Credit Card Fraud Detection using Deep Learning Techniques

Oona VOICAN
Ministry of Transport and Infrastructure, Bucharest, Romania
oona.voican@yahoo.com

The objective of this paper is to identify credit card fraud and this topic can be solved with the help of advanced machine learning and deep learning techniques. Due to the fact that credit card fraud is a serious worldwide problem, we have chosen to create a model for detecting imposter scams by using deep neural networks. The purpose is to understand, determine and learn the normal behavior of the user and more precisely the detection of identity fraud. Each person has a trading pattern, uses certain operating systems, has a specific time to complete the transaction and spends large amounts of money usually within a certain time range. Transactions made by a certain user have a certain pattern, which can be identified with the help of neural networks. Machine learning involves teaching computers to recognize patterns in data in the same way as our brains do. Deep learning is just a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain. Deep learning at the core is the ability to form higher and higher level of abstractions of representations in data and raw patterns. The data used to train the model is real, and it will be processed using the one-hot encoding method, so that categorical data/variables can be used by the machine learning algorithm.

Keywords: *Artificial Intelligence, Credit Card Fraud, Deep Learning, Neural Network Model, User Behavior*
DOI: 10.24818/issn14531305/25.1.2021.06

1 Introduction

Machine learning has its foundations between the 1950s and 1960s, when researchers built a series of perceptrons, also called "artificial brains" at that time. In the late 1950s, Frank Rosenblatt suggested a very simplified mathematical model, called the perceptron, which mimics the recognition process of the human brain [1]. They were used to explore the incomprehensible problem of how the human brain manages to learn. These perceptrons were trained and instructed to detect differences between the faces of women and men. A human being can easily place a person in one of the two categories. However, at that time, a computer needed a series of complex rules in order to be able to make this distinction between sexes.

The perceptron worked by giving it many examples of portraits, including haircuts that are more unusual. After being trained, it was given a new dataset, with people it had never seen before, and it was able to classify the subjects correctly. However, that approach has disappeared.

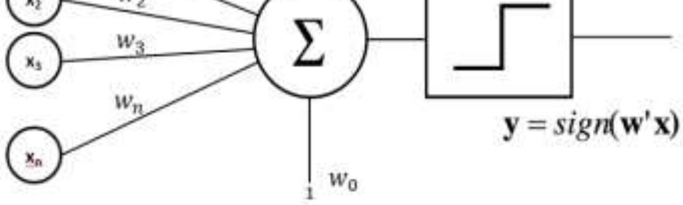


Fig. 1. The Perceptron

From 1970 through the late 1980s, there was a period called "AI Winter", signifying a low interest in research in the field of artificial intelligence. In 1956, the first multi-layer neural network architecture was created, but with a small number of neurons. The activation functions used were polynomial and the training methods were statistical.
In 1980, the first convolutional neural network appeared in Fukushima, inspired by the way the visual cortex works. Since 1989, the famous *backpropagation algorithm* started to be applied by Yann LeCun, which was used to handwritten zip code recognition [2]. He was also the one who had the idea to reduce the dimensionality behind the concept of autoencoder.
Since the 1990s, neural networks have become popular and dominant, mainly due to the discovery of the backpropagation algorithm for multi-layer neural networks.
In 1997, computer scientists Sepp Hochreiter and Jürgen Schmidhuber created the gradient-based method called "Long short-term memory", which was used for language recognition [3]. Around the 2000s, kernel methods gained interest, largely due to the instability in neural networks. Around 2010, neural networks began to become popular again, due to the *deep learning concept*. In 2011, Google invested in artificial intelligence for a voice command project for smartphones that used the Android operating system. In 2016, there was an increase in accuracy of approximately 60% for translations performed by Google. Nowadays, many companies use deep learning. Deep Learning finds applications anywhere from self-driving cars to fake news detection to even predicting earthquakes. Facebook, for example, uses these techniques for facial recognition, natural language processing, managing advertisements, and providing recommendations. Netflix also uses deep learning algorithms to suggest to users, certain movies and series based on other people's profiles with the same preferences.

1.1 Classification of machine learning

Machine learning is a discipline in which we define a program not by writing it entirely ourselves, but by learning from data. Researchers believe that machine learning is the best way to make progress in the direction of artificial intelligence. In this context, machine learning refers to the identification of patterns and structures and making decisions based on observations extracted from the data received [4].
Traditional programming requires a computer on which to run a program with a dataset, so that the process provides valid output data, as can be seen in the figure 2.

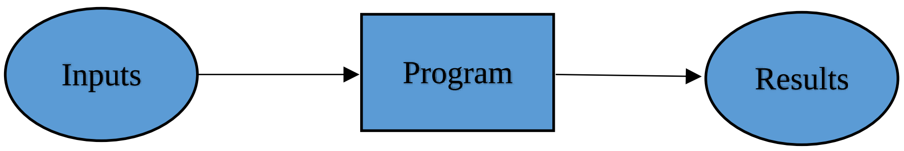


Fig. 2. Representation of a traditional program (adaptation after [5])

Machine learning requires a set of input data and sometimes output data that are interpreted by a computer, resulting, thus, in a program. Machine learning explores algorithms that can learn from the datasets so that computers learn without being explicitly programmed to do so. The simplified representation of machine learning can be seen in the figure 3.

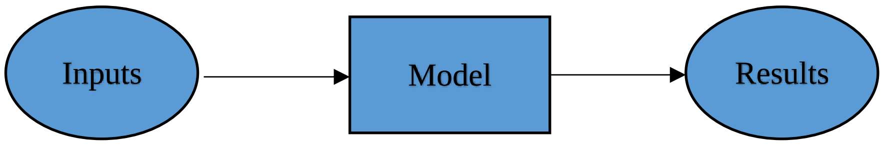


Fig. 3. Machine learning model (adaptation after [5])

In 1959, Arthur Lee Samuel, an American pioneer, in the field of machine learning defined it as the “field of study that gives computers the ability to learn without being explicitly programmed”. Kevin Murphy, author of the famous work "Machine Learning: A Probabilistic Perspective", defined machine learning as a series of algorithms that automatically detect structures and patterns in datasets [6].

2 Literature Review

There are several methods that can be used to detect fraudulent transactions which they were proven to be effective such as: *decision trees, genetic algorithms,* and *clustering techniques* or *neural networks*.

The idea of similarity tree was developed based on the idea which was used in the case of decision trees. Similarity trees are recursively defined, starting from labelling nodes with attribute names, labelling margins with attribute values that correspond to a required condition, and leaf nodes that represent a proportion of the ratio between the number of transactions that comply with the conditions and the total legitimate transactions [7]. This graphically illustrated method is very easy to understand and to put into practice. The less pleasant part could be the need to check each feature of each transaction in order to be categorized. On the other hand, it turned out that the similarity trees had very good results in other types of problems to detect fraud.

In 2000, an algorithm recommended by P. Bentley to establish logic-based rules for dividing suspicious and unsuspicious transactions into different categories was based on genetic algorithms. The experiment described in the study contained a database of 4,000 credit card transactions and 62 fields. Like the classification and similarity trees, the dataset was divided into training set and test set. All types of rules that could emerge from the available fields were tested. The best was the rule with the highest predictability. The method used by them proved to be highly efficient in the field of real estate insurance and was considered a possible effective way to detect credit card fraud [8].

