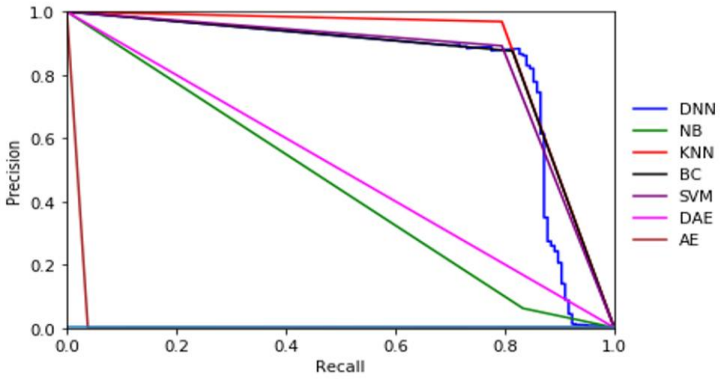

Figure 5. PR curves of the algorithms.

3.4. Collective performance analysis with all criteria

A collective analysis that takes Figure 4, Figure 5, and Table 5 into consideration reveals the following findings:

- The performance of all three algorithms in the machine learning category was quite well in terms of accuracy (more than 90%) and the AUROC score (around 90%). In the artificial neural network category, the performance of BC was much better than that of AE in all criteria. More importantly, the results demonstrated that AE showed the worst performance among all seven algorithms.
- In all performance criteria, DNN outstripped DAE in the deep learning category. Besides, DNN was the best algorithm among all with the AUROC score of 99.9%.

3.5. Performance analysis with working times

Figure 6 includes the training times of each of the seven algorithms whose performances were analyzed in this study. According to this figure, the fastest and slowest algorithms that complete the training process in the analysis of fraudulent credit card transactions are NB and SVM, respectively. It seems that the large size of the dataset has affected the training process of the SVM algorithm quite negatively compared to other algorithms. On the other hand, the DNN algorithm whose AUROC performance was the best among all finished the training process in a fairly reasonable time, while KNN, the algorithm with the highest AUPRC score, needed almost one-third of SVM time to train its model.

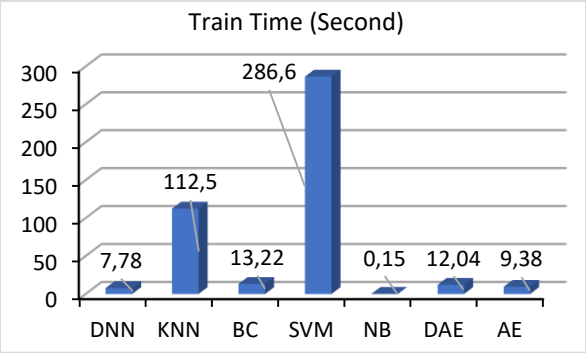

Figure 6. Training times of the algorithms.

The test times of the algorithms are shown in Figure 7. From this figure, it can be concluded that the testing times follow a trend parallel to the training times. In other words, when it comes to testing, SVM and KNN are the slowest algorithms, whereas NB is the fastest one.

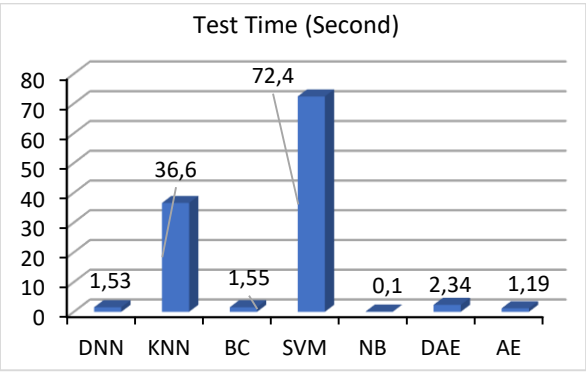

Figure 7. Test times of the algorithms.

3.6. Performance comparison with similar studies in the literature

In this subsection, DNN, the best algorithm of this study in terms of accuracy and AUROC score, is compared with seven different studies in the literature that use the same dataset.