A year earlier, in 1999, a specialized algorithm model was developed to predict suspicious behavior. The original part used in the research is brought by the fact that the model is evaluated based on a cost model, while in other studies the evaluation is used based on prediction rates and error rates [9].

In 2000, Wheeler and Aitken developed the idea of combining several algorithms in order to maximize predictive power. They wrote an article explaining diagnostic algorithms, probabilistic curve algorithms, as well as best fit, density selection, and negative selection algorithms. Following their extensive investigation, an adaptive diagnosis algorithm combining several neighbourhood-based and probabilistic algorithms was found to have the best performance, and these results indicate that an adaptive solution can provide fraud filtering and case ordering functions for reducing the number of final-line fraud investigations necessary. These algorithms can be further improved by using additional diagnostic algorithms in order to make decisions in exceptional cases or to calculate the level of confidence [10].

Regarding the approach to fraud detection using clustering techniques, Bolton and Hand suggested in 2002 two techniques to solve the problem of behavioural fraud [11]. Group analysis helps to identify accounts that behave differently at a certain time compared to other transactions recorded on the same card at different times in the past. Thus, the different behavior leads to the detection of fraud. Suspicious transactions cause the accounts from which they were debited to be marked. Following this filtering, those cases are investigated in more detail. It is assumed that if the account holder attached to the card has a certain behavior and a pattern of making payments, it is unlikely that he/she will behave completely differently only once.

The approach of breakpoint analysis is different from that presented by the previous method. Instead of tracking the account attached to credit frameworks, each transaction is tracked for each credit card separately. An example of such a suspicious incident would be the sudden trading of a large sum of money or a high frequency of use.

Another interesting approach to solve the problem of fraudulent transactions is given by neural networks.

In 1977, Gilles Dorronsoro and other collaborators developed an accessible system based on a neural classifier. The data, however, had to be grouped according to the type of account [12]. Concepts of similar notions were also found in the studies made by E. Aleskerov and B. Freisleben, in the paper called "Card Watch", related to a database mining system, which is based on neural networks.

In 1995, K. Leonard spoke about the systems of expertise used in credit card fraud with the help of the credit card [13]. In 1996, researchers Ezawa and Norton applied a technique for detecting fraud in the telecommunications industry, using Bayesian neural networks [14].

Four years later, in 2000, Maes and a small team applied techniques based on Bayesian neural networks to detect bank card frauds [15]. However, regardless of how they are detected, the characteristics of the transactions are the same. The number of real transactions will always be the majority and the model either has to deal with this variable distribution or the data needs to be rebalanced.

3 Methodology

A major problem we are facing this century, both as far as companies and individuals are concerned, is fraud using cards, especially credit cards.

Two types of fraud may occur: through the physical presence of the card (approximately 27% of all cases) and through the physical absence of the card (approximately 73% of cases).

The first category includes frauds at ATMs, POS using cloned cards, which contain stolen data of victims' cards or stolen and lost cards.

Representatives of the European Central Bank stated in an official report in 2018 that fraud without physical possession of the card has increased significantly in direct proportion to the development of online commerce. Fraudulent payments online, by phone or even by mail represent three quarters of the total.

With over 270,000 reports, credit card fraud was the most common type of identity theft last year and more than doubled from 2017 to 2019. Almost 165 million records containing personal data were exposed through data breaches in 2019.

Of the more than 3.2-million fraud cases reported to the Federal Trade Commission (FTC) in 2019, identity theft accounted for 20.33% of cases and was the most-common type of fraud [16].

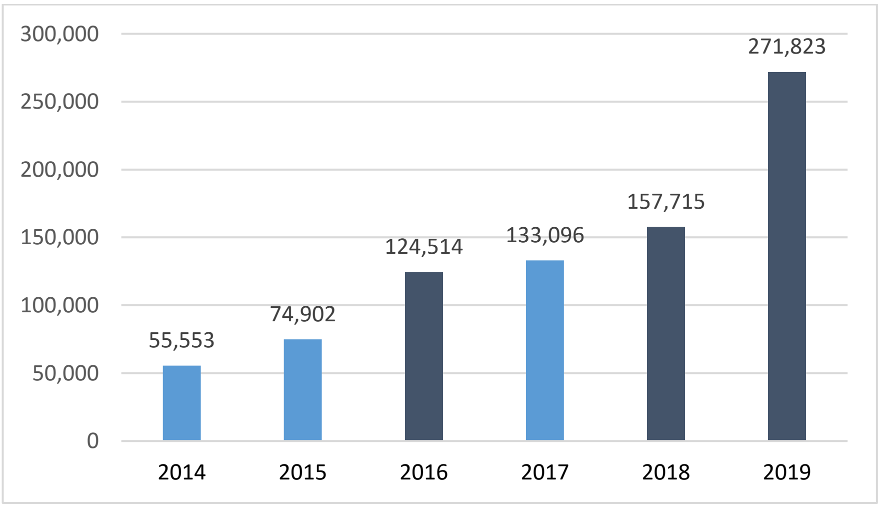


Fig. 4. Credit card fraud reports in the US by year (author processing based on [16])

Credit card fraud has been steadily increasing over the years, but it exploded in 2019, with the number of reports increasing by 72.4% from 2018.

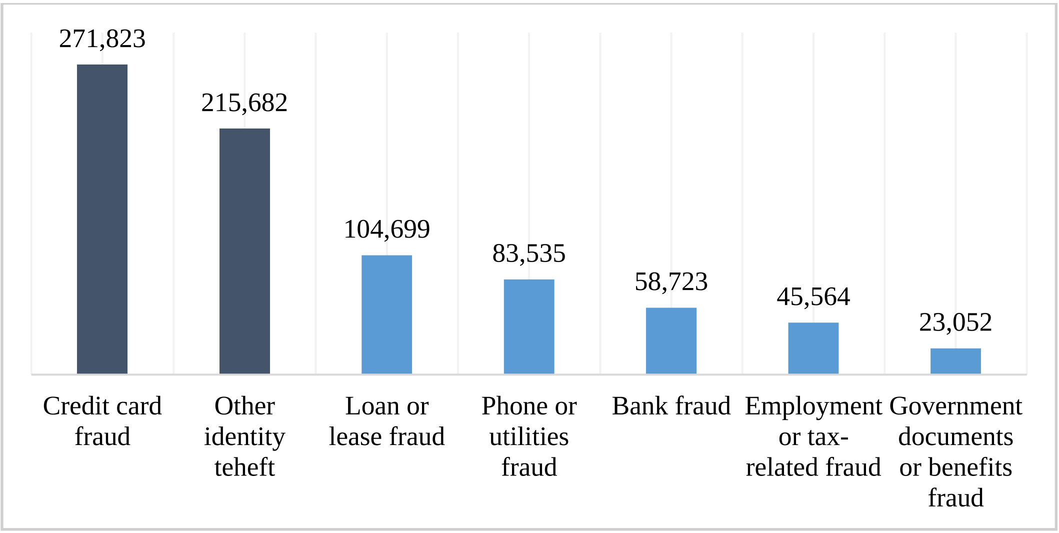


Fig. 5. Graph of identity fraud reports in 2019 (author processing based on [16])

As we can see in the figure above, in 2019, credit card fraud is the most common and popular identity theft, accounting for about 33.84% of all existing identity theft reports.
It is well known that thieves can target and obtain the personal information of the masses through frauds, use of personal information or even theft. They can use personal information such as birthday, address or PIN to retrieve from existing bank accounts. They can also open new accounts or even obtain loans on behalf of victims.
It has been proven that imposter scams come from a variety of different services. According to the Consumer Sentinel Network in 2019, most consumers affected by fraud were contacted by imposter scams by phone. Estimates of losses are about $ 493 million due to this type of fraud.

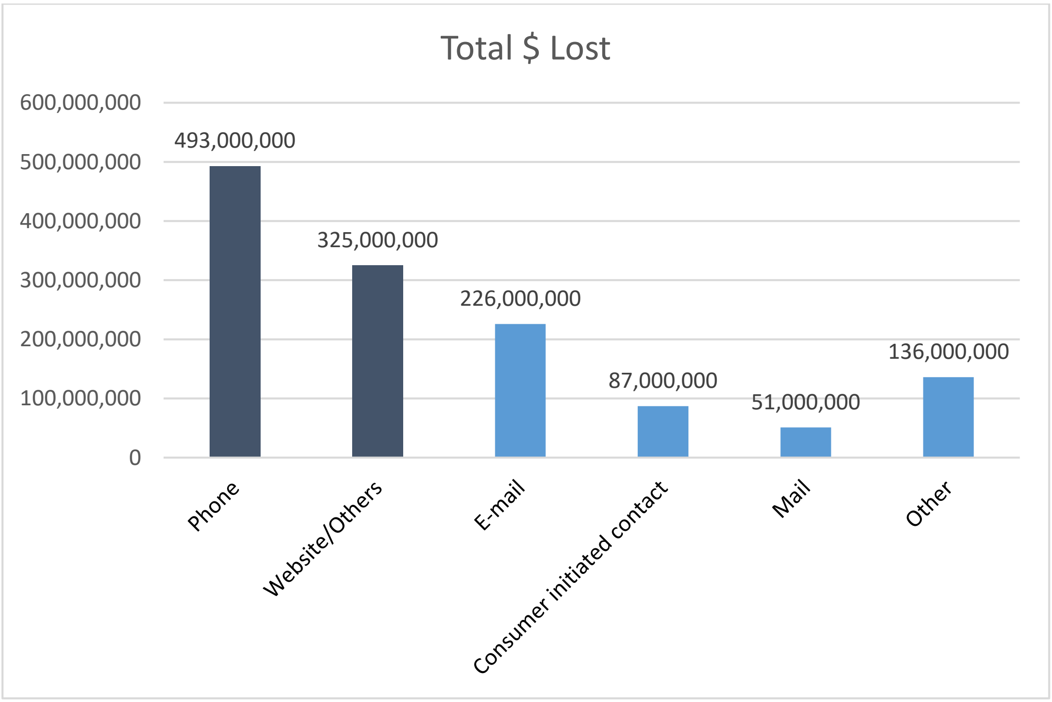


Fig. 6. Amount lost by contact method (author processing based on [16])

4 Implementing the Solution

Why Deep Learning and Why Now? To answer that question, we actually have to go back and understand traditional machine learning at its core first. Traditional machine learning algorithms typically try to define a set of rules or features in the data and these are usually hand engineered and because their hand engineered often tend to be brittle in practice. The key idea of deep learning is that we will need to learn the features just from raw data, so we're going to just take a bunch of images of faces and then the deep learning algorithm is going to develop some hierarchical representation of first detecting lines and edges in the image using these lines and edges to detect corners and eyes and mid-level futures like eyes, noses, mouths, ears then composing these together to detect higher-level features like jaw lines side of the face etc. which then can be used to detect the final face structure. Why now? The fundamental building blocks of deep learning have existed for decades and they are under underlying algorithms for training these models have also existed for many years so why are we studying this now?

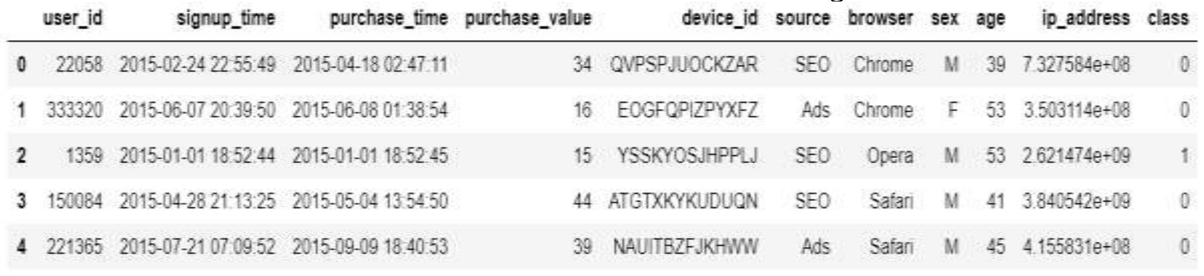
For one data has become much more pervasive. We are living in the age of big data and these algorithms are hungry for huge amounts of data to succeed. Secondly, these algorithms can benefit tremendously from modern GPU architectures and hardware acceleration that simply did not exist when these algorithms were developed and finally due to open-sources tool boxes like TensorFlow and Pytorch building and deploying these models has become extremely streamlined.

Python language appeared, according to official documentation, in the early 1990s, and is a multi-platform that can run on both Linux/Unix, Windows, Mac OS, and Raspberry Pi. This language was created by Guido van Rossum in the Netherlands. Python is actually a successor to the ABC language. Even though Guido van Rossum is the main author, many other people have contributed to the development of the language [17].

Python 3.9.1 is the newest major release of the Python programming language and was released on December 7, 2020. All versions released are open-source. Python has a very powerful standard library. It also has many other libraries used in data analysis such as: Pylearn2, Pybrain, MontePython, Pattern, Hebel and MILK. Other more common libraries are NumPy, Pandas, PyMongo, Matplotlib, Scikit-learn. The Python language can be run on several developed media (IDEs) such as: Netbeans, Eclipse, Thonny, PyCharm. It also provides support for connecting to databases [17].

Tabular modelling takes data in the form of a table (like a spreadsheet or CSV). The objective is to predict the value in one column based on the values in the other columns. The data used to develop the solution are in csv format "*date_frauda.csv*" and the table contains 151,114 rows and 11 columns. The data represent real recorded transactions.