In the first study (Lakshmi and Kavilla, (2018)), three algorithms namely Logistic Regression, Decision Tree, and Random Forest were implemented. Table 6 shows that our DNN model was more successful in terms of accuracy criterion compared to these three algorithms. (In (Lakshmi and Kavilla, (2018)), AUROC was not considered as a performance measure.)

Random Forest and Random Forest with Grid Search methods from the second study (Meker, 2018) produced the same result as DNN in terms of AUROC score. On the other hand, the remaining six methods lagged behind the performance of DNN. (In (Meker, (2018)), accuracy was not considered as a performance measure.)

DNN performance could not be approached in terms of both accuracy and AUROC score in all of the algorithms implemented in the third (Mukhandi, (2018)), fourth (Awoyemi et al. (2017)), and fifth (Itoo and Singh, (2020)) studies.

Three different algorithms were implemented within the scope of the sixth (Gitonga, (2018)) and seventh (Soylu, (2018)) studies, and DNN was of them. The performances of these DNNs

was similar to our DNN implementation in terms of accuracy criterion. On the other hand, the performance of our DNN implementation is decisively ahead when AUROC criterion is considered. It is assumed that this difference may be due to the feature scaling applied in the preprocessing step, and the number of hidden layers, the regularization technique, the number of neurons and the activation function utilized.

Table 6. Comparison of our DNN implementation and other related studies in the literature.

Algorithm	**Accuracy**	**AUROC**
Logistic Regression	0.900	
Decision Tree	0.943	-
Random Forest	0.955	
(Lakshmi and Kavilla, (2018))		
Logistic Regression		0.933
Logistic Regression with Grid Search		0.967
Random Forest		0.999
Random Forest with Grid Search		0.999
Naive Bayes	-	0.919
Decision Tree		0.960
Decision Tree with Grid Search		0.978
Support Vector Machine Classifier		0.920
(Meker, 2018)		
AdaboostM1	0.940	0.974
Logistic Regression	0.932	0.974
Random Forest	0.948	0.979
KNN	0.928	0.950
(Mukhandi, (2018))		
Naive Bayes	0.976	0.970
K-Nearest Neighbor	0.979	0.968
Logistic Regression	0.548	0.557
(Ravi, 2016)		
Logistic Regression	0.959	0.918
Naive Bayes	0.915	0.829
K-Nearest Neighbor	0.751	0.633
(Itoo and Singh, (2020))		
Feed Forward Neural Network	0.975	0.944
Deep Neural Networks	0.999	0.904
Support Vector Machine	-	0.989
(Gitonga, 2018)		
Deep Neural Networks		0.963
Random Forest	0.999	0.956
Classifier Stack		0.979
(Soylu, 2018)		
DNN Implementation of This Study	**0.999**	**0.999**

4. Conclusion and Future Work

Within the scope of this study, techniques based on machine learning, artificial neural network and deep learning were evaluated to develop an effective solution to the customer victimization experienced by e-commerce companies due to fraudulent credit card transactions. For this purpose, three machine learning algorithms (*K-Nearest Neighbor, Naive Bayes and Support Vector Machine*), two artificial neural network algorithms (*Binary Classifier, Autoencoder*), and two deep learning algorithms (*Deep Autoencoder and Deep Neural Network Classifier*) were implemented. The effectiveness of these algorithms was tested with a famous dataset called *creditcard*, which is widely used in the literature. Experimental results showed that the Deep Neural Network Classifier outperformed the other algorithms used in this study and the best study ever reported in the literature in terms of accuracy and AUROC performance criteria. More importantly, DNN was able to achieve this performance advantage in a very reasonable time, both during the training and testing of the model.

In future studies, it is planned to consider other datasets used in the related studies in the literature to detect credit card fraud.

Ethics in Publishing

There are no ethical issues regarding the publication of this study.

References

An, J, Cho, S. 2015. "Variational autoencoder based anomaly detection using reconstruction probability", *Special Lecture on IE;* 2(1): 1-18.