The paper aims to understand and identify patterns in normal user behavior and it is vital to use meaningful and relevant features. Thus, the significant characteristics that we will use to create the model are user id, value purchased, device id used, and the source through which the transaction was made: through advertisements, by own access, SEO (Search Engine Optimization), browser used, gender, age and IP address. There is also the class feature, which decides whether the transaction is fraudulent. The structure of the data used can be seen in the Figure 7.



	user_id	signup_time	purchase_time	purchase_value	device_id	source	browser	sex	age	ip_address	class
0	22058	2015-02-24 22:55:49	2015-04-18 02:47:11	34	QVPSPJUOCKZAR	SEO	Chrome	M	39	7.327584e+08	0
1	333320	2015-06-07 20:39:50	2015-06-08 01:38:54	16	EOGFQPIZPYXFZ	Ads	Chrome	F	53	3.503114e+08	0
2	1359	2015-01-01 18:52:44	2015-01-01 18:52:45	15	YSSKYOSJHPPLJ	SEO	Opera	M	53	2.621474e+09	1
3	150084	2015-04-28 21:13:25	2015-05-04 13:54:50	44	ATGTXKYKUDUQN	SEO	Safari	M	41	3.840542e+09	0
4	221365	2015-07-21 07:09:52	2015-09-09 18:40:53	39	NAUITBZFJKHWW	Ads	Safari	M	45	4.155831e+08	0

Fig. 7. Representation of the data

Due to the fact that each person has a certain pattern of making payments, transactions made by fraudulent users will have a different pattern than the ones the model learned by training. Using the significant features presented above I want to process them so that I can use them to create the neural network. Although there are many supervised learning methods (decision trees, random forest), which have no problem in using categorical variables, neural networks need numerical variables. The optimal method to transform categorical variables into numeric variables is the one-hot encoding technique from the sklearn.preprocessing library [18].

In the official documentation, provided by sklearn, the explanation is to encode the categorical integers using a one-of-K-one-of-K scheme. In other words, this is a process by which categorical variables are converted into a form that can be used in machine learning algorithms. To illustrate in a simplified way how encoding works, we will assume that we want to transform only the column "Browser used" from categorical variables.

The table 1 shows the initial form to be processed.

Table 1. Table illustrating categorical variables

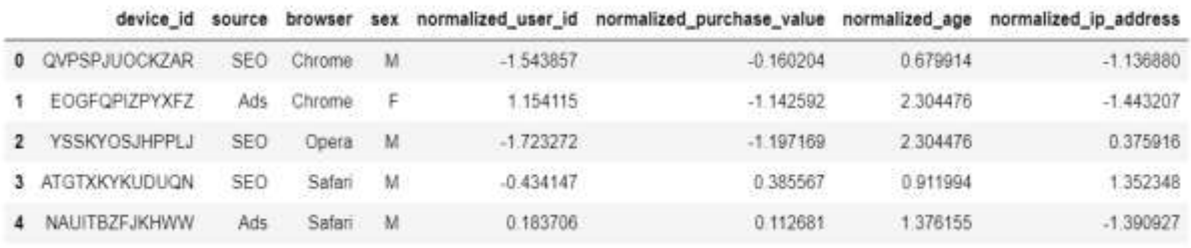
Id device	**Browser**
22058	Chrome
333320	Opera
1359	Safari
150084	Firefox
360585	Chrome
79203	Chrome
31383	Safari
159842	Microsoft Edge

After processing with one-hot encoding, the table below using binary values resulted. A value of 0 indicates that the value exists, and 1 means no value. In addition, new columns have appeared for each distinct value existing in the old table, on the column of the search engine (browser) used.

Table 2. Table illustrating the transformation from categorical variables using one-hot encoding

Id device	**Chrome**	**Opera**	**Safari**	**Firefox**	**Microsoft Edge**
22058	1	0	0	0	0
333320	0	1	0	0	0
1359	0	0	1	0	0
150084	0	0	0	1	0
360585	1	0	0	0	0
79203	1	0	0	0	0
31383	0	0	1	0	0
159842	0	0	0	0	1

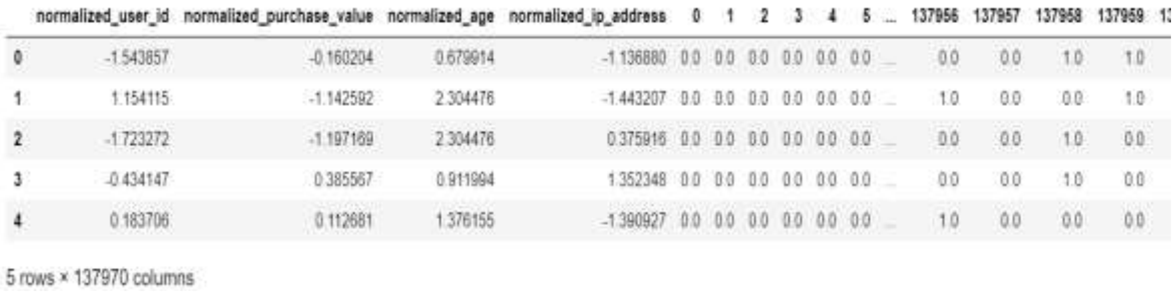
We can observe that, as the number of distinct values of the categorical variables increases, the complexity of the new objects that will have a significantly higher number of columns also increases.
Returning to the data processed in the project, first, we will divide the data into 2 objects such as "*pandas.core.frame.DataFrame*". X contains all the columns in the csv, except the "class" column and Y contains only the "class" column.
Next, we will process the X dataframe. For the columns whose values were of numeric type, we applied the normalization of the data distributions using StandardScaler from the package "*sklearn.preprocessing*". The idea used behind the StandardScaler technique is to transform the analysed data so that the distribution has the average value 0 and the standard deviation 1. In our case, this will be done according to the characteristics, thus independent for each column. This step is a very important one because we want to start from the premise that we use a normal distribution. I want to normalize the numerical data because I want the values of each column to have the same importance in the analysis. On the example of the studied dataset, we cannot leave the "user_id" column to have values of tens of thousands and hundreds of thousands, and the "purchase_value" column to have values of tens or hundreds of units of measurement. The values modified because of the normalization process that can be viewed in the figure 8.



	device_id	source	browser	sex	normalized_user_id	normalized_purchase_value	normalized_age	normalized_ip_address
0	QVPSPJUOCKZAR	SEO	Chrome	M	-1.543857	-0.160204	0.679914	-1.136880
1	EOGFQPIZPYXFZ	Ads	Chrome	F	1.154115	-1.142592	2.304476	-1.443207
2	YSSKYOSJHPPLJ	SEO	Opera	M	-1.723272	-1.197169	2.304476	0.375916
3	ATGTXKYKUDUQN	SEO	Safari	M	-0.434147	0.385567	0.911994	1.352348
4	NAUITBZFJKHWW	Ads	Safari	M	0.183706	0.112681	1.376155	-1.390927

Fig. 8. Representation of normalized numerical data

Then, like the example presented in the tables above, in the project, one-hot encoding was applied for the columns: "device id", "source", "SEO", "browser used", " "sex" and "IP address".
An alternative to encoding would have been Label Encoding, but this is not an effective method, as it provides numbers from 0 to the number of distinct elements. In this way, in the training phase, the model could consider, in the process of identifying the rules, that any X_i index is more important than a X_{i-1} index. Therefore, to avoid creating such rules, we chose to use one-hot encoding.
Finally, after these steps, we can see that instead of 11 columns, as we had initially, 137970 columns appeared. In the figure below we can see the first 5 lines. The data is mapped as a *DataFrame* object named *X_final*.



	normalized_user_id	normalized_purchase_value	normalized_age	normalized_ip_address	0	1	2	3	4	5	...	137956	137957	137958	137959
0	-1.543857	-0.160204	0.679914	-1.136880	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	1.0
1	1.154115	-1.142592	2.304476	-1.443207	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	1.0
2	-1.723272	-1.197169	2.304476	0.375916	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0
3	-0.434147	0.385567	0.911994	1.352348	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0
4	0.183706	0.112681	1.376155	-1.390927	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0

5 rows × 137970 columns

Fig. 9. Representation of the transformation of categorical variables into binary numbers

The next step is to divide the datasets X and Y. X represents the input data, and Y the output data, labelled, which determines whether the transaction is fraudulent or not. The data is divided into X_train, Y_train, X_test and Y_test. This step is done using "*train_test_split*" from the "*sklearn.model_selection*" library. As a rule, about three-quarters of the data is used to train the model, and a quarter to test the model. Initially, we divided the data into 70% for training and 30% for testing. However, due to the fact that there are 151,112 rows and 137,970 columns, we did not have a computer powerful enough to perform the training on the entire dataset, so we used 3,000 transactions for training and 2,000 for testing out of the total of 151,112 transactions. The next step involves transforming the X_train, Y_train, X_test, and Y_test dataframes into "*numpy*" strings.

5 Creating the Neural Network

For this stage, we will use *deep learning*, which is just a modern area in the more general discipline of machine learning. The process encountered in deep learning methods will automatically extract the significant features from the large volume of data in order to make connections.

A deep learning neural network is essentially an artificial neural network with several hidden layers, at least three layers, including the input and output ones. Each layer can have its own number of neurons. Deep learning neural networks, whether in a linear or nonlinear relationship, use mathematical functions to obtain output data by processing input data. The model created within the project will obtain probabilities for all output data after passing through all the layers. Each hidden layer in neural network extract the implicit information of the features. Hidden layers in the neural network capture more and more complexity with every layer by discovering relationships between features in the input. Therefore, complex data can be modelled with fewer units [19].

The network will be trained by giving it input and output data. The information considered the most relevant for solving the classification problem will be extracted. These important features will be denoted $x_1, x_2, ..., x_{137970}$. x_1 is the user id, x_2 is the transaction value, x_3 is the age, x_4 is the ip address, and so on. This data will form the perceptron input dataset. This is the basic unit of any neural network. The inputs will be multiplied by weights. We will note the weights: $w_1, w_2, ..., w_{137970}$. In official documentation, the bias is marked with b. The bias is actually a constant, an input value of each neuron. The output data of the neural network will be of binary type: 1 or 0.

The idea behind training a neural network is to discover those optimal values $w_1, w_2, ...,$ w_{137970}, and b. These values must be optimal for the entire dataset and not just for a transaction. After processing the data, the neural network model is created. For this part, we need the Keras library. Keras is an open-source library specialized in neural networks, written in the Python language. It can run on TensorFlow, R, Microsoft Cognitive Toolkit, PlaidML or Theano. Keras contains countless blocks of code used in neural networks such as activation functions, optimizers and layers to simplify the process of creating a network. The code is kept on GitHub, and is well documented. It also provides support for recurrent and convolutional neural networks.

Another useful aspect is the possibility for users to develop models on mobile devices with Android and iOS operating systems or even the web via the JVM. For the model part, we need Sequential, and for the network layers we need Dense and Dropout technique.

In Keras one can create a sequential pattern. Keras defines the sequential model as a linear stack of layers. When calling the Sequential constructor, we entered a list of two Dense objects, a Dropout object, and three more dense objects. Dense objects are actually the most basic types of layers and connect each value of the input with each value of the output of the layer. In the dense object constructor, there is the *input_dim* property, which corresponds to the number of columns in the input data size. It is necessary to define it only for the first layer because you will know how to determine the next ones without the need to specify them explicitly. Thus, for the first layer we assigned to the input_dim property the value 137970, meaning the number of columns related to the shape of the *X_train* object. Therefore, the first layer will have 137970 nodes. The last layer, the output layer, will have a single node.

We use dense layers, because a simple linear regression cannot be enough. Thus, we concatenate different perceptrons into more complex networks. As you can see in the image below, the input data are x_1, x_2, ..., and the weights we want to train are w_{11}, w_{12}, w_{21}, w_{22}, etc.

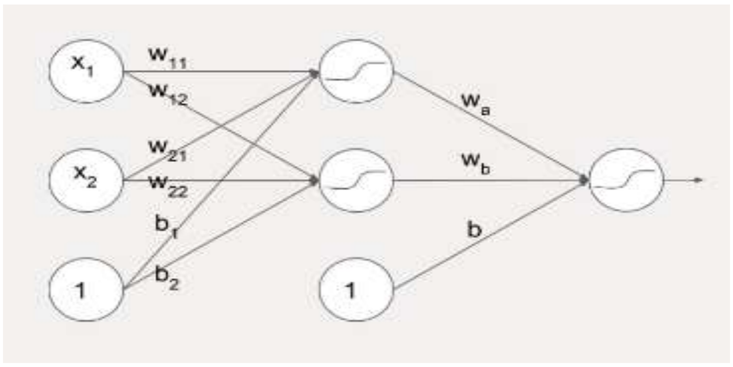
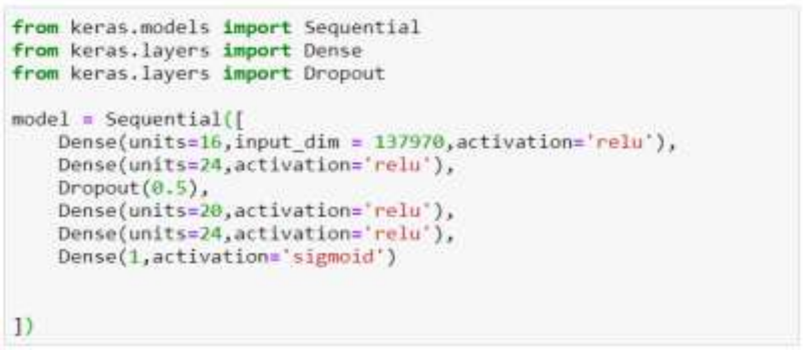


Fig. 10. Complex neural network (using dense layers)

Then, the perceptrons are grouped together, at the same level, connecting the output of one layer with the input of the next layer. In this way, a complete architecture can be created. To get more accurate results, we chose to use 5 layers, and after the first 2 layers the dropout was applied. Dropout is a regularization technique that was introduced in 2012 by Geoffrey E. Hinton et al. [20]. The basic idea is to change randomly some activations to zero at training time. This makes sure all neurons actively work toward the output. Simply put, dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons, which is randomly chosen.

The neural network model will take the form below:



```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

model = Sequential([
    Dense(units=16,input_dim = 137970,activation='relu'),
    Dense(units=24,activation='relu'),
    Dropout(0.5),
    Dense(units=20,activation='relu'),
    Dense(units=24,activation='relu'),
    Dense(1,activation='sigmoid')


])
```

Fig. 11. Neural network model

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neurons ‘input is relevant for the model's prediction.

For the first four layers, I chose to use the ReLU activation function. The name is an acronym for "rectified linear unit" and it is the most widely used activation function f(x) = max (0, x) and gives an output of x if x is positive and 0 otherwise.