Arı, A, Berberler, M, E. 2017. "Yapay sinir ağları ile tahmin ve sınıflandırma problemlerinin çözümü için arayüz tasarımı", *Acta Infologica;* 1(2): 55-73.

Awoyemi, J, O, Adetunmbi, A, O, Oluwadare, S, A. "Credit card fraud detection using machine learning techniques: A comparative analysis", *In 2017 International Conference on Computing Networking and Informatics (ICCNI),* Lagos, Nigeria, 2017, 1-9.

Aydoğdu, Ü, R, Karamustafaoğlu, O, Bülbül, M, Ş. 2017. "Akademik Araştırmalarda Araştırma Yöntemleri ile Örneklem İlişkisi: Doğrulayıcı Doküman Analizi Örneği", *Dicle University Journal of Ziya Gokalp Education Faculty*; (30): 556-565.

Carcillo, F, Le Borgne, Y, A, Caelen, O, Bontempi, G. 2018a. "Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization", *International Journal of Data Science and Analytics;* 5(4): 285-300.

Carcillo, F, Dal Pozzolo, A, Le Borgne, Y, A, Caelen, O, Mazzer, Y., Bontempi, G. 2018b. "Scarff: a scalable framework for streaming credit card fraud detection with spark", *Information fusion;* 41: 182-194.

Carcillo, F, Le Borgne, Y, A, Caelen, O, Kessaci, Y, Oblé, F, Bontempi, G. 2019. "Combining unsupervised and supervised learning in credit card fraud detection", *Information sciences*.

Dal Pozzolo, A, Boracchi, G, Caelen, O, Alippi, C, Bontempi, G. 2017. "Credit card fraud detection: a realistic modeling and a novel learning strategy", *IEEE transactions on neural networks and learning systems;* 29(8): 3784-3797.

Dal Pozzolo, A, Caelen, O, Johnson, R, A, Bontempi, G. "Calibrating probability with undersampling for unbalanced classification", *In 2015 IEEE Symposium Series on Computational Intelligence*, Cape Town, South Africa, 2015, pp 159-166.

Dal Pozzolo, A, Caelen, O, Le Borgne, Y, A., Waterschoot, S, Bontempi, G. 2014. "Learned lessons in credit card fraud detection from a practitioner perspective", *Expert systems with applications*; 41(10): 4915-4928.

Gitonga, J, T. 2018. "Fraud detection using machine leaning: a comparative analysis of neural networks & support vector machines", Doctoral dissertation, *Strathmore University*.

Gulati, P. 2020. "Hybrid Resampling Technique to Tackle the Imbalanced Classification Problem".

Gupta, S, Agrawal, A, Gopalakrishnan, K, Narayanan, P. "Deep learning with limited numerical precision", *In International Conference on Machine Learning*, Lille, France, 2015, pp 1737-1746.

Han, H, Lim, S, Suh, K, Park, S, Cho, S, J, Park, M. "Enhanced Android Malware Detection: An SVM-Based Machine Learning Approach", *In 2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Busan, Korea (South), 2020, pp 75-81.

Itoo, F, Singh, S. 2020. "Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection", *International Journal of Information Technology*, 1-9.

Kaggle Machine Learning Group. "Credit Card Fraud Detection Dataset", https://www.kaggle.com/mlg-ulb/creditcardfraud (accessed at 16.06.2020).

Kaynar, O, Aydın, Z, Görmez, Y. 2017. "Sentiment analizinde öznitelik düşürme yöntemlerinin oto kodlayıcılı derin öğrenme makinaları ile karşılaştırılması", *Bilişim Teknolojileri Dergisi*; 10(3): 319-326.

Kaynar, O, Yıldız, M, Görmez, Y, Albayrak, A. "Makine öğrenmesi yöntemleri ile Duygu Analizi", *In International Artificial Intelligence and Data Processing Symposium (IDAP'16)*, Malatya, Turkey, 2016, pp 17-18.