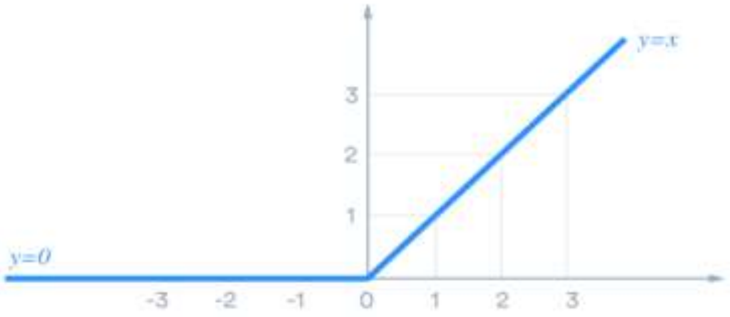


Fig. 12. Graphical representation of the ReLU function

For the last layer, I use the Sigmoid activation function, because this function is used for models where we have to predict the probability as an output (the transaction may or may not be fraudulent).

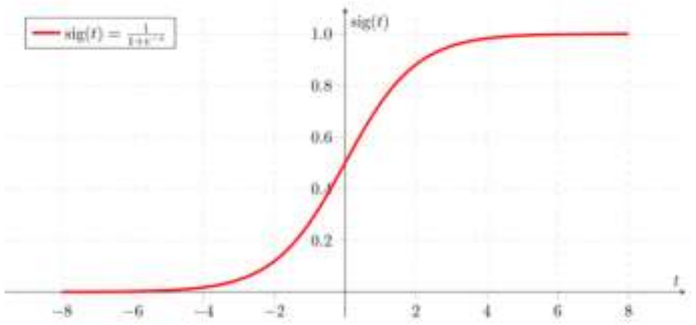


Fig. 13. Graphical representation of the Sigmoid function

The Sigmoid function is similar to the step function, which means that the output of each perceptron is 0 or 1, but it can take any values in the range [0; 1]. There is a wide range of sigmoid functions, which also includes hyperbolic and logistic tangent functions. In general, we can consider any function with an "S" shape as sigmoid. The Sigmoid function is concave for values greater than 0, and convex for values less than 0.

Feed forward is the process of calculating output data from input data in a neural network and it is also the first step in the process called *backpropagation*. This step will also make the prediction for the input dataset. We define a set of inputs to that neuron as $x_1, x_2 \ldots\ldots, x_m$ and each of these inputs have a corresponding weight $w_1, w_2\ldots\ldots, w_m$. We multiply them correspondingly together and take a sum of all of them; then we take this single number that's summation and we pass it through a nonlinear activation function and that produces our final output y. We also have what's called a bias term (b) in this neuron and the purpose of the bias term is to allow shifting the activation function to the left and to the right regardless of the inputs. The resulting value of the output will be compared with the value of the labels, always present in supervised learning. Having the results of the expected output and the calculated output we can check if the prediction made was correct. If the results do not match, then we use the backpropagation for the error across the neural network. In this way, we can figure out why the error occurred and it can be corrected. Thus, we will identify the *weights* that led to the occurrence of errors, in order to be rectified. The next feed forward process will have a better result than the previous one. The process will be repeated several times until the results are satisfactory. To do this, where used *epochs*. An *epoch* is one complete pass through the dataset. Within the project, the process has been iterated for 5 epochs. The problems that can be encountered are overfitting and *underfitting*. Overfitting, means that the model fits the data very well when we provided a set of data containing a lot of features.

The model tends to memorize the data, rather than to learn from it, which makes it unable to predict the new data. A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough.

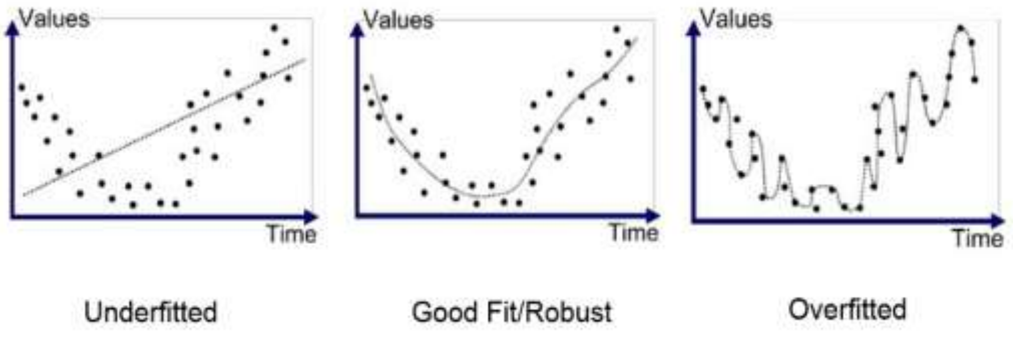


Fig. 14. Graphical representation of underfitting and overfitting [21]

To avoid overfitting, we decided to use the Dropout regularization form. The main idea is to remove neural components (outputs) randomly from a layer of the neural network. It is similar to the concept of bagging. The purpose of its use in the paper is to prevent overfitting and co-adaptation. Co-adaptation means that some neurons are highly dependent on others. If those independent neurons receive “bad” inputs, then the dependent neurons can be affected as well, and ultimately it can significantly alter the model performance, which is what might happen with overfitting. Therefore, the simplest solution to overfitting is dropout, and works as followed: each neuron in the network is randomly omitted with a probability between 0 and 1. It is used a hidden neuron probability of 0.5. When the neurons are eliminated, the neuron inputs decrease as the weights increase to compensate. After training, the weights will be scaled for a good network prediction. Then the weights are multiplied by the *dropout* rate. It is used dropout as a model because many studies has been shown that it leads to significantly better losses compared to many other types.

6 Model Training and Testing

In 2017, machine learning specialist Jason Brownlee published an article on the *Adam* optimization algorithm. The differences given by the optimizers are related to the learning time of the model. Good results of a model can be obtained in a few minutes, a few hours or a few days, depending on the choice of the optimizer. This optimizer is the extension of the stochastic gradient descent (SGD). The term gradient descent literally means descending a slope to reach the lowest point on that surface. The goal is to find the value of x for which f (x) = y is minimal. Jason Brownlee described the stochastic gradient descent as an iterative algorithm that has a random point in a function as a starting point and which descends until it finds the minimum value allowed by that function. [22]. The learning rate remains constant for each weight update.

Returning to the Adam optimizer, it can be used to replace the classic stochastic gradient descent procedure to iteratively improve the weights of the data training network. Advanced learning specialist Diederik P. Kingma said at a conference in Toronto, in 2015, that the name Adam is not actually an acronym, but is "derived from adaptive moment estimation" [23].

The advantages of using the Adam optimizer are computational efficiency, ease of implementation, low memory requirements, diagonal invariability of gradient rescaling, efficiency recorded on large volumes of data. Compared to the classical stochastic gradient descent this optimizer combines the advantages of two other gradient descent extensions, namely AdaGrad and RMSProp.

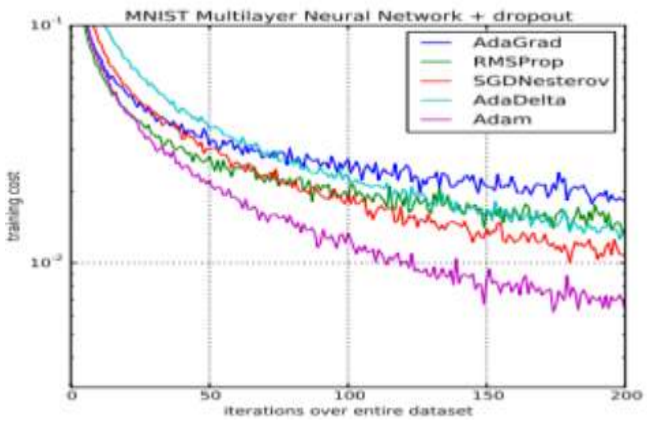
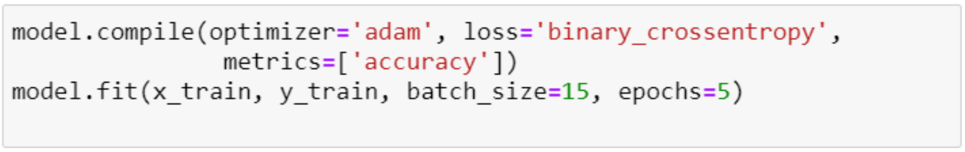


Fig. 15. Neural networks using dropout stochastic regularization [23]

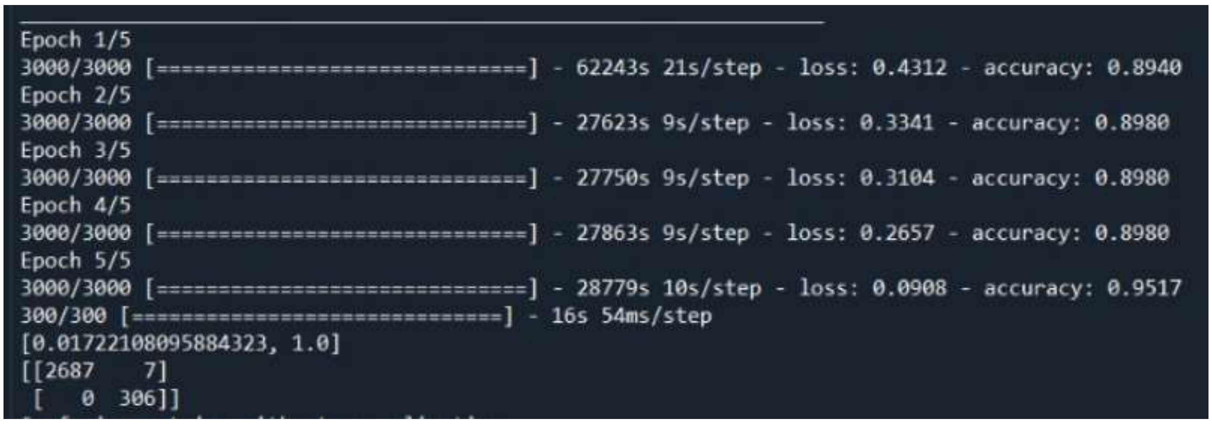
Adam optimizer and binary cross entropy loss function were used during model training. The binary cross entropy loss function is used because the result of the model is a binary one (the transaction may or may not be fraudulent).
The 3000 data used in the test set were covered in 5 epochs, with a batch of 15. The *batch* size refers to the number of samples that will be sent to the neural network at a given time. Thus, the 3000 data were divided, segmented into 15 batches. An epoch is one complete pass through the dataset. We have 3000 data representing the transactions with which we want to train the network to identify the type of transaction. The batch size is 15. This means that 15 transactions will be grouped in one batch at a time to the network. Given that an epoch is one complete pass through the dataset it will take 3000/15 = 200 batches to contain all the data of an epoch.
In general, the larger the batch size, the faster the model will complete an epoch during training.
This is because, depending on the computational resources of the computer, it will be able to process multiple samples simultaneously. However, even if the computer processes faster, a larger number of batches could alter the quality of the model. Thus, depending on the model result, the batch size can be adjusted. Then, calling *model.fit*, the training will start according to the given specifications.



```
model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=15, epochs=5)
```

Fig. 16. Model training

Any optimization problem must have a goal. The goal is to minimize the value of the loss function. Thus, the optimizer will set the weights so that this loss function has a value as close as possible to 0. The loss refers to the error obtained by comparing the predicted value with the expected one. If the values do not match, it turns out we have an error. Thus, it is normal to want the loss to be as small as possible. If the model fails to identify the transaction label correctly, at the next iteration, it will change the weights so that the same data as in the past epoch is closer to a better result. In this way, the model will learn. After training the model, a high accuracy can be seen in the figure below. From the first epoch, where the accuracy was 89%, to the last epoch, it increased to 95%. Accuracy has increased significantly while loss has decreased significantly. We can also see that the error was very large, namely loss = 43%. In the last epoch, the error decreased to the value of loss = 9%.



```
Epoch 1/5
3000/3000 [==============================] - 62243s 21s/step - loss: 0.4312 - accuracy: 0.8940
Epoch 2/5
3000/3000 [==============================] - 27623s 9s/step - loss: 0.3341 - accuracy: 0.8980
Epoch 3/5
3000/3000 [==============================] - 27750s 9s/step - loss: 0.3104 - accuracy: 0.8980
Epoch 4/5
3000/3000 [==============================] - 27863s 9s/step - loss: 0.2657 - accuracy: 0.8980
Epoch 5/5
3000/3000 [==============================] - 28779s 10s/step - loss: 0.0908 - accuracy: 0.9517
300/300 [==============================] - 16s 54ms/step
[0.01722108095884323, 1.0]
[[2687    7]
 [   0  306]]
```

Fig. 17. Training results

7 The interpretation of the results

As can be seen in the confusion matrix related to the training model and testing model, were recorded some very good results.

In the model involved, out of 3000 transactions, of which 306 were fraudulent and 2694 were real, the model correctly identified all 306 fraudulent, and of real ones identified 2687. This means that it considered 7 non-fraudulent transactions as fraudulent. This means that in the percentage of (3000-7) / 3000 = 99.76% the classification was correct. In the model tested, out of 2000 transactions, of which 217 were fraudulent and 1783 were real, the model correctly identified all 217 fraudulent ones, and of the real ones identified 1778. This means that it considered 5 non-fraudulent transactions as fraudulent. This means that in the percentage of (2000-5) / 3000 = 99.75% the classification was correct.