Kolter, J, Z, Maloof, M, A. 2006. "Learning to detect and classify malicious executables in the wild", *Journal of Machine Learning Research*; 7: 2721-2744.

Lakshmi, S, V, S, S, Kavilla, S, D. 2018. "Machine learning for credit card fraud detection system", *Int. J. Appl. Eng. Res.*; 13(24): 16819-16824.

Lebichot, B, Le Borgne, Y, A, He-Guelton, L, Oblé, F, Bontempi, G. "Deep-learning domain adaptation techniques for credit cards fraud detection", *In INNS Big Data and Deep Learning conference*, Genova, Italy, 2019, pp 78-88.

Li, J, Sun, L, Yan, Q, Li, Z, Srisa-an, W, Ye, H. 2018. "Significant permission identification for machine-learning-based android malware detection", *IEEE Transactions on Industrial Informatics;* 14(7): 3216-3225.

Liang, J, Liu, R. "Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network", *In 2015 8th International Congress on Image and Signal Processing (CISP),* Shenyang, China, 2015, pp 697-701.

Liu, H, Zhang, S. 2012. "Noisy data elimination using mutual k-nearest neighbor for classification mining", *Journal of Systems and Software;* 85(5): 1067-1074.

Lopez, C, C, U, Cadavid, A, N. "Machine learning classifiers for android malware analysis", *In 2016 IEEE Colombian Conference on Communications and Computing (COLCOM),* Cartagena, Colombia, 2016, pp 1-6.

Meker, T. 2018. "Credit card fraud detection analysis and machine learning application", *MEF Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, Türkiye.

Mukhandi, H. 2018. "Developing machine learning methods for network anomaly detection", Master of Science Thesis, 2-3.

Nielsen, M, A. "Neural networks and deep learning", *Determination Press,* San Francisco, 2015.

Osowski, S, Siwek, K, Markiewicz, T. "Mlp and svm networks-a comparative study", *In Proceedings of the 6th Nordic Signal Processing Symposium,* Piscataway, NJ, USA, 2004, pp 37-40.

Ravì, D, Wong, C, Deligianni, F, Berthelot, M, Andreu-Perez, J, Lo, B, Yang, G, Z. 2016. "Deep learning for health informatics", *IEEE journal of biomedical and health informatics;* 21(1): 4-21.

Schultz, M, G, Eskin, E, Zadok, F, Stolfo, S J. "Data mining methods for detection of new malicious executables", *In Proceedings 2001 IEEE Symposium on Security and Privac,* Washington, DC, USA, 2001, pp 38-49.

Shin, H, C, Orton, M, Collins, D, J, Doran, S, Leach, M, O. "Organ detection using deep learning", *In: Kevin Zhou S (ed) Medical image recognition, segmentation and parsing, 1rd edn,* FL, United States. 2016, pp 123-153, Academic Press.

Soylu, K. 2018. "Kredi kartı sahte işlem tespiti", *Ankara Üniversitesi Fen Bilimleri Enstitüsü,* Ankara, Türkiye, Yüksek Lisans Tezi, 1-57.

Şeker, A, Diri, B, Balık, H, H. 2017. "Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme", *Gazi Mühendislik Bilimleri Dergisi*; 3(3): 47-64.

Yousefi-Azar, M, Varadharajan, V, Hamey, L, Tupakula, U. "Autoencoder-based feature learning for cyber security applications", *In 2017 International joint conference on neural networks (IJCNN),* Anchorage, AK, USA, 2017, pp. 3854-3861.

Wang, X, Yang, Y, Zeng, Y, Tang, C, Shi, J, Xu, K. "A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection", *In Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, Paris, France, 2015, pp 15-22.

Wu, J, Cai, Z, Zhu, X. "Self-adaptive probability estimation for naive bayes classification", *In The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, 2013, pp 1-8.

Varol, A, B, İşeri, İ. 2019. "Lenf Kanserine İlişkin Patoloji Görüntülerinin Makine Öğrenimi Yöntemleri ile Sınıflandırılması", *Avrupa Bilim ve Teknoloji Dergisi;* 404-410.