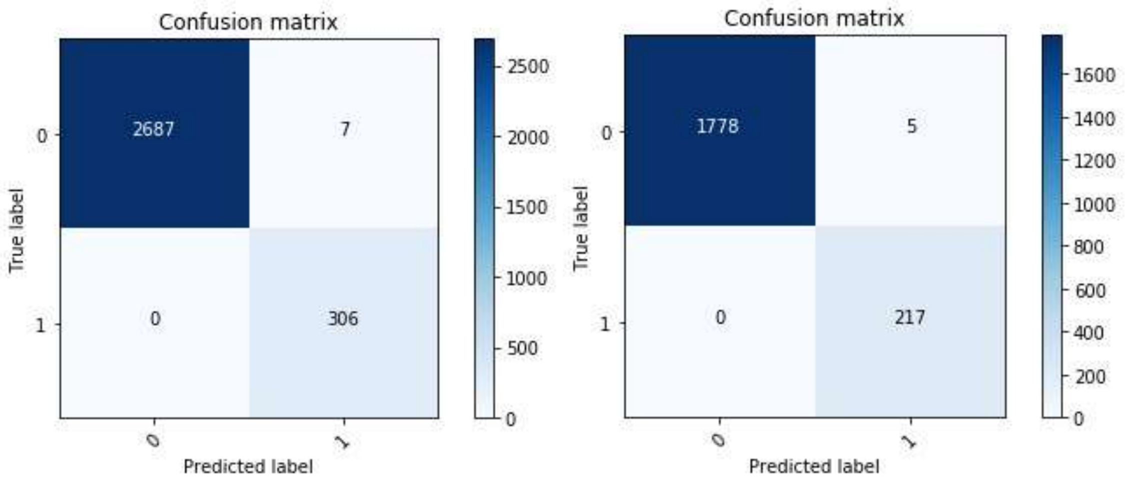


Fig. 18., Fig. 19. Confusion matrix of the training and testing model results

The resulting model is a good one, because the proportion of the correct classification is very high. It is also important to note that all fraudulent transactions have been correctly identified. In this way, even if certain payments will be falsely categorized as fraudulent, a person can check again manually to confirm or deny the model result. From my point of view, it is much safer for the model to make mistakes considering some transactions as frauds even if they are not, than to consider fraudulent transactions as real because, in the latter case, there would be either frauds that would remain unidentified, or all transactions that have been detected as real should be checked again by another method, as we cannot be sure which of them could be possible fraudulent.

In order to obtain good results, a method was applied to balance the dataset, so that there is no disproportion between the number of fraudulent transactions and the number of real transactions. The sampling techniques that have been used are *undersampling* and *oversampling* using the SMOTE technique that creates a new vector between two existing points. Another option could be to increase the number of layers so that the model is more complex, or to train over several epochs.

8 Conclusions

The model obtained in this paper can be used to be integrated into banks 'mobile applications in order to identify fraudulent transactions and to debit money from victims' credit accounts. The integration of mobile applications with Android and iOS operating systems is done using the TensorFlow Lite framework. Thus, the already trained model can be converted into a buffer using the converter provided by the platform maintained by Google. Then, the compressed file obtained with the *.tflite extension* will be uploaded to the application on the mobile device.

The integration will be done as follows: within the banking application, which already contains details regarding the transactions (the amount traded, the GPS location from which the transfer is made, the ID of the device from which the transfer is made, and much more) the trained model will be entered. An alternative would be to create the neural network from scratch after the process of storing the significant features offered by customers through their purchases, transactions and payments. Once there is enough information, the actual training can begin, and then new predictions can easily be made.

If a transaction is considered suspicious, the bank will be able to manually verify the transaction and in case of irregularities or suspicions, will contact the cardholder to confirm the operation before debiting the amount. In case of confirmation of a fraud, the card will be blocked, the amount of money will be blocked and customers will be satisfied.

We can see that the machine learning technique is an excellent tool in terms of information processing and data processing, but it does not have creativity, strategic thinking, intuition, value judgment and moral principles. That is why we will always need human resources.

References

[1] D. Li and Y. Di, *Artificial Intelligence with uncertainty.* Taylor & Francis, LLC, 2008.

[2] Y. LeCun, B. Boser, J. S. Denker and D. Henderson, “Backpropagation Applied to Handwritten zip code recognition”, *Neural Computation*, vol. 1, Winter 1989, pp. 541-551.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, November 1997, pp. 1735-1780.

[4] A. Burkov, *The Hundred-Page Machine Learning Book.* Andriy Burkov, 2019.

[5] J. Howard and S. Gugger, *Deep Learning for Coders with fastai and PyTorch*, O'Reilly Media, Inc, July 2020.

[6] K. Murphy, Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.

[7] A.I. Kokkinaki, “On Atypical Database Transactions: Identification of Probable Frauds using Machine Learning for User Profiling”, Proc of IEEE Knowledge and Data Engineering Exchange Workshop; 107-113, 1997.

[8] P. Bentley,”Evolutionary, my dear Watson: Investigating Committee-based Evolution of Fuzzy Rules for the Detection of Suspicious Insurance Claims”, Proc. of GECCO2000.

[9] W. Fan, M. Miller, S. Stolfo, W. Lee and P Chan., “Using Artificial Anomalies to Detect Unknown and Known Network Intrusions”, Proceedings of ICDM, pp. 123-248, 2001.

[10] R. Wheeler and S. Aitken, „Multiple Algorithms for Fraud Detection”. Knowledge-Based Systems, 13; 93-99, 2000.

[11] R. Bolton, and D. Hand, „Statistical Fraud Detection: A Review”, *Statistical Science*, vol. 17, no.3, pp. 235-249, 2002.

[12] J. Dorronsoro, F. Ginel, C. Sanchez and C. Cruz, „Neural Fraud Detection in Credit Card Operations”. IEEE Transactions on Neural Networks, 8; 827-834, 1997.

[13] K. Leonard, „The development of a rule based expert system model for fraud alert in consumer credit”, *European Journal of Operational Research*, 80; 350-356, 1995.

[14] K. Ezawa and S. Norton, „Constructing Bayesian Networks to Predict Uncollectible Telecommunications Accounts”, IEEE Expert, October 1996; 45-51.

[15] S. Maes, K. Tuyls, B. Vanschoenwinkel and B. Manderick, “Credit Card Fraud Detection using Bayesian and Neural Networks”, Proc. of the 1st International NAISO Congress on Neuro Fuzzy Technologies, 2002.

[16] Consumer Sentinel Network, Data Book 2019, Federal Trade Commission, January 2020. Available: https://www.ftc.gov/system/files/documents/reports/consumer-sentinel-neworkdata-book-2019/consumer_sentinel_network_data_book_2019.pdf

[17] P. Vothihong, M. Czyga, I. Idris, M.V. Persson, L.F. Martinez, *Python: End-to-end Data Analysis (Learning Path)*, Packt Publishing, May 2017.

[18] K. Potdar, T. Pardawala and C. Pai, “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers”, *International Journal of Computer Applications*, vol. 175, 7-9, 2017.

[19]M. Buckmann, „Deep Learning Model For Bank Crisis Prediction”. Available: https://swisscognitive.ch/2020/03/30/deep-learning-model-for-bank-crisis-prediction/

[20] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors”, 2012. Available: 1207.0580.pdf (arxiv.org).

[21] A. Bhande, “What is underfitting and overfitting in machine learning and how to deal with it”, 2018. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[22] J. Brownlee, „Better Deep Learning - Train Faster, Reduce Overfitting, and Make Better Predictions”, 2018. Available: https://machinelearningmastery.com/better-deep-learning/.

[23] D.P. Kingma, J.L. Ba, „Adam: A Method for Stochastic Optimization”. Published as a conference paper at ICLR 2015, https://arxiv.org/pdf/1412.6980.pdf.

Oona VOICAN has graduated from the Faculty of Cybernetics, Statistics and Economic Informatics at the Bucharest University of Economic Studies, Economic Cybernetics specialization. She followed a master’s degree in Business Analysis and Business Performance Control with the thesis *Data mining methods used in the banking system*. She worked 15 years in the banking system and now she is a senior adviser at the Ministry of Transportation, Infrastructure and Communications. Currently, she is a PhD Student in the field of Economic Informatics. Her scientific fields of interest include Data Mining, Big Data, Business Intelligence